Dissertation

submitted to the

Combined Faculties of the Natural Sciences and Mathematics
of the Ruperto-Carola University of Heidelberg, Germany

for the degree of

Doctor of Natural Sciences

Put forward by

Lukas Kades

born in: Werneck, Germany
Oral examination: November 2nd, 2021

Deep Learning and Neuromorphic Computing in Quantum Chromodynamics and Beyond

Referees: Prof. Dr. Jan M. Pawlowski
Prof. Dr. Manfred Salmhofer

Deep Learning and Neuromorphic Computing in Quantum Chromodynamics and Beyond

Accompanied by the fast evolution of graphical processing units, there is a rapid development of deep learning methods with applications in almost all natural and applied sciences. Simultaneously, a growing interest is emerging around alternative, more energy- and time-efficient computing devices. Driven by these developments, we propose in this thesis several possible directions in the field of quantum chromodynamics and beyond. We start with the exploration of novel frontiers to perform scientific computing tasks on the spike-based BrainScaleS neuromorphic device. This includes numerical computations in statistical physics and the representation of entangled quantum states. We continue by establishing a new mathematical framework to tackle the so-called sign problem impeding a statistical analysis of many physical systems, including quantum chromodynamics at finite density. Dealing with the same problem, a machine learning-driven algorithm is proposed in the subsequent part of the thesis. Utilizing deep neural networks to recognize undiscovered structures and to get novel insights into physical data is a further direction pursued by employing methods from explainable machine learning and by proposing a new unsupervised training algorithm for generating lower-dimensional representations. The thesis concludes with a supervised learning framework for approaching the inverse problem of reconstructing spectral functions from Euclidean propagators.

Deep Learning und neuromorphes Rechnen in der Quantenchromodynamik und verwandten Anwendungsgebieten

Deep Learning Methoden wurden in den letzten Jahren durch die schnelle Entwicklung von Grafikkarten und der dadurch erhöhten Rechenleistung enorm vorangetrieben und finden in nahezu allen Bereichen sinnvolle Anwendungen. Gleichzeitig wächst das Interesse an alternativen, energetisch effizienteren sowie schnelleren Rechensystemen. An die Entwicklungen anknüpfend, beschäftigt sich diese Arbeit mit vielversprechenden Anwendungen im Bereich der Quantenchromodynamik und verwandten Gebieten. Die Arbeit beginnt mit der Untersuchung möglicher numerischer Berechnungen von physikalischen Systemen sowie der Darstellung von verschränkten Quantenzuständen in einem feuernden neuronalen Netz auf dem spike-basierten neuromorphen BrainScaleS System. Der zweite Teil der Arbeit stellt einen neuen mathematischen Formalismus für die Entwicklung numerischer Methoden für eine mögliche Lösung vorzeichenbehafteter Probleme vor. Das sogenannte Vorzeichenproblem behindert die Berechnung von vielen physikalischen Systemen, wie zum Beispiel Quantenchromodynamik bei endlicher Dichte. Im Anschluss an eine weitere Machine Learning-basierten Methode zur Behandlung dieses Problems, wird ein unüberwachter Trainingsalgorithmus für die Erstellung niedrigdimensionaler Darstellungen eingeführt. Ziel ist es, unter anderem mit der Hilfe von Explainable Machine Learning-Methoden, verborgene Strukturen in physikalischen Daten zu erkennen, um so ein besseres Verständnis des zugrundeliegenden Systems zu erlangen. Die Arbeit endet mit einem überwachtem Trainingsansatz, welcher das Ziel hat, das inverse Problem der spektralen Rekonstruktion von Euklidischen Propagatordaten zu lösen.

Table of Contents

CHAPTER 1

Introduction

1.1 Motivation and structure

Physicists aim to describe and understand the fundamental principles of nature. This includes our immediate environment and everything related to the world and the universe. Physical models, which are mostly simplified, abstract representations of nature, are utilized to reach this goal. To be able to develop such models as well as for a possible exchange and discussion about them, a framework is required, allowing for a decent and clear description of the respective physical theory. Mathematics represents such a framework. It possesses necessary properties facilitating a formulation and quantification of observations and phenomena in physics and other fields in terms of relations and numbers. Given theories are, if possible, verified and also motivated by experimental results.

Quantum chromodynamics (QCD) is the theory of strong interaction and describes the interplay of quarks and gluons within the Standard Model of particle physics. It is one of the four known fundamental interactions; besides gravity, the electromagnetic interaction and the weak interaction. Different scales in temperature and density of the particles are studied in the QCD phase diagram. It comprises a rich structure with respect to different phenomena and phases of matter such as the quark-gluon plasma, neutron stars, as well as the early universe and experiments of heavy-ion collisions or the confined phase where quarks are binding to hadrons.

The physical motivation of this work can be embedded into the computation of statistical properties in strongly correlated systems and beyond, or, more specifically, into an exploration of the phase diagram of quantum chromodynamics. This thesis discusses several approaches and novel methods to tackle problems related to the numerical computation of named systems.

The quantum statistical properties of a physical system are described by its partition function Z. In particular, observables and correlation functions can be computed according to the path integral

$$\langle \mathcal{O}(\phi) \rangle = \frac{1}{Z} \int \mathcal{D}\phi \, \mathcal{O}(\phi) \exp(-S(\phi)) \,, \tag{1.1}$$

where $S(\phi)$ denotes the Euclidean action of the considered system. The field $\phi := \phi(x)$ describes the state of the system in Euclidean spacetime x. In lattice physics, spacetime is discretized and the fields live on a d+1-dimensional hybercubic lattice, allowing a numerical computation of observables [12, 13].

If $S(\phi)$ is real-valued, the weight $Z^{-1}\exp(-S(\phi))$ can be interpreted as a probability measure. This analogy enables a numerical computation of correlation functions based on standard Monte Carlo techniques. Therefore, the computation of a Euclidean quantum field theory turns into a simulation of a statistical system coupled to a heat bath. Its properties can be accessed by computing expectation values of a stationary distribution generated by a stochastic process in some fictitious time. This approach is referred to as stochastic quantization [13–16].

Computing observables based on Eq. (1.1) comes with certain analytical as well as numerical challenges. The statistical representation of the path integral (1.1) becomes soon very high-dimensional for lattices and can, therefore, mostly only be accessed by numerical methods. In addition, the actual physics can only be observed in the continuum limit, obtained by an extrapolation of results of repeated computations at different lattice spacings. This entails a potentially very high computational cost for simulating a theory. A numerical evaluation of the Eq. (1.1) is additionally hindered by the so-called sign problem in many physical systems, as in QCD at finite baryon density, for example. The sign problem prevents an application of most of the existing numerical methods.

The high dimensionality combined with the complexity of such systems demonstrates the need for numerical computations and respective computing devices. Using computers in scientific computing dates probably back to the first computer itself. Since then, plenty of research areas emerged where numerical computations are used as a supportive tool or as the main approach to study the problem at hand with applications ranging from physics to biology, chemistry, engineering, computer vision, medicine and economics. Research in scientific computing is boosted by the accompanied development of numerical methods and the broad field of machine learning such as Monte Carlo methods which enable us to numerically study high-dimensional probability distributions. In particular, the field of deep learning has benefited from a constant improvement of the performance of graphic processing units (GPUs) within the last decades.

Besides GPUs, a large research frontier has formed with the goal to develop other computing devices, as von-Neumann computers are rapidly approaching fundamental physical limitations of conventional semiconductor technology. A strong focus is on an increased computational power, mostly achieved by parallel computing units. Therefore, research on alternative devices to GPUs and CPUs is also motivated by the on-going increasing demand for more efficient computing devices regarding computation time, energy consumption and scalability. A number of alternative computing architectures are currently being explored. Among them, neuromorphic devices [17–19], which take inspiration from the way the human brain works, hold promise to have a wide range of applications, in particular in machine learning and artificial intelligence [20–28]. Possible applications range from an effective implementation of artificial neural networks and further machine learning methods [29–34] over a better understanding of biological processes in our brains [35, 36] to the computation of interesting physical and stochastic systems [37–41].

The need for more efficient computing devices can also be accommodated by devices tailored to the numerical method in use. Respective tailored computing devices can resolve the high structural overhead resulting from the desired general usability of von-Neumann architectures and, instead, be optimized for the specific computing task. Bearing in mind the current occupancy rates of the world's largest super computers, lattice computations and neural networks represent two prominent candidates motivating a development of such tailored devices. In both cases, an on-chip implementation of the underlying algorithm results in an expected high-performance boost with respect to the computational efficiency. A strong focus on the implementation of neural networks on tailored devices is justified by its usefulness in almost every setting; not only in research, but also in industry, ranging from applications in self-driving cars, in the field of natural language processing

and computer vision, or, possibly, in smartphones for a fast analysis of image and text data. Tensor Processing Units (TPUs) are an example for such tailored devices. Specialized for neural networks, they are designed for an efficient computation of large matrix multiplications. The training of neural networks is based on this kind of computations and can be implemented in parallel, resulting in a highly improved and more efficient training.

In the first part of this work, we discuss potential implementations of numerical methods in the context of lattice field theories and neural networks on the spike-based BrainScaleS neuromorphic computing device [28, 42, 43]. The chip realizes fast analog dynamics with the potential to boost computationally expensive tasks. Hereby, our focus is on studying to what extent the neuromorphic system is tailored to scientific computations of physical systems. Besides an in-depth analysis of underlying dynamics, we investigate the capability of the neuromorphic chip to represent probability distributions in a reliable and numerically exact manner from both a theoretical as well as an experimental point of view.

Apart from a computation of physical systems on the BrainScaleS system, in the second part of the thesis, we present novel insights and numerical methods to overcome or milder the sign problem. The remainder of the thesis covers further approaches and pathways related to numerical problems in lattice physics and beyond, with a strong focus on the utilization of deep learning algorithms.

The thesis is structured based on the following main building blocks:

- A short reminder of important concepts of numerical methods in statistical physics and stochastic quantization as well as an introduction to the BrainScaleS hardware system (Chapter 2).
- Exploring interrelations and differences between the dynamics of numerical methods and of the neuromorphic computing devices BrainScaleS/BrainScaleS-2 to utilize the computational power of a parallel processing architecture for a more energy-efficient simulation on large scales (Chapters 3, 5, 4 and 6).
- Understanding and developing new numerical methods in order to tackle the sign problem, preventing a computation of the path integral (1.1) for complex actions by standard numerical methods in QCD and beyond. (Chapters 7, 8 and 9).
- Representation learning and the generation of embeddings of physical systems to gain a better understanding of the systems themselves and to discover uncovered structures in the underlying data supporting the development of new, more efficient, algorithms and approaches to easing lattice simulations and to overcome the sign problem (Chapters 10 and 11).
- Investigating the inverse problem of spectral reconstruction with deep neural networks as a black box universal inverse transformation tool (Chapter 12).

The common ground of these building blocks is their focus on method development and novel tools, in particular, also related to the BrainScaleS neuromorphic hardware device, for approaching mathematical and numerical challenges originating from the studied physical systems of interest. The utilized mathematical and statistical tools range from statistical physics to techniques of machine learning and deep learning. In the following, we give a more detailed introduction to the different approaches studied in this thesis by pointing out important concepts and by embedding the main building blocks into a broader context.

The human brain, neurormorphic computing and statistical physics

In comparison to human-made computing devices, the human brain has been developed, optimized and perfected by nature over the course of hundreds of thousands years of evolution. With a power consumption of 20W [44], it still outperforms all existing computing devices by order of magnitudes in terms of its computational power to energy consumption ratio.

Also with respect to its dynamics, the human brain works entirely different to von-Neumann architectures regarding memory as well as in the way information is transmitted, via action potentials, so-called spikes, in a neural network consisting of synapses and neurons. Inspired by the astonishing efficiency of the human brain and the accompanying high level of intelligence, artificial neural networks (ANNs) have been developed as a possible abstraction of a neural network in the brain. Signals are transmitted between neurons via trainable synaptic weights, introducing the concept of neural plasticity, and processed by an aggregation and a non-linear activation in each neuron [45–49].

A closer look at the comparison of ANNs and the neural networks in the human brain reveals that the two systems bear some significant differences. This concerns in particular the absence of spikes in ANNs. It raises the question whether there are even more effective abstractions of neural networks, incorporating similar learning techniques and the same computation power and energy-efficiency as the human brain. Spiking neural networks represent a possible answer to this question as the third generation of neural network models [50–58]. In contrast to ANNs, information exchange between neurons takes place via spikes, which are crucial for communication in the neural system of the human brain. The way information is represented via spikes is subject to the research area of neural coding [59]. Several neural coding schemes of spike patterns have been studied together with concepts of neural plasticity, the ability of the nervous system to reorganize and modify itself [60–62]. Examples are, rate coding, where information is encoded in the firing rate of a neuron or time coding, where information is transmitted by individual spikes; for more details see Refs. [51, 63–70].

Neuromorphic computing devices, such as the BrainScaleS project, help to unveil the potential of spiking neural networks for being the most effective abstraction of the human brain from an experimental point of view [18, 19, 71–73]. With respect to their implementation based on electric circuits instead of biological substrates, an illustrative analogy to neuromorphic computing devices is the construction of airplanes. The wing shape of birds has been identified as the most important and easiest to copy feature for a human-engineered prototype. The example demonstrates that certain key features of a working system can be sufficient to rebuild a system with similar properties with the advantage of a simpler and better realizable setup. Similarly, the spiking mechanism and the resulting neural representation in spike patterns might correspond to the essential feature of spiking neural networks. Therefore, spike-based neuromorphic devices have the potential for paving the way to get full access to the computational power of the human brain, but also, for supporting medical research with respect to the treatment of cognitive diseases or physical traumata.

In terms of its computational power and energy-efficiency, the BrainScaleS device [28] represents a perfectly tailored substrate for spiking neural networks and respective computational tasks in scientific computing based on this type of neural networks. In this work, we pursue a different direction by analysing to what extent the BrainScaleS device can be used for other kinds of scientific computations. We study this subject to a possible tailoring of already existing algorithms to the dynamics of the system, which is in strong contrast to the construction of a tailored device for a given algorithm. In particular, we focus on utilizing the BrainScaleS-2 chip [28], developed in the context of Europe's Human Brain Project [74], as a physical substrate to implement numerical computations. Besides the low energy consumption, this is motivated by the fast analog dynamics of

the chip which can be used to boost computationally expensive tasks. With respect to its biological counterpart, the time constants on the chip are smaller by a factor of thousand, resulting in an impressive speed-up for analysing neural dynamics [28].

The BrainScaleS-2 chip, inspired by structural and dynamical properties of the biological brain, emulates networks consisting of leaky integrate-and-fire (LIF) neurons, which are, in general, suitable for representing probability distribution [75–77]. The mixed-signal neuromorphic platform is centered around an analog core: neuro-synaptic states are represented as voltages and currents in integrated electronic circuits and evolve in continuous time. Its configurable connectivity of neurons allows us to explore various different network topologies, including shallow, as well as deep and densely connected ones. As a parallel computing platform, the chip in the long run has the potential to simulate large physical systems in a very energy-efficient way and, thus, to facilitate the analysis of problems that cannot be solved by existing other devices.

Due to the fixed physical structure and dynamics of a neuromorphic computing device, there are, in principle, two possible pathways for performing scientific computations. The first one refers to utilizing the given device for numerical computations where the dynamics of the underlying numerical method is incorporated in a one-to-one correspondence by the neuromorphic system. In this case, the realized algorithm and the hardware are fully compatible in the sense that the dynamics of the neuromorphic device implements the numerical method itself. The other pathway consists of investigating possible ways to either translate existing numerical methods on the neuromorphic hardware in a reasonable way or to look for dynamics of algorithms that are adjustable or at least partially related to the dynamics of the hardware. A drawback of the latter pathway is that one might introduce certain cumbersome routines or workarounds affecting the numerical and energetic efficiency of the device. Nevertheless, the benefits of a parallel computing platform with an overall low energy consumption are, in both cases, a good reason to evaluate the limits of these approaches. Both pathways have been studied in this thesis and are discussed in more detail in the following.

Concerning the first pathway for scientific computations on neuromorphic hardware devices, Chapter 3, Chapter 4 and Chapter 5 are motivated by the similarity of Langevin dynamics [78] and leaky integrate-and-fire (LIF) neurons for performing stochastic inference [77]. Indeed, the fundamental dynamics of LIF neurons is governed by Langevin dynamics. Apart from its obvious relevance for the description of stochastic processes, the Langevin equation [78] can also be used for simulating quantum field theories with stochastic quantization [12–14]. The Euclidean path integral measure is obtained in this approach as the stationary distribution of a stochastic process. This paves the way to the heuristic approach of using complex Langevin dynamics as a potential method for accessing real time dynamics and sign problems. The latter problem is, e.g. prominent in QCD at finite chemical potential [79–81]. A further interesting application of the Langevin equation can be found in Ref. [82]. Here, Langevin dynamics is combined with a stochastic gradient descent algorithm to perform Bayesian learning which enables an uncertainty estimation of resulting parameters.

The BrainScaleS-2 chip features different use cases. In the following, we distinguish between a spiking mode and a non-spiking mode. In the spiking-based framework, spikes are the carriers for synaptic signals and information and are used by the neurons to communicate with each other in the neural network. They are generated by stimuli of other neurons or noise. If a neuron is in the firing mode, spikes are emitted. The emission of several spikes in a certain time span is a so-called spike train. The firing and the non-firing mode of a neuron allows the distinction of two possible states, which we will also refer to as an active or an inactive state. A system of N neurons can be interpreted in this case effectively as a discrete system with 2^N possible configurations. We want to point out once more that, despite the seemingly simple nature of spikes, there exist many

different ways to transfer information, as studied in the research area of neural coding [59]. Another important aspect is the non-linearity of the synaptic-potentials that are used to transmit spikes. The superposition of respective non-linear signals results in an even larger pool of possible ways for encoding information.

Because of the discrete nature of the resulting network of spiking neurons, a straightforward implementation of Langevin dynamics is not possible in the spiking mode. Chapter 3 focuses on an application of Langevin dynamics in discrete systems as a possible workaround. Therefore, it pursues the path to adapt the algorithm to be in concordance with the natural dynamics of the neuromorphic device. The resulting dynamics simplifies on a neural network system and proposes the Langevin machine as a novel network architecture which implements Boltzmann statistics. In Chapter 4, we compare the properties of the Langevin machine with an alternative approach for emulating Boltzmann statistics on the hardware. Besides a detailed introduction of the different dynamics, simplified models of the neuromorphic hardware are studied with a focus on controlling emerging sources of errors.

Apart from the spiking-based framework, the chip also enables a constant synaptic input from external sources. Reading out the membrane voltage of a neuron and re-injecting it as a synaptic input facilitates a communication without spikes. The system of neurons corresponds to a continuous system in the sense that each neuron is represented by a continuous value, given by the membrane potential of the neuron. This non-spiking framework is realized by the Hagen mode [83, 84], explained in more detail in Sec. 2.3. Possible applications for the implementation of Langevin dynamics in the non-spiking mode are elaborated in detail in Chapter 5.

Chapter 6 follows the second pathway to perform scientific computations by utilizing the hardware as a sampling device for Boltzmann distributed statistics. More specifically, we take advantage of a possible realization of restricted Boltzmann machines on the BrainScaleS-2 chip [28] by networks of LIF neurons. The chapter focuses on using this implementation to emulate measurement outcomes in quantum physics [85], which are inherently probabilistic in nature. We demonstrate an approximate representation of quantum states with spiking neural networks on BrainScaleS-2 that is sufficiently precise for encoding genuine quantum correlations. Since any quantum state can be mapped to a probability distribution [86, 87], it can in turn, also be represented by the generated probability distribution of the neuromorphic hardware device. The accelerated analog circuit dynamics and the inherently parallel nature of the neuromorphic substrate enable a rapid generation of samples which carries the potential of scaling benefits as compared to von-Neumann devices.

Lattice QCD and the numerical sign problem

In many physical theories, the measure $\exp(-S(\phi))$ in Eq. (1.1) turns out to be complex. Besides real-time dynamics [88–91], this is, for example, the case for the Hubbard model [92–97], for spin- or mass-imbalanced systems [98–102] and graphene [103–105] or for quantum chromo-dynamics at finite density [79, 106–116]. In these theories, the fermionic part of the system contributes a multiplicative fermion determinant to the path integral measure in Eq. (1.1). The determinant can be complex, resulting in an oscillating behavior of the integrand. This makes a direct application of stochastic techniques infeasible since the integral weight no longer represents a probability measure. Due to a possible cancellation of negative and positive contributions in the integral, almost every configuration is equally important. As a result, configurations with a small or negative Boltzmann weight are as important as samples with a large weight. To get numerical results with small errors

the entire configuration space needs to be covered by the simulation method, which is infeasible for high-dimensional systems. This limitation is referred to as the *sign problem* [15, 97, 115, 117, 118].

Complex Langevin dynamics is considered as a promising numerical method for computing observables for systems that are subject to a sign problem [15, 117]. However, two major problems of the method are numerical instabilities, such as, runaway trajectories, and a possible convergence to an unphysical solution [79, 80, 97, 119–121].

Applying complex Langevin dynamics, to models plagued by a sign problem is an active area of research, see Refs. [116, 122] for recent reviews. In Refs. [97, 115] an overview of other methods tackling the sign problem is given, such as reweighting or a deformation of the integration contour into the complex plane.

In Chapter 7, we introduce a framework that generalizes complex Langevin dynamics and allows deriving the algorithm from first principles. The framework comprises an interpretation of complex Langevin dynamics as a standard Markov chain Monte Carlo algorithm. This point of view opens up perspectives on making use of knowledge from several decades in research on Monte Carlo algorithms. Beyond providing a foundation for complex Langevin dynamics, the framework serves as a basis for deriving new algorithms for theories with and without a sign problem. We open up a perspective that has the potential to facilitate the development and improvement of novel algorithms and to better evaluate and understand existing approaches to tackling the sign problem.

Complex Langevin-type sampling by compensation is one method developed within this framework. The method allows deriving sampling algorithms with similar properties as complex Langevin dynamics and is presented in detail in Chapter 8.

In Chapter 9, we propose a further approach for tackling the sign problem. The new approach utilizes reweighting for the training of a machine learning algorithm to stabilize complex Langevin dynamics and to prevent a convergence to unphysical solutions of the sampling process.

Representation learning

Machine Learning (ML) has been applied to a variety of problems in the natural sciences. For example, it is regularly deployed in the interpretation of data from high-energy physics detectors [123, 124]. Algorithms based on learning have shown to be highly versatile, with their use extending far beyond the original design purpose. In particular, deep neural networks have demonstrated unprecedented levels of prediction and generalisation performance, for reviews see e.g. Refs. [49, 125]. These networks have proven to be capable of efficiently identifying high-level features in a broad range of data types—in many cases, such as speech or image recognition, with spectacular success [126–129].

Accordingly, there is growing interest to harness the capabilities of these algorithms in the lattice community, both for high energy physics and condensed matter systems, and in theoretical physics, in general. Applications include predictive objectives, such as detecting phase transitions, the identification of order parameters from lattice configurations as well as generative modelling to accelerate and support lattice simulations [2, 3, 38, 130–162]. We recommend Ref. [163] as an introduction to ML for physicists and Ref. [164] as a general review for ML applications in physics.

The high dimensionality of lattice configurations render machine learning, and, in particular, deep learning algorithms perfectly suited tools to gain more insights into the physics of a considered

system. Lattice configurations, drawn by Monte Carlo sampling algorithms, provide a representative ensemble of the probability distribution of the considered physical system. Thus, they give implicit access to study and explore correlations and hidden structures of lattice quantum field theories. Besides valuable information about hidden structures, the retrieved knowledge might also be used in a second step for the development of novel simulation algorithms or other techniques to tackle model-specific problems related to the sign problem and beyond, such as specific expansion schemes, for example.

A straightforward way to access structures in lattice configurations is given by the possibility to formulate objective functions for the training of deep learning algorithms. The architecture of neural networks allows for a successive layer-wise reduction of the dimensionality of a single configuration. This can be realized in both a supervised learning framework, based on a simple classification task, for example, and an unsupervised learning approach. In the latter case, no labeled data is needed. Instead, the algorithm is trained based on an objective for generating a compact latent representation.

Finding expressive lower-dimensional representations belongs to the research area of representation and feature learning, respectively, see Ref. [165] for a comprehensive review. Many representation learning algorithms were developed from an information-theoretic point of view or can be interpreted as such [166–180]. Thereby, a particular focus is on representation learning by maximizing the mutual information between the input data and the compressed, latent representation [181–183]. A notorious problem with this approach is that mutual information is generally intractable for continuous variables if the exact probability distribution is unknown, rendering an exact computation of the entropy and the partition sum infeasible [183–188]. The problem can be circumvented by defining lower bounds for the mutual information term, as proposed, for example, in Ref. [186], or by rewriting the maximization objective. These approaches lead to a successful integration of the concept of mutual information maximization in many different representation algorithms ranging from unsupervised learning of representations [189, 190] to applications in generative adversarial networks [191, 192], autoencoders [179], variational autoencoders [170, 193–195] and graph neural networks [196].

In Chapter 10, we follow a slightly different approach by implicitly optimizing the entropy of the embedded data. We propose a novel unsupervised representation learning method called neural adversarial embedding algorithms. The algorithm is motivated by the goal to embed graph-structured data into a lower-dimensional representation space. We refer to an optimization of the entropy instead of the mutual information since an estimation of the mutual information is non-trivial due to the different geometric structure of the input space and the embedded space. Graphs, used in many domains to represent data, have a non-Euclidean, discrete structure. Due to these properties, an application of standard deep learning algorithms is more challenging compared to other data representations as images or lattice configurations, as will be discussed in more detail [197–204].

Having an expressive lower-dimensional representation allows for the application of various learning tasks such as clustering, a notion for a distance and similarity, classification and regression. Furthermore, tools from explainable and interpretable machine learning can help to understand and comprehend the way data is processed by the algorithm and to interpret learned representations [205–207]. Explainable artificial intelligence addresses the fact that available algorithms, in particular those based on deep learning, often demonstrate remarkable performance in the search for previously unidentified features, but tend to lack transparency if applied naively. In Chapter 11, the analysis of lattice configurations for a possible detection of hidden structures is studied in the light of explainable machine learning. In the first part, we summarize the findings of Ref. [3]. In

this work representation learning is proposed in combination with interpretability methods as a framework for the identification of observables based on a supervised representation learning task. In the second part, we review a possible application of unsupervised learning methods with a focus on the proposed method in Chapter 10. In particular, we relate the generation of graph embeddings to other unsupervised learning techniques and point out possible advantages and drawbacks for an application on lattice configurations.

Spectral reconstruction and inverse problems

Ill-conditioned inverse problems lie at the heart of some of the most challenging tasks in modern theoretical physics. One pertinent example is the computation of real-time properties of strongly correlated quantum systems. Take e.g. the phenomenon of energy and charge transport, which so far has defied a quantitative understanding from first principles. This universal phenomenon is relevant to systems at vastly different energy scales, ranging from ultracold quantum gases created with optical traps to the quark-gluon plasma born out of relativistic heavy-ion collisions.

While static properties of strongly correlated quantum systems are by now well understood and routinely computed from first principles, a similar understanding of real-time properties is still subject to ongoing research. The thermodynamics of strongly coupled systems, such as the quark gluon plasma, has been explored using the combined strength of different non-perturbative approaches, such as functional renormalisation group methods and lattice field theory calculations. There are two limitations affecting most of these approaches: Firstly, in order to carry out quantitative computations, time has to be analytically continued into the complex plane, to so-called Euclidean time. Secondly, explicit computations are either fully numerical or at least involve intermediate numerical steps.

This leaves us with the need to eventually undo the analytic continuation of Euclidean correlation functions, which are known only approximately. The most relevant examples are two-point functions, the so-called Euclidean propagators. The spectral representation of quantum field theory relates the propagators, be they in Minkowski or Euclidean time, to a single function encoding their physics, the so-called spectral function. The number of different structures contributing to a spectral function are in general quite limited and consist of poles and cuts. If we can extract from the Euclidean two-point correlator its spectral function, we may immediately compute the corresponding real-time propagator.

If we know the Euclidean propagator analytically, this information allows us in principle to recover the corresponding Minkowski time information. In practice, however, the limitation of having to approximate correlator data (e.g. through simulations) turns the computation of spectral functions into an ill-conditioned problem. The most common approach to give meaning to such inverse problems is Bayesian inference. It incorporates additional prior domain knowledge we possess on the shape of the spectral function to regularise the inversion task. The positivity of hadronic spectral functions is one prominent example. The Bayesian approach has seen continuous improvement over the past two decades in the context of spectral function reconstructions. While originally it was restricted to maximum a posteriori estimates for the most probable spectral function given Euclidean data and prior information [208–210], in its most modern form it amounts to exploring the full posterior distribution [211]. An important aspect of the work is to develop appropriate mock data tests to benchmark the reconstruction performance before applying it to actual data. Generally, the success of a reconstruction method stands or falls with its performance on physical data. While this

seems evident, it was in fact a hard lesson learned in the history of Bayesian reconstruction methods, a lesson which we want to heed.

Inverse problems of this type have also drawn quite some attention in the machine learning community [212–215]. In Chapter 12, we explore artificial neural networks as a tool for the reconstruction of spectral functions from imaginary time Green's functions, a classic ill-conditioned inverse problem. Our approach is based on a supervised learning framework in which prior knowledge is encoded in the training data and the inverse transformation manifold is explicitly parameterized through a neural network. We systematically investigate this novel reconstruction approach, providing a detailed analysis of its performance on physically motivated mock data, and compare it to established methods of Bayesian inference. The reconstruction accuracy is found to be at least comparable, and potentially superior in particular at larger noise levels. We argue that the use of labeled training data in a supervised setting and the freedom in defining an optimization objective are inherent advantages of the present approach and may lead to significant improvements over state-of-the-art methods in the future. Potential directions for further research are discussed in detail.

1.2 Publications

While the thesis has been written solely by the author, many of the results were obtained in joint work with collaborators from different fields. Large parts of this work are already published or in preparation. Text and figures taken from the articles are not marked explicitly. In addition to the list below, we refer at the beginning of each chapter to used content of respective articles:

[1] **Discrete langevin machine: Bridging the gap between thermodynamic and neuromorphic systems**
L. Kades and J. M. Pawlowski
Published in Phys. Rev. E 101, 063304 (2020)
E-Print: arXiv:1901.05214 [cs.NE]
Comment: Chapter 3, Chapter 4 and App. A as well as parts of Chapter 1 and Chapter 2.

[2] **Spectral reconstruction with deep neural networks**
L. Kades, J. M. Pawlowski, A. Rothkopf, M. Scherzer, J. M. Urban, S. J. Wetzel, N. Wink, and F. P. G. Ziegler
Published in Phys. Rev. D 102, 096001 (2020)
E-Print: arXiv:1905.04305 [physics.comp-ph]
Comment: Chapter 12, App. F and parts of Chapter 1.

[3] **Towards novel insights in lattice field theory with explainable machine learning**
S. Blücher, L. Kades, J. M. Pawlowski, N. Strodthoff, and J. M. Urban
Published in Phys. Rev. D 101, 094507 (2020)
E-Print: arXiv:2003.01504 [hep-lat]
Comment: Parts of Chapter 1 and Chapter 11.

[4] **Spiking neuromorphic chip learns entangled quantum states**
S. Czischek, A. Baumbach, S. Billaudelle, B. Cramer, L. Kades, J. M. Pawlowski, M. K. Oberthaler, J. Schemmel, M. A. Petrovici, T. Gasenzer, and M. Gärttner
E-Print: arXiv:2008.01039 [cs.ET] (waiting to be published)
Comment: Chapter 6, App. D and parts of Chapter 1 and Chapter 2.

[5] **Towards sampling complex actions**
L. Kades, T. Gasenzer, M. Gärttner, J. M. Pawlowski
E-Print: arXiv:2106.09367 [hep-lat] (submitted to Phys. Rev. E)
Comment: Chapter 7, Chapter 8, App. B and App. C as well as parts of Chapter 1 and 2.

In addition, content of so far unpublished work is incorporated in the thesis:

[6] **Complex Langevin dynamics on a neuromorphic device**
A. Baumbach, L. Kades, J. M. Pawlowski, M. A. Petrovici and J. Schemmel
Comment: Chapter 5.

[7] **Unsupervised neural graph embedding**
L. Kades and K. Shmilovich
Comment: Chapter 10 and parts of Chapter 11.

[8] **Step-wise reweighting criterion for correctness of complex Langevin dynamics**
A. Hosak, L. Kades and J. M. Pawlowski
Comment: Parts of Chapter 9.

[9] **Self-consistent sampling of complex actions**
M. Bauer, L. Kades and J. M. Pawlowski
Comment: Chapter 9.

Moreover, two software frameworks that are not further discussed and not published yet were developed in the course of this thesis:

[10] **An easy to use Markov chain Monte Carlo sampling framework for lattice field theories**
L. Kades, J. M. Pawlowski
URL: https://github.com/statphysandml
Comment: A set of different repositories that simplifies setting up, running and evaluating Markov chain Monte Carlo simulations. The modular structure of the framework allows for the implementation of systems and models at different levels complexity. Many of the results in this thesis were obtained based on this framework.

[11] **Visualizing the functional renormalization group - Finding fixed points in high-dimensional spaces**
K. Höfling, L. Kades, M. Reichert, J. M. Pawlowski and F. Sadlo
URL: https://github.com/statphysandml/ODEVisualisation
Comment: Software framework for finding fixed points, for instance, in the context of asymptotically-safe quantum gravity or tensor models for quantum gravity by means of the functional renormalization group. The framework has been developed in combination with my Master in Applied Computer Science.

CHAPTER 2

Background

This chapter is in parts based on Refs. [1, 4, 5].

We start with a brief review of the sign problem as well as important concept of numerical methods in statistical physics and lattice field theory in Secs. 2.1 and 2.2. Sec. 2.3 provides an introduction to LIF sampling and details to the design and the different functionalities of the BrainScaleS-2 neuromorphic chip.

2.1 The sign problem, stochastic quantization and complex Langevin dynamics

The objective of this section is to introduce the basics of stochastic quantization, the sign problem and, specifically, complex Langevin dynamics.

2.1.1 The sign problem for a toy model

To illustrate how the sign problem appears in this context, we choose a toy model, which we will use in Chapter 7 for comparison of different algorithms. The zero-dimensional polynomial model [119, 216, 217] is defined by the action

$$S(\phi) = \frac{1}{2}\left(\sigma_{\mathrm{Re}} + i\sigma_{\mathrm{Im}}\right)\phi^2 + \frac{\lambda}{4}\phi^4\,, \tag{2.1}$$

a function depending on the real-valued scalar field $\phi \in \mathbb{R}$ and real-valued couplings $(\lambda, \sigma_{\mathrm{Re}}, \sigma_{\mathrm{Im}})$[1].

The objective is to compute observables over ϕ:

$$\langle \mathcal{O}(\phi) \rangle = \int_{-\infty}^{\infty} \mathrm{d}\phi\, \mathcal{O}(\phi)\rho(\phi)\,, \tag{2.2}$$

[1] This toy model is widely used in studying the sign problem. It is one of the simplest non-trivial quantum mechanical models [119] and describes, e.g., a single-mode relativistic interacting Bose gas at nonzero chemical potential $\mu \propto \sigma_{\mathrm{Im}}$ [216, 218].

with equilibrium Boltzmann measure

$$\rho(\phi) = \frac{1}{Z} \exp\left(-S(\phi)\right) , \tag{2.3}$$

which is normalized by the partition function

$$Z = \int_{-\infty}^{\infty} \mathrm{d}\phi \, \exp(-S(\phi)) \, , \tag{2.4}$$

and where the usual energy divided by temperature, βE, is upgraded, in quantum theory, to the Euclidean action S.

Standard Monte Carlo methods rely on sampling from a probability distribution. Since the action (2.1) is complex, these methods are, at first sight, inapplicable here.

2.1.2 Real Langevin dynamics

The Langevin equation, originally formulated to model Brownian motion [78], is central to the stochastic quantization approach to quantum field theory [13–16]. In the simplest case, it describes the evolution of a real, scalar field $\phi(x)$, governed by a real Euclidean action $S(\phi)$, in an additional, fictitious time dimension, the Langevin time τ. It reads

$$\frac{\partial}{\partial \tau} \phi(\tau) = -\frac{\delta S}{\delta \phi(\tau)} + \eta(\tau) \, , \tag{2.5}$$

where we suppress the dependence of ϕ and η on x for brevity.

Similar to thermal fluctuations in a thermodynamic system with energy E, the noise term η emulates quantum fluctuations in the case of a Euclidean quantum field theory. The distribution of the noise term η is usually taken to be centred at zero,

$$\langle \eta(\tau) \, \eta(\tau') \rangle_\eta = 2\delta(\tau' - \tau) \, , \quad \langle \eta(\tau) \rangle_\eta = 0 \, , \tag{2.6}$$

where $\langle \cdot \rangle_\eta$ denotes the expectation value with respect to the noise distribution. A common choice for this distribution is Gaussian white noise. Under these conditions, the τ-dependent distribution of ϕ is subject to the Fokker-Planck equation [13, 14, 97],

$$\frac{\partial \rho(\phi, \tau)}{\partial \tau} = \int \mathrm{d}^d x \frac{\delta}{\delta \phi(\tau)} \left(\frac{\delta S}{\delta \phi(\tau)} + \frac{\delta}{\delta \phi(\tau)} \right) \rho(\phi, \tau) \, . \tag{2.7}$$

Its stationary solution is the Boltzmann distribution (2.3),

$$\lim_{\tau \to \infty} \rho(\phi, \tau) = \rho(\phi) \, . \tag{2.8}$$

For the case of a real-valued action $S(\phi)$ considered here, it can be shown that the Langevin evolution converges, in the limit $\tau \to \infty$, to the desired equilibrium distribution as a stationary solution and that the convergence is exponentially fast [13]. After an equilibration period of time $\bar{\tau}$, observables can be evaluated by

$$\langle \mathcal{O}(\phi) \rangle_\rho \simeq \frac{1}{T} \int_{\bar{\tau}}^{\bar{\tau}+T} \mathrm{d}\tau \, \mathcal{O}(\phi(\tau)) \, , \tag{2.9}$$

where T is a suitable time to correctly estimate equilibrium expectation values through temporal averaging. Hence, by discretizing both x and τ, the Langevin equation provides a means to sample lattice quantum field theories, as long as the accumulation of numerical errors caused by the discretization is controllable.

2.1.3 Complex Langevin dynamics

A generalization of stochastic quantization to complex distributions $\rho(\phi)$ has been proposed as a means to numerically access observables of systems with a sign problem [13, 15, 117]. For a complex action $S(\phi)$, the Langevin equation (2.5) in general describes an evolution leading to complex values for $\phi = \phi_x + i\phi_y$. The real and imaginary components are then commonly evolved according to the equations

$$\begin{aligned} \frac{\partial}{\partial\tau}\phi_x(\tau) &= -\mathrm{Re}\left[\frac{\delta S}{\delta\phi(\tau)}\bigg|_{\phi_x+i\phi_y}\right] + \eta_x(\tau)\,, \\ \frac{\partial}{\partial\tau}\phi_y(\tau) &= -\mathrm{Im}\left[\frac{\delta S}{\delta\phi(\tau)}\bigg|_{\phi_x+i\phi_y}\right]\,, \end{aligned} \tag{2.10}$$

where only the equation for ϕ_x is of the Langevin form with, commonly, white noise η_x, while the equation for ϕ_y describes a pure drift. A noise term in the imaginary part can introduce stability problems which is why the evolution is usually driven by purely real noise [216, 219]. We will show in Chapter 7 that the missing imaginary noise term is also justified on formal grounds.

The stochastic process converges to solutions governed by a real-valued steady-state distribution $P(\phi_x, \phi_y) = \lim_{\tau\to\infty} P(\phi_x, \phi_y, \tau)$ in the ϕ_x-ϕ_y-plane. Observables

$$\langle\mathcal{O}(\phi_x + i\phi_y)\rangle_p = \int \mathrm{d}\phi_x \int \mathrm{d}\phi_y\, \mathcal{O}(\phi_x + i\phi_y) P(\phi_x, \phi_y, \tau) \tag{2.11}$$

can be numerically computed by sampling from the resulting distribution. The expectation values coincide under certain constraints with the expectation values with respect to the original complex distribution $\rho(\phi, \tau)$,

$$\langle\mathcal{O}(\phi_x + i\phi_y)\rangle_P = \langle\mathcal{O}(\phi)\rangle_\rho\,. \tag{2.12}$$

In the standard approach to analysing the convergence of complex Langevin, one compares two independent time-dependent stochastic processes (2.10), namely, the evolutions of the distribution $P(\phi_x, \phi_y, \tau)$, and of the underlying complex distribution $\rho(\phi, \tau)$ by means of their respective Fokker-Planck equations [97, 122, 219]. Details about the existence and the properties of a stationary distribution $P(\phi_x, \phi_y)$, which satisfies Eq. (2.12), can be found, for example, in Ref. [220].

An issue with the complex Langevin ansatz is the absence of a guaranteed convergence to the correct equilibrium distribution, which, in most cases, can only be verified a posteriori. As pointed out, a correct convergence is ensured only under certain conditions that have been elaborated in the past, see, for example Refs. [16, 122, 216, 217, 219–225].

Besides the requirement of ergodicity, model actions and distributions should ideally be holomorphic. Studying models with meromorphic poles is, in principle, also possible, but more care has to be taken to ensure convergence [226]. Lastly, the numerically sampled distributions of the observables need to decay fast enough in the imaginary direction [217, 223].

There exist different ways for checking if these criteria are fulfilled. One important way involves computing boundary terms by considering the derivative of a quantity $F_{\mathcal{O}}(t,\tau)$ with respect to the Langevin time τ [219, 221, 224, 227]. The function $F_{\mathcal{O}}(t,\tau)$ interpolates between the two observables in Eq. (2.12). Other approaches are based, for example, on an analysis of the decay of the sampled probability distribution [217, 223, 228].

2.2 Markov chain Monte Carlo sampling and Langevin dynamics

2.2.1 Master equation, equilibrium and detailed balance

The time evolution of a distribution $\rho(x,\tau)$ of a stochastic state variable x, subject to transition probabilities $W(x \to x')$, can in general be described by a master equation [229]:

$$\frac{\mathrm{d}\rho(x,\tau)}{\mathrm{d}\tau} = \sum_{x'} \left[\rho(x',\tau)\, W(x' \to x) - \rho(x,\tau)\, W(x \to x')\right] . \tag{2.13}$$

The right-hand side of the equation contains gain and loss terms for the state x to go over to x' and vice versa.

The standard work flow for setting up Markov chain Monte Carlo (MCMC) algorithms is to choose transition probabilities in such a way that the evolution converges, in the infinite-time limit, to an equilibrium distribution. The equilibrium distribution is expected to coincide with the probability distribution of interest. In our case, we aim at sampling from an explicitly given distribution $\rho(x)$.

The system is defined to be in equilibrium if its state distribution does not change anymore over time. This is the case when the sum on the right-hand side of the master equation (2.13) evaluates to zero. This translates into the equilibrium condition

$$\rho(x',\tau) \stackrel{!}{=} \sum_{x} \rho(x,\tau)\, W(x \to x') , \tag{2.14}$$

as can be derived by using the normalization of $W(x \to x')$ in the second term on the right-hand side of Eq. (2.13) and a respective renaming of x and x'. However, it needs to be taken into account that the above condition does not guarantee a correct sampling from the desired distribution due to possible limit cycles occurring in the Markov chain [229].

A more restrictive equilibrium condition is that the transition probabilities satisfy the detailed-balance equation:

$$\rho(x) W(x \to x') = \rho(x') W(x' \to x) . \tag{2.15}$$

Detailed balance implies that the sum on the right-hand side of the master equation (2.13) vanishes separately for every summand and that the process thus samples from the equilibrium distribution.

The transition probabilities introduced above are used in a Markov chain to draw samples from the equilibrium distribution. Observables, as defined in Eq. (1.1), are then numerically accessible by computing expectation values according to:

$$\langle \mathcal{O}(x) \rangle = \frac{1}{N} \sum_{i=1}^{N} O(x_i) , \tag{2.16}$$

where the sum runs over the drawn samples.

Besides a time-independent state distribution, it is important that further necessary conditions, like ergodicity, are fulfilled, for more details see, for example, Ref. [229].

2.2.2 Langevin dynamics as a Markov chain Monte Carlo algorithm

This section shows that Langevin dynamics can be assigned to the class of Markov chain Monte Carlo algorithm. The considerations are in concordance with the former comparisons of Langevin dynamics and Monte Carlo algorithms [230–232]. In particular, we demonstrate that the transition probabilities of Langevin dynamics satisfy the detailed-balance equation (2.15).

The transition probability $W(\phi \to \phi')$ for Langevin dynamics is derived in App. A.1. It is computed based on a discrete form of the Langevin equation where an infinitesimal update step of a field $\phi := \phi(\tau)$ to a field $\phi' := \phi(\tau + \epsilon)$ is considered. The resulting transition probability can be rewritten as

$$W(\phi \to \phi') \propto \frac{1}{\sqrt{2\epsilon}} \varphi\left(\frac{\phi' - \phi}{\sqrt{2\epsilon}}\right) \exp\left[-\frac{S(\phi') - S(\phi)}{2}\right] . \tag{2.17}$$

The first factor can be interpreted as the selection probability and the second term as the acceptance probability for the proposed field ϕ'.

A correct sampling of the Boltzmann distribution $\rho(\phi) \propto \exp(-S(\phi))$ can be verified by inserting the transition probability into the detailed-balance equation (2.15). The selection probability is symmetric with respect to an exchange of ϕ' and ϕ. It drops out, resulting in a satisfaction of the detailed-balance equation.

The continuous formulation of Langevin dynamics can be restored in the limit of $\epsilon \to 0$. The limit results in a correct normalization of the transition probability, as can be seen by identifying the selection probability as a representation of the delta distribution,

$$\int_{-\infty}^{\infty} d\phi' W(\phi \to \phi') = \int_{-\infty}^{\infty} d\phi' \delta(\phi' - \phi) \exp\left[-\frac{S(\phi') - S(\phi)}{2}\right] = 1 . \tag{2.18}$$

Hence, the step size between the current field and the proposal field becomes infinitesimally small.

We conclude that the Langevin equation (2.5) can be interpreted as a standard Monte Carlo algorithm with a Gaussian distribution as selection probability. The proposal state is chosen implicitly by an absorption of the acceptance probability into the selection probability and by a corresponding sampling with Gaussian noise. Since the nearest neighbor sites can be assumed to be nearly constant in one Monte Carlo step, it is possible to switch from a random sequential update formalism to a parallel update of the entire lattice. The Langevin time is introduced as a temporal measure for a lattice update. In principle, the delta distribution can be exchanged by any other positive representation.

2.3 LIF sampling and neuromorphic computing

This section reviews the dynamics of leaky integrate-and-fire (LIF) neurons and provides an introduction to the BrainScaleS-2 chip [28]. It is shown how the neuromorphic hardware system can be

used to perform stochastic inference with spiking neurons in the high-conductance state [28, 76, 77]. In this state, the LIF neurons receive a strong synaptic stimulus, in this case, Poisson stimulus, leading to accelerated, stochastic membrane dynamics [233]. The section ends with an introduction of the so-called Hagen mode, a non-spiking working mode of the neuromorphic hardware facilitating vector matrix-multiplications.

2.3.1 Spiking neural networks by LIF sampling

Simplified LIF dynamics

The spikey neuromorphic system of the BrainScaleS project emulates spiking neural networks with physical models of neurons and synapses implemented in mixed-signal microelectronics [43, 77]. With the help of Poisson-driven leaky integrate-and-fire (LIF) neurons, it is possible to obtain stochastic inference with deterministic spiking neurons. The dynamics of the free membrane potential $u_{\text{eff}}(t)$ of a neuron can be approximated in the resulting high-conductance state by an Ornstein-Uhlenbeck process,

$$\frac{du_{\text{eff}}(t)}{dt} = \theta\left[\mu - u_{\text{eff}}(t)\right] + \sigma\tilde{\eta}(t)\,. \tag{2.19}$$

In Eq.(2.19), θ determines the strength of the attractive force towards the mean value $\mu = \mu^{\text{leak}} + \mu^{\text{average noise}}$. The mean value consists of some leak potential and an additional averaged noise contribution. The stochasticity required for sampling is induced by adding a random component to the generation of spikes; for LIF networks, this can be ensured by sufficiently noisy membrane potentials [75, 77]. For the Ornstein-Uhlenbeck process (2.19), the Gaussian noise term $\tilde{\eta}(t)$, characterized by a zero mean and a unit variance,

$$\langle\tilde{\eta}(t)\,\tilde{\eta}(t')\rangle_{\tilde{\eta}} = \delta(t'-t)\,, \quad \langle\tilde{\eta}(t)\rangle_{\tilde{\eta}} = 0\,, \tag{2.20}$$

is emulated by the synaptic input of additive Possion process. Hence, the parameter σ depends on the Poisson background [77].

Inspired by a biological neuron [234], the neuron emits a neural spike (action potential) when the membrane potential exceeds a certain threshold ϑ. It is active and is reset to ϱ for a refractory time τ_{ref} afterwards, otherwise the neuron is considered as inactive. This is also sketched in Fig. 2.1. One has to distinguish between the effective membrane potential $u_{\text{eff}}(t)$ (red curve), referring to the instantaneous target voltage, given by the current state of the synaptic input, and the real membrane potential $u(t)$ (blue curve). In the high-conductance state, the convergence of $u(t)$ from ϱ to $u_{\text{eff}}(t)$ takes place in a negligible time after the finite refractory time has elapsed [77].

Communication in a network of LIF neurons is realized via neural spikes. In this case, the mean value μ depends in addition on an additive interaction term $\mu^{\text{interaction}}$. Whenever the membrane potential of a neuron, which is the result of a leaky integration of the synaptic input, exceeds a threshold, it sends a spike to the neurons connected to it. The way the other neurons receive the signal of the spike is determined by the so called post-synaptic potential (PSP). The shape of this potential is in general non-linear. In the theoretical considerations in Chapter 4, we simplify the interaction by a rectangular PSP shape.

In a spike-based sampling framework, the refractory period τ_{ref} following a spike can be interpreted to encode a state $z = 1$ (the neuron is active), while $z = 0$ at all other times (the neuron is inactive).

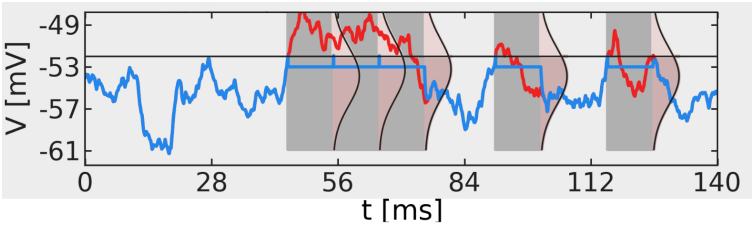

Figure 2.1: Example evolution of the free membrane potential (red) and the actual membrane potential (blue). After the membrane potential crosses the threshold ϑ, a spike is emitted and the potential is set to a reset potential ϱ. If the effective membrane potential is still above ϑ after the refractory period τ_{ref}, the neuron spikes again. At the end of a "burst" of n spikes the free and the actual membrane potential converge in negligible time (scheme taken from Ref. [234]).

Accordingly, neural spikes are used to mark transitions between discrete states (active and inactive neurons) and thereby effectively carry out a sampling process in discrete states.

Activation function

An implicit expression for the neural activation function of LIF neurons in the high-conductance state is derived for the neuromorphic hardware system in Ref. [77]:

$$P(z=1) = \frac{\sum_n P_n n \tau_{\text{ref}}}{\sum_n P_n \left(n\tau_{\text{ref}} + \sum_{k=1}^{n-1} \overline{\tau_k^b} + T_n \right)} \,. \tag{2.21}$$

Here, the burst length n refers to the number of consecutive spikes. The respective occurrence probability of a burst with length n is given by P_n. Similarly, the mean time interval, starting from the endpoint of the last refractory period after a burst with length n and ending at the next spike, is denoted as T_n. Lastly, $\overline{\tau_k^b}$ corresponds to the average time after the k-th refractory period within a burst that it takes for the effective membrane potential to drift from the resting potential ρ to the threshold potential Θ. More details on the computation of P_n, T_n and $\overline{\tau_k^b}$ by means of the transfer function $p(u_{i+1}|u_i)$ of the Ornstein-Uhlenbeck process and by taking into account the spiking mechanism of the neurons can be found in Refs. [77, 234].

Interactions

Interactions between neurons in a spiking neural network, see also Sec. 2.3, are non-trivial. The so-called postsynaptic potential (PSP) refers to the input potential of a connected neuron in case of firing. As shown in Ref. [77], the interaction term, utilized, for example, in Eq. (4.1), can be implemented by

$$\mu_i(t)^{\text{interaction}} = \sum_{\text{syn } j} \sum_{\text{spk } s} A_{ij} \kappa(t, t_{s,j}) \,, \tag{2.22}$$

with $A_{ij} = \frac{w_{ij}\left(E_{ij}^{\text{rev}} - \langle u_{\text{eff}} \rangle\right)}{\langle g_{\text{tot}} \rangle}$ and where $t_{s,j}$ is the time of the last spike. The interaction kernel $\kappa(t, t_{s,j})$ describes the PSP shape and depends in general on the time constants: τ_{ref}, τ_{syn} and τ_{eff}.

The actual PSP shape in a network of LIF neurons is of the following form [77]:

$$\kappa(t, t_{s,j}) = \frac{\exp\left[-\frac{t-t_{s,i}}{\tau_{\text{eff}}}\right] - \exp\left[-\frac{t-t_{s,j}}{\tau_{\text{syn}}}\right]}{\tau_{\text{eff}} - \tau_{\text{syn}}} \,. \tag{2.23}$$

A possible approach to translate weights W_{ij} onto the neuromorphic system is to assume that the area under a PSP shape is equal to $W_{ij}\tau_{\text{ref}}\alpha$, where α represents a scaling factor,

$$W_{ij}\tau_{\text{ref}}\alpha = \int_0^{\tau_{\text{ref}}} A_{ij}\kappa(t, t_{s,j})\,\mathrm{d}t \,. \tag{2.24}$$

In a simplified description, the PSP shape has a rectangular form,

$$\kappa(t, t_{s,j})^{\text{rect}} = \Theta\left[t - t_{s,j}\right] - \Theta\left[t - t_{s,j} - \tau_{\text{ref}}\right] \,. \tag{2.25}$$

For $\tau_{\text{ref}} \to 0$, the neuron j is in the active state $z_j = 1$ as long as $u_j(t) > \vartheta_j$ and the interaction term turns to

$$\mu_i(t)^{\text{interaction}} = \sum_{\text{syn}j} A_{ij}\Theta\left[u_j(t) - \vartheta_j\right] \,. \tag{2.26}$$

The findings in Chapter 4 are with respect to rectangular PSP shapes, as simplified abstractions of interactions on neuromorphic systems.

2.3.2 Representing and training Boltzmann machines and other distributions

In classical machine learning, generative models based on artificial neural networks are used to encode and sample from probability distributions [235]. Similarly, spiking neural networks can be viewed as approximating Markov-chain Monte-Carlo sampling, albeit with dynamics that differ fundamentally from standard statistical methods [234].

Following the distinction of a neuron to be active or inactive, we restrict ourselves in the following to a sampling of discrete states. A network of n spiking neurons spans an n dimensional state space resulting in 2^n possible configurations. In Chapter 5, we go beyond this and discuss a sampling scheme in an abstract model description of a coupled set of LIF neurons. In this scheme, the membrane potentials are utilized as continuous, microscopic states of the system.

Samples from the system of discrete states can be drawn, for example, by observing the activity of the neurons in regular time intervals. Collecting these samples results in a probability distribution which we denote as $p(\vec{z}; \theta)$, where θ refers to a given configuration of hyperparameters of the LIF dynamics and the hardware, respectively. The distribution is inherently related to the neuromorphic device and the respective underlying dynamics. There exists no closed expression for this distribution due to the complexity of the dynamics and hardware-related noise in the system parameters. Nevertheless, the neuromorphic hardware system is not a black box. Based on the fundamental dynamics of a single neuron, one can derive model descriptions for the system. An example is given by the derived activation function of a single neuron in Eq. (2.21) with LIF dynamics in the high-conductance state as a starting point [77, 236].

Model descriptions allow an in-depth understanding of the neuromorphic device. Corresponding models can be helpful in scientific computing tasks in statistics and machine learning, such as Bayesian inference and sampling or the implementation of generative models. In the following, we

focus on emulation of Boltzmann machines by means of hierarchical spiking neural networks [77, 237]. From a practical point of view, this is motivated by the binary nature of the given distribution and the similarity of the activation function (2.21) of LIF neurons in the high-conductance state and the one of a Boltzmann machine. Furthermore, Boltzmann distributed systems have a wide field of application in machine learning and physics [34, 38, 238–242].

A Boltzmann machine [243] is a neural network consisting of neurons $\vec{z} = (z_1, \ldots, z_N)$ following the Boltzmann distribution

$$p(\vec{z}) = \frac{1}{Z} \exp(-E(\vec{z})) \tag{2.27}$$

with the partition function

$$Z = \sum_{z_1,\ldots,z_n} \exp(-E(\vec{z}))\,. \tag{2.28}$$

The sum runs over all possible configurations of the neuron states $z_i \in \{0, 1\}$. The energy function $E(\vec{z})$ is defined in the common way as

$$E(\vec{z}) = -\sum_{i<j} W_{ij} z_i z_j - \sum_i b_i z_i\,, \tag{2.29}$$

where W_{ij} are symmetric, synaptic weights between the neurons i and j and b_i is an additional bias.

Boltzmann distributed statistics can be sampled by a sequential update scheme of the neurons with the logistic distribution as activation function,

$$P(z_i = 1) = \frac{1}{1 + \exp\left[-\sum_j W_{ij} z_j - b_i\right]}\,, \tag{2.30}$$

where the sum runs over all synaptic connections of the neuron i to other neurons j. The activation function can be derived, for example, by rewriting the Boltzmann distribution as a conditional distribution defined by z_i and $\vec{z}_{\setminus z_i}$, where the latter corresponds to the set of neurons without the neuron z_i.

$$P(z_i = 1) := p(z_i = 1|\vec{z}_{\setminus z_i}) = \frac{p(z_i = 1, \vec{z}_{\setminus z_i})}{p(z_i = 0, \vec{z}_{\setminus z_i}) + p(z_i = 1, \vec{z}_{\setminus z_i})}\,. \tag{2.31}$$

In a detailed computation one finds that all terms in the energy function that are independent of z_i drop out and arrives at Eq. (2.30), cf. Refs. [244, 245].

In the so-called spike-based sampling scheme, the hardware generates samples distributed closely to a Boltzmann distribution, see Refs. [77, 234, 237] for a detailed quantitative analysis. We analyse emerging sources for errors in more detail based on several simplified models of the hardware in Sec. 4.2.

In a restricted Boltzmann machine (RBM), an additional distinction between visible and hidden layers allows the representation and training of a wider class of distributions obtained by employing the marginal distribution over the set of visible neurons. The neurons are ordered on a bipartite graph and the represented probability distribution is defined as a function of the neurons in the visible layer. The latent code in the hidden layers and the respective increased number of weights is the reason for a higher representational power towards Boltzmann machine.

The weights and the biases can be adapted by respective parameters of the neuromorphic system, cf. Sec. 2.3.3. These parameters affect the emulated probability distribution and can be trained

in an unsupervised or supervised training framework. The presence of a decent, explicit model for the sampled distribution brings several benefits. In particular, it allows a well-defined sampling of the respective distribution as well as a direct utilization in standard training algorithms and loss functions of machine learning. In Chapter 6, we make use of the possible model description as a restricted Boltzmann machine for an optimization of the so-called Kullback-Leibler divergence, an asymmetric measure between probability distributions. It is used to the represent entangled quantum states on the neuromorphic hardware device.

In the following, we want to go into more detail with respect to advantages and drawbacks of model description. Firstly, we want to emphasize that the analytic expression of the Boltzmann distribution (2.27) is only a model description and, therefore, an approximation of the system for emulating Boltzmann machines. This can be resolved by estimating errors in the represented distribution. Secondly, it also means that a change in the weights or biases in the model distribution does not necessarily affect the sampled distribution on the neuromorphic hardware device in the same manner. This also implies that there needs to be a sufficient overlap between the model description and the dynamics of the hardware for a successful training. It is furthermore interesting to note that this overlap implicitly determines an upper (in the case of a maximization) or a lower bound (in the case of a minimization) for the optimization of the loss function.

A possible way to overcome this restriction is a training framework which does not need an explicit expression for the represented probability distribution. Due to a lack of this expression, it is non-trivial to infer the impact of single system parameters on the observed distribution in an optimization step. Approaches into this direction are presented in Refs. [246, 247].

Another possible approach is to choose more complicated model distributions. For example, a promising ansatz for reconstructing more complicated graphical models of arbitrary binary distributions in terms of an effective action is proposed in Refs. [248, 249]. However, also in this case, it might be difficult to define reliable correlations between system-related and model-dependent parameters. In an optimal setting, these dependencies are optimized, for example, by a machine learning algorithm.

Lastly, we want to point out that there is also great progress in the development of novel training algorithms designed specifically for spiking neural networks [19, 61, 62, 246, 247, 250]. Introduced alternatives to gradient-based optimization algorithms do not rely on an explicit model distribution. For example, a training framework based on surrogate stochastic gradients has been proposed recently in Ref. [246]. Superior to the discussed representation of Boltzmann machines, loss functions can be formulated based on both the membrane potential and the spiking activity. In addition, the presented in-the-loop training framework automatically takes into account analog hardware imperfections through self-calibration [246]. Respective training algorithms on the level of spikes are closer to the dynamics of the neuromorphic hardware device resulting in a potentially higher performance regarding computational power and energy-efficiency.

2.3.3 Neuromorphic computing: BrainScaleS-2

The BrainScaleS-2 system is a mixed-signal neuromorphic platform. Its analog core is composed of neuron and synapse circuits with inherent time constants of the order of microseconds. An application-specific integrated circuit (ASIC) for the BrainScaleS-2 system features 512 neuron circuits, which emulate the adaptive exponential integrate-and-fire model. These individual compartments can be wired to resemble more complex structured neurons. An on-chip analog parameter memory as well as integrated static random-access memory (SRAM) cells allow us to individually configure and

optimize the dynamics of each circuit. Each neuron integrates input from 256 dedicated synapses, which carry a 6-bit weight and can be either excitatory or inhibitory [28].

The analog core of the BrainScaleS-2 chip is accompanied by supporting logic, including circuitry for communication and configuration. Further functionality is provided by high-bandwidth spike sources, which can emit either regular or Poissonian spike trains of configurable frequency. These on-chip sources can be used to generate the Gaussian noise contribution necessary for the implementation of LIF neurons in the high-conductance state according to Eq. (2.19). A routing module allows mixing these spikes with external stimuli and recurrent events. It allows, in combination with in-synapse event filtering, the implementation of arbitrary network topologies.

A custom embedded processor allows the modification of the entire configuration space during the runtime of an experiment [251, 252]. Tightly coupled to the synaptic arrays, they allow the efficient and flexible implementation of learning rules based on observables such as neuronal potentials, firing rates, and synaptic correlations.

The respective weight (6-bit) matrix and an additional (10-bit) bias vector are configurable and determine together with the stochastic input the probability distribution the network is sampling from. Based on the model description in the previous section, these can be related to the weights and the biases of the Boltzmann machine, facilitating the definition of a training algorithm. Inherent properties of LIF sampling and of the hardware system can be the origin for possible deviations between the actually emulated probability distribution and the Boltzmann distribution as an approximated model distribution. Respective properties are, for example, non-linear post-synaptic potentials for the interaction with spikes and a finite calibration accuracy of configurable hardware-related parameters.

As an experimental result, the BrainScaleS-2 chip returns a list of all spike times and associated neuron IDs. This information is sufficient to reconstruct the network state at any point in time. The distribution, sampled by the network, is estimated by observing its state at regular intervals. To ensure an optimal estimate, the observation frequency needs to be at least $(\tau_{\text{ref}}/2)^{-1}$ (cf. App. D.4). Using $(\tau_{\text{ref}}/5)^{-1}$ guarantees a large safety margin. The resulting binary configurations are collected in a histogram. This can be used, for example, for the numerical evaluation of the Kullback-Leibler divergence or other observables and statistics, cf. Chapters 4 and 6. Further details for implementing a sampling spiking network on BrainScaleS-2, utilized in Chapter 6, can be found in App. D.4.

2.3.4 Hagen mode

The Ornstein-Uhlenbeck process defined in Eq. (2.19) for the description of LIF neurons in the high-conductance state is not the only dynamics that the BrainScaleS-2 chip can implement. In dependence of further system parameters, in particular, time constants, the dynamics can also fundamentally differ from the described one. In particular, the dynamics is usually without any external noise sources. This translates the system into a deterministic one, while inherent noise of the hardware stays as the only source for non-determinism.

In this section, we want to give a brief introduction of the so-called Hagen mode, which allows a different kind of usage of the hardware [83, 84]. Simply put, the mode allows the computation of multiply-accumlate (MAC) operations. With respect to the topology of the hardware, this can be understood as the computation of the following matrix-vector multiplication with additional additive contributions

$$y_i = \sum W_{ij} x_j \tag{2.32}$$

where the weight matrix is related again to the configurable weight matrix on the BrainScaleS-2 chip [28]. The multiplication is emulated by an utilization of the synaptic weight matrix as an integrator of a current I over a time interval Δt,

$$Q = I\Delta t\,, \tag{2.33}$$

where Δt encodes the input value x and I is proportional to the synaptic weight. The charge Q result in a change of the output membrane voltage which is identified with the outcome y. In addition, external inputs, and also external noise can be added to the membrane potential.

As discussed in more detail in Refs. [83, 84], one has to take into account a finite read-and write precision for the weights and biases a well as for the input and output vectors. Furthermore, the linearity of the operation cannot always be guaranteed and highly depends on the value regimes of the different factors. Fixed pattern noise sources and trial-to-trial variations entail further perturbations in the computation of the matrix-vector multiplication.

Despite these obstacles, scientific computations can benefit from an utilization of the Hagen mode due to a promising high energy efficiency on the BrainScaleS-2 chip for this kind of matrix-vector multiplications. The mode is designed specifically for hybrid computations of fast and energy-efficient evaluations of matrix-vector multiplications on the neuromorphic chip and additional computations on a digital computer. It can be supportive in applications with a heavy workload on inference tasks based on feed-forward networks by an energy-efficient and fast computation of the forward pass within the different layers. Such a high-performance is required, for example, in real-time industrial settings as autonomous driving.

A possible integration of the Hagen mode into the parallel sampling process of (complex) Langevin dynamics is discussed in Chapter 5. The integration is motivated by the high energy efficiency of the neuromorphic chip and a potential scalability to computations of larger physical systems, closer to the continuum limit and, therefore the actual physics.

CHAPTER 3

Langevin dynamics for discrete systems

This chapter is in parts based on Ref. [1].

We introduce in this chapter a formulation of Langevin dynamics that can be applied on discrete systems. This new kind of dynamics is motivated by the similarity of the underlying dynamics of the BrainScaleS-2 chip and Langevin dynamics and the accompanied discrete nature of a spike-based sampling framework on a neuromorphic hardware system.

This similarity suggests the formulation of a discrete analog to continuous Langevin dynamics. In this chapter, we show that a formulation of Langevin dynamics for discrete systems leads to a class of a generic stochastic process, namely, the *Langevin equation for discrete systems*,

$$\phi' = \phi + (\nu - \phi)\,\Theta\left[-1 - \frac{\epsilon}{2\lambda_\epsilon}\Delta S(\nu, \phi) + \sqrt{\epsilon}\tilde{\eta}\right], \tag{3.1}$$

where ϕ is the current state and ϕ' the updated state. The process is driven by a Gaussian noise term $\tilde{\eta}$. The parameter ϵ has an impact on the acceptance probability of the proposed state ν and should be chosen small, with $\epsilon > 0$. The term $\Delta S(\nu, \phi) := S(\nu) - S(\phi)$ measures the change of the action S of the system under a transition from state ϕ to the proposed state ν. $\Theta(x)$ represents the Heaviside function. The dynamics leads in the limit of $\epsilon \to 0$ to Boltzmann distributed states. The limit can be realized by an extrapolation of observables for several simulations with small values of ϵ. The additional scaling factor λ_ϵ, defined in Eq. (3.5), supports a faster convergence of the process to a Boltzmann distribution for finite values of ϵ.

A more detailed derivation of Eq. (3.1) including a discussion of its properties is given in Sec. 3.1. Numerical results of the new dynamics are presented based on the q-state clock model in Sec. 3.2. Furthermore, we analyze the dynamics with respect to its applicability and compatibility with spiking neural networks in Chapter 4.

The Langevin equation for discrete systems can also be applied on systems with continuous states. Interestingly, the Langevin equation for discrete systems converges in a certain limit to continuous Langevin dynamics. This limit is given by an infinitesimally small distance between the proposed state ν and the current state ϕ. We derive Langevin dynamics in this limit in Sec. 3.3, and complex Langevin dynamics in Sec. 3.4. The respective equivalence of the two dynamics confirms that Eq. (3.1) indeed represents an analog of Langevin dynamics for a possible application on discrete systems.

3.1 General definition

The formulation of the *Langevin equation for discrete systems* is inspired by the interpretation of Langevin dynamics as a Markov chain Monte Carlo algorithm and a respective satisfaction of a detailed balance equation, cf. Sec. 2.2.2. Following this interpretation, the here presented general formulation of a Langevin equation for discrete systems is also a Markov chain Monte Carlo algorithm. The process is driven by a Gaussian noise contribution, similar to Langevin dynamics. The transition probability to a proposed state can be regulated by replacing the Gaussian noise term by truncating Gaussian noise, cf. Eq. (3.10). It is shown that the accuracy of the process strongly depends on the intrinsic parameter ϵ and the scale of the energy contribution.

Certain necessary properties of a possible Langevin equation for discrete systems can be stated beforehand based on properties of the transition probability of Langevin dynamics studied in Sec. 2.2.2. An infinitesimal change of the microscopic state/field is not possible in a discrete system. Therefore, one has to switch from a parallel to a random sequential update mechanism. The proposal field has to be chosen from a discrete distribution. One may select the proposal field according to some distribution around the current field. However, since a parallelization is not possible, a uniform selection probability can be used.

Assuming the same acceptance probability as in the continuous case, a starting point is the following proportionality of the transition probability $W(\phi \to \phi')$:

$$W(\phi \to \phi') \propto \exp\left[-\frac{S(\phi') - S(\phi)}{2}\right] . \tag{3.2}$$

The transition probability from a state ϕ to a proposed state ϕ' can be rewritten with the help of the following relation between the cumulative Gaussian distribution

$$\Phi(x) = \int_{-\infty}^{x} dt \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2}\right) = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right] \tag{3.3}$$

and the exponential function:

$$\lim_{\epsilon \to 0} \frac{\Phi\left(-\frac{1}{\sqrt{\epsilon}} + \sqrt{\epsilon}\frac{x}{\lambda_\epsilon}\right)}{\Phi(-\frac{1}{\sqrt{\epsilon}})} = \exp(x) + \mathcal{O}(\epsilon x^2) . \tag{3.4}$$

The scaling factor λ_ϵ is defined as

$$\lambda_\epsilon = \frac{\sqrt{\epsilon}\,\varphi\left(-\frac{1}{\sqrt{\epsilon}}\right)}{\Phi\left(-\frac{1}{\sqrt{\epsilon}}\right)} . \tag{3.5}$$

with the Gaussian distribution

$$\varphi(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) . \tag{3.6}$$

The relation is derived in App. A.2.

With this relation and $\Delta S(\phi', \phi) = S(\phi') - S(\phi)$, the transition probability turns for small values of ϵ into:

$$\begin{aligned} W(\phi \to \phi') &\propto \Phi\left(-\frac{1}{\sqrt{\epsilon}} - \frac{\sqrt{\epsilon}}{2\lambda_\epsilon}\Delta S(\phi', \phi)\right) \\ &= P\left(\tilde{\eta} \leq -\frac{1}{\sqrt{\epsilon}} - \frac{\sqrt{\epsilon}}{2\lambda_\epsilon}\Delta S(\phi', \phi)\right), \end{aligned} \tag{3.7}$$

where the Gaussian noise contribution $\tilde{\eta}$ is uncorrelated and has variance of one,

$$\langle \tilde{\eta}, \tilde{\eta}' \rangle_{\tilde{\eta}} = \delta(t' - t), \qquad \langle \tilde{\eta}_i \rangle_{\tilde{\eta}} = 0. \tag{3.8}$$

Here, t' and t refer to different points in time in the sampling process and $\langle \cdot \rangle_{\tilde{\eta}}$ to the expectation value with respect to the Gaussian distribution. The second line in the above equation (3.7) for the transition probability reflects the property of a cumulative Gaussian distribution to account for samples smaller or equal to the given argument.

Taking the current state ϕ into account, one can transform the sampling from the cumulative normal distribution into a general stochastic update rule with Gaussian noise $\tilde{\eta}$ and a proposal state ν. This leads us to (3.1), which was already presented at the beginning of this chapter,

$$\phi' = \phi + (\nu - \phi)\,\Theta\left[-1 - \frac{\epsilon}{2\lambda_\epsilon}\Delta S(\nu, \phi) + \sqrt{\epsilon}\tilde{\eta}\right], \tag{3.9}$$

where ϵ needs to be chosen sufficiently small and $\Theta(x)$ is the Heaviside function.

The update formalism corresponds to a single spin flip Monte Carlo algorithm with a random sequential update mechanism, driven by Gaussian noise. It can be immediately seen within the present form that a flip to a proposed field gets the more unlikely the smaller ϵ. Adaptations of the Gaussian noise term to truncated Gaussian noise can help to improve the dynamics, i.e., to increase the probability of a spin flip. In principle, this corresponds to a rescaling of the transition probability term. This is similar to a maximization of the spin flip probability in a Metropolis algorithm [253].

The truncated Gaussian noise term can be expressed by the following parametrization:

$$\tilde{\eta}^T \in \left[\frac{1}{\sqrt{\epsilon}} + \alpha, \infty\right], \tag{3.10}$$

where α is in the range of

$$-\infty \leq \alpha \leq -\frac{\sqrt{\epsilon}}{2\lambda_\epsilon}\Delta_{\max}, \quad \text{with} \quad \Delta_{\max} = |\Delta S(\nu, \phi)|. \tag{3.11}$$

The improved update rule is

$$\phi' = \phi + (\nu - \phi)\,\Theta\left[-1 - \frac{\epsilon}{2\lambda_\epsilon}\Delta S(\nu, \phi) + \sqrt{\epsilon}\tilde{\eta}^T\right]. \tag{3.12}$$

For $\alpha \to -\infty$ this reduces to the update formalism (3.9) and for $\alpha = -\frac{\sqrt{\epsilon}}{2\lambda_\epsilon}\Delta_{\max}$ one obtains spin flip probabilities up to 1. This can be seen under consideration of the explicit transition probability

of the update rule (3.12),

$$W(\phi \to \nu) = \frac{\Phi\left(-\frac{1}{\sqrt{\epsilon}} - \frac{\sqrt{\epsilon}}{2\lambda_\epsilon}\Delta S(\nu,\phi)\right)}{\Phi\left(-\frac{1}{\sqrt{\epsilon}} - \alpha\right)} \,. \tag{3.13}$$

Transition probabilities of several standard Monte Carlo algorithms can be emulated by other choices of α. Note that for a uniform random number $r \in [0,1[$ and a proposal field ν, an equivalent formulation to (3.12) can be stated for the transition probability defined in Eq. (3.2),

$$\phi' = \phi + (\nu - \phi)\Theta\left[\exp\left(-\frac{S(\nu) - S(\phi) + \Delta_{\max}}{2}\right) - r\right] \,. \tag{3.14}$$

Processes with a different value of α, i.e., a different rescaling of the transition probability, can always be mapped onto each other by a respective rescaling of the time. Given a transition probability $W(\phi \to \mu)$ and a scaling factor a, the following relation holds:

$$W(\phi \to \mu) \to aW(\phi \to \mu) \quad \Leftrightarrow \quad t \to \frac{t}{a} \,. \tag{3.15}$$

Most of the existing single spin flip algorithms can be reformulated into a Langevin equation for discrete systems with the same derivation, as presented in this section. In general, it holds for the presented dynamics that

$$\lim_{\epsilon \to 0} P_{\text{eq}}(\phi) \propto \exp\left[-S(\phi)\right] \,, \tag{3.16}$$

since the detailed balance equation (2.15) for a Markov chain Monte Carlo algorithm is satisfied in the limit $\epsilon \to 0$. The fulfilment is self-explanatory as we used the transition probability (3.2) as starting point for deriving the novel algorithm.

The update formalism (3.12) represents a Langevin-like equivalent for discrete systems to Langevin dynamics of continuous systems. As for continuous systems, the dynamics depends on Gaussian noise and is based on a rather simple expression. The algorithms can also be applied to continuous systems due to the equivalence to standard Monte Carlo algorithms in the limit $\epsilon \to 0$.

3.2 Application: q-state clock model

The q-state clock model [254, 255] describes spins $\theta_i = \frac{2\pi n}{q}$ with q different states which are parametrised by $n \in \{1, 2, \ldots, q\}$. It is used to numerically verify the Langevin equation for discrete systems, as a first example. The model has the following Hamiltonian:

$$H_c = -J_c \sum_{\langle i,j \rangle} \cos\left(\theta_i - \theta_j\right) \,, \tag{3.17}$$

with J_c representing a coupling constant determining the interaction strength. The sum runs over all nearest neighbor spin pairs $\langle i, j \rangle$ on a rectangular lattice. In a complex plane one can interpret the spin states as equally distributed states on an unit circle. The common Potts model [256] can be derived from this initial model. For $q = 2$ the model corresponds to the Ising model and in the limit of $q \to \infty$, it describes the continuous XY model.

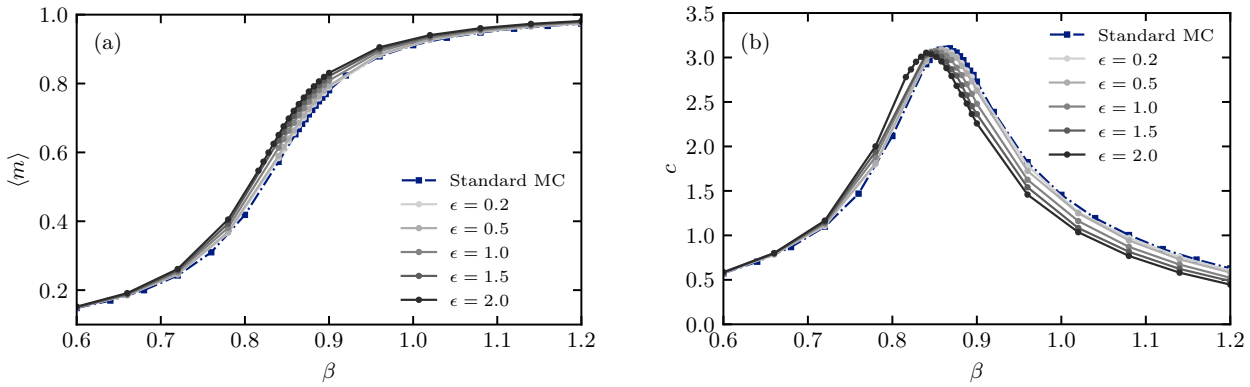

Figure 3.1: Comparison of the magnetization (a) and the specific heat (b) obtained by a standard Monte Carlo algorithm and by the Langevin equation for discrete systems for the 4-state clock model on a 16×16 lattice. The results converge for $\epsilon \to 0$ to the results of the standard Monte Carlo algorithm. Relative deviations of the inverse critical temperatures in dependence of ϵ are illustrated in Fig. 3.2.

The clock model exhibits a second order phase transition for $q \leq 4$. For $q = 4$, an exact solution of the inverse critical temperature exists [257]:

$$J_c \beta_c^{q=4} = 2 J_c \beta_c^{q=2} \,, \tag{3.18}$$

where the Boltzmann constant k_b has been set to 1. In this case, the system emulates two independent Ising models.

An appropriate order parameter for the q-state clock model is the average magnetization per spin defined as

$$m = \frac{1}{N} \left| \sum_{k=1}^{N} e^{\frac{i 2\pi n_k}{q}} \right| , \tag{3.19}$$

where the sum runs over all spins n_k of a lattice with N sites. The specific heat capacity c per spin is considered as a further observable and is defined by means of the energy E of the system,

$$c = \frac{\beta^2}{N} \left(\langle E^2 \rangle - \langle E \rangle^2 \right) , \tag{3.20}$$

where $\langle \cdot \rangle$ denotes the expectation value with respect to sampled lattice configurations [229].

Numerical results for the expectation value of the magnetization and the specific heat are illustrated in dependence on the inverse temperature β and on different values of ϵ in Fig. 3.1. Results of the Metropolis algorithm as a standard Monte Carlo algorithm (MC) serve as benchmark [253]. The inverse critical temperature can be read off from the maximum of the specific heat. In Fig. 3.2, the relative deviations of the inverse critical temperature to the inverse critical temperature of the Metropolis algorithm are plotted against ϵ.

The resulting deviations for finite values of ϵ can be explained by a detailed error analysis of the transition probabilities of the Langevin equation for discrete systems. For this purpose, one has to check the compliance of the detailed balance equation, cf. Eq. (2.15), resulting in

$$\frac{W(\theta \to \theta')}{W(\theta' \to \theta)} \left[1 + \mathcal{O}\left(\epsilon \Delta H_c(\theta', \theta)^3 \right) \right] = \frac{P_{\mathrm{MC}}(\theta')}{P_{\mathrm{MC}}(\theta)} \,. \tag{3.21}$$

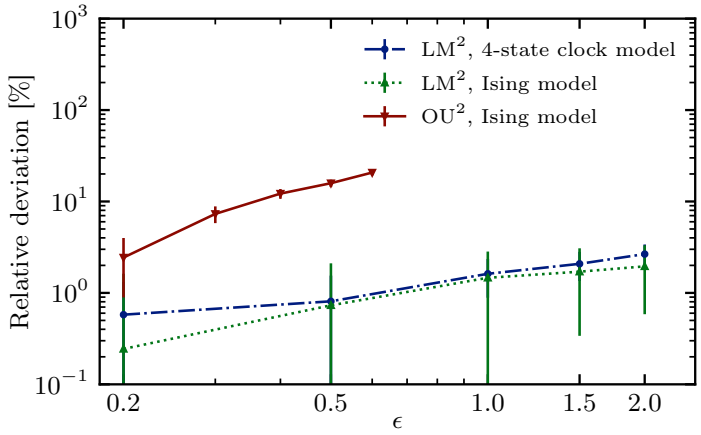

Figure 3.2: Relative deviations of the obtained inverse critical temperatures for finite values of ϵ to the inverse critical temperature of a standard Monte Carlo algorithm. Here, LM^2 refers to results based on the Langevin equation for discrete systems, see also Sec. 4.3.1 and OU^2 to the so-called sign-dependent Ornstein-Uhlenbeck process introduced in Sec. 4.3.2.

The deviation can be traced back to the approximation of the transition probability $W(\theta \to \theta')$ in terms of the cumulative Gaussian distribution in Eq. (3.7) for finite values of ϵ and a respective error propagation in the fraction of the transition probabilities. The analysis reflects that the sampling process is only exact in the limit of $\epsilon \to 0$.

The observed shift of the magnetization and the specific heat to lower values of the inverse temperature in Fig. 3.1 can be explained by taking a closer look at the properties of relation (3.4). In the left part of Fig. A.1, it can be seen that the absolute error of the cumulative Gaussian distribution is asymmetric around $x = 0$. This imbalance leads to a shift of the effective fraction of transition probabilities and therefore to a change of the equilibrium distribution of the spin states. The strength of this shift grows for larger values of x, which corresponds to larger values of $\beta\Delta H_c$, and larger values of ϵ. The effect can be nicely observed in the change of the specific heat in the right part of Fig. 3.1 with growing β. In general, it holds: the larger $|\beta\Delta H_c|$, the worse is the compliance of the detailed balance equation and the larger is the resulting shift of the equilibrium distribution.

3.3 Deriving Langevin dynamics in the limit of infinitesimal step sizes

In the following, we show that the Langevin equation for discrete systems, cf. Eq. (3.1), converges in the limit of infinitesimally small step sizes in the field ϕ to Langevin dynamics, cf. Eq. (2.5). The derivation reveals the similarities between the two dynamics from a pure mathematical point of view.

We start be introducing the following notation for the Langevin equation for discrete systems:

$$\begin{aligned} \phi' &= \phi + (\nu - \phi)\Theta\left[-1 - \frac{\epsilon}{2\lambda_\epsilon}\Delta S(\nu,\phi) + \sqrt{\epsilon}\tilde{\eta}\right] \\ &\equiv \phi + (\nu - \phi)A(\nu,\phi)\,, \end{aligned} \tag{3.22}$$

with

$$\langle \tilde{\eta}, \tilde{\eta}' \rangle_{\tilde{\eta}} = \delta(t' - t)\,, \qquad \langle \tilde{\eta} \rangle_{\tilde{\eta}} = 0\,. \tag{3.23}$$

Note again that, in contrast to Langevin dynamics, the Gaussian distribution has a variance equal to one.

The acceptance term $A(\nu, \phi)$, given by the Heaviside function, can be rewritten in terms of the following approximation of the Heaviside function:

$$\Theta\left[x\right] = \lim_{k\to 0} \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{x}{k}\right)\right] = \lim_{k\to 0} \Phi\left[\frac{\sqrt{2}x}{k}\right]\,, \tag{3.24}$$

where $\Phi[\cdot]$ refers again to the cumulative Gaussian distribution. This leads to a smooth representation of the acceptance term, denoted as $\tilde{A}(\nu, \phi)$:

$$\tilde{A}(\nu, \phi) = \lim_{k\to 0} \Phi\left[-\frac{1}{k} - \frac{\epsilon}{2k\lambda_\epsilon}\Delta S(\nu, \phi) + \frac{\sqrt{\epsilon}}{k}\tilde{\eta}\right]\,, \tag{3.25}$$

where we absorbed the factor of $\sqrt{2}$ in k. As discussed in the previous section, the Langevin equation for discrete systems samples only in the limit of $\epsilon \to 0$ from the Boltzmann distribution. We combine the two limits in k and ϵ while keeping ϵ/k^2 equal to one. The acceptance term then reads:

$$\tilde{A}(\nu, \phi) = \lim_{\epsilon\to 0} \Phi\left[-\frac{1}{\epsilon} - \frac{\epsilon}{2\lambda(\epsilon)}\Delta S(\nu, \phi) + \tilde{\eta}\right]\,, \tag{3.26}$$

where we have replaced k by ϵ. Note that, because of this, the scaling factor has changed from λ_ϵ, cf. Eq. (3.5), to $\lambda(\epsilon)$ defined as,

$$\lambda_{\epsilon^2} \equiv \lambda(\epsilon) = \frac{\epsilon\,\varphi\left(-\frac{1}{\epsilon}\right)}{\Phi\left(-\frac{1}{\epsilon}\right)}\,. \tag{3.27}$$

Next, we aim to expand $\tilde{A}(\nu, \phi)$ around small values of $\Delta S(\nu, \phi)$. This can be achieved by making use of another relation between the cumulative Gaussian distribution and the exponential function,

$$\lim_{\epsilon\to 0} \frac{\Phi\left[-\frac{1}{\epsilon} + \frac{\epsilon}{\lambda(\epsilon,\tilde{\eta})}\,x + \tilde{\eta}\right]}{\Phi\left[-\frac{1}{\epsilon} + \tilde{\eta}\right]} = \exp(x)\,, \tag{3.28}$$

with

$$\lambda(\epsilon, \tilde{\eta}) = \frac{\epsilon\,\varphi\left(-\frac{1}{\epsilon} + \tilde{\eta}\right)}{\Phi\left(-\frac{1}{\epsilon} + \tilde{\eta}\right)}\,. \tag{3.29}$$

The relation can be derived in the same manner as Eq. (3.4), cf. App. A.2. By using additionally that

$$\lim_{\epsilon\to 0} \lambda(\epsilon, \eta) = \lim_{\epsilon\to 0} \lambda(\epsilon)\,, \tag{3.30}$$

we find with the help of Eq. (3.28) the following expression for the acceptance term (3.26):

$$\tilde{A}(\nu, \phi) = \lim_{\epsilon\to 0} \Phi\left[-\frac{1}{\epsilon} + \tilde{\eta}\right] \exp\left(-\frac{\Delta S(\nu, \phi)}{2}\right)\,. \tag{3.31}$$

The transition from the Heaviside function to its continuous approximation has certain implications on the update rule (3.22). Firstly, one needs to correctly normalize the transition probability to new states ϕ'. In contrast to the original formulation, a proposed state ν is no longer accepted or

rejected. Instead, the novel state ϕ' is defined by,

$$\phi' = \phi + (\nu - \phi)\frac{\tilde{A}(\nu, \phi)}{N}\,, \tag{3.32}$$

and can take arbitrary continuous values. The problem is formally resolved in the above update rule by introducing a normalization factor N. Secondly, using the expansion (3.28) is only valid for small values of $\Delta S(\nu, \phi)$. This is taken into account by proposing states ν that are in an infinitesimal small distance $\delta\phi$ to the current state ϕ. We define the following proposal probability

$$q(\phi \to \nu) = \frac{1}{\sqrt{2\varepsilon}}\varphi\left(\frac{\nu - \phi}{\sqrt{2\varepsilon}}\right)\,. \tag{3.33}$$

Infinitesimally small step sizes can be implemented by the limit $\varepsilon \to 0$.

Next, we approximate the difference between the proposed state $\nu = \phi + \delta\phi$ and the current state ϕ in the action according to the finite difference method by

$$\Delta S(\phi + \delta\phi, \phi) = S(\phi + \delta\phi) - S(\phi) \simeq \delta\phi\frac{\delta S}{\delta\phi}\,. \tag{3.34}$$

The update rule (3.32) turns with the above changes into

$$\begin{aligned}
\phi' &= \phi + \delta\phi\,\frac{\tilde{A}(\phi + \delta\phi, \phi)}{N} \\
&= \phi + \delta\phi\,\frac{\lim_{\epsilon\to 0}\Phi\left[-\frac{1}{\epsilon} + \tilde{\eta}\right]}{N}\,\exp\left(-\frac{\delta\phi}{2}\frac{\delta S}{\delta\phi}\right) \\
&= \phi + \tilde{N}^{-1}\left[\delta\phi - \frac{(\delta\phi)^2}{2}\frac{\delta S}{\delta\phi} + \mathcal{O}\left((\delta\phi)^2\right)\right]\,.
\end{aligned} \tag{3.35}$$

In the last line, we expanded the exponential distribution around zero and introduced

$$\tilde{N}^{-1} := \frac{\lim_{\epsilon\to 0}\Phi\left[-\frac{1}{\epsilon} + \tilde{\eta}\right]}{N} \tag{3.36}$$

for better readability. Proposal states can be sampled by

$$\nu = \phi + \sqrt{2\epsilon}\eta\,. \tag{3.37}$$

This can be derived based on a transformation of the probability density of the proposal distribution (3.33) (similar to Eq. (C.7)). According to the above sampling of proposal states, we can replace $\delta\phi$ by $\sqrt{2\epsilon}$ and $(\delta\phi)^2$ by its expectation value of 2ϵ. After reordering the terms, the update rule (3.35) simplifies to:

$$\phi' = \phi - \tilde{N}^{-1}\left[\varepsilon\frac{\delta S}{\delta\phi} + \sqrt{2\varepsilon}\eta\right]\,, \tag{3.38}$$

which corresponds with the exception of the additional normalization factor to the discretised update rule for Langevin dynamics (A.1),

$$\phi' = \phi - \epsilon\frac{\delta S}{\delta\phi_x} + \sqrt{\epsilon}\eta\,. \tag{3.39}$$

This implies that the dynamics is correctly normalized by $\tilde{N}^{-1} = 1$.

3.4 Deriving Complex Langevin dynamics for complex actions

Complex Langevin dynamics can be derived based on a similar way as Langevin dynamics. The first steps of the derivation follow the same reasoning as the one in the previous section. We continue here starting from the intermediate update rule (3.32) and consider complex actions $S(\phi)$, where ϕ is still real-valued.

As a preparation step, we replace ϕ by complex-valued states:

$$\phi \to \phi_x + i\phi_y \tag{3.40}$$

and refer to the complex action as $S(\phi_x, \phi_y) = S_{\mathrm{Re}}(\phi_x, \phi_y) + iS_{\mathrm{Im}}(\phi_x, \phi_y) = S_{\mathrm{Re}} + iS_{\mathrm{Im}}$. It is assumed that a proposed state can only differ from the previous state in the real direction. This assumption can be justified by the fact that also in the first place, changes are only considered in real-valued fields in the intermediate update dynamics (3.32). The complexification is therefore a result of the first update step based on a complex action, but not directly related to the action difference.

For a complex action $S(\phi_x, \phi_y)$, the acceptance term (3.31) turns into

$$\begin{aligned}\tilde{A}\left(\nu + i\phi_y, \phi_x + i\phi_y\right)) = \lim_{\epsilon\to 0} \Phi\left[-\frac{1}{\epsilon} + \tilde{\eta}\right] \times \exp\left(-\frac{\Delta S_{\mathrm{Re}}(\phi', \phi)}{2}\right) \\ \times \left[\cos\left(\frac{\Delta S_{\mathrm{Im}}(\phi', \phi)}{2}\right) - i\sin\left(\frac{\Delta S_{\mathrm{Im}}(\phi', \phi)}{2}\right)\right],\end{aligned} \tag{3.41}$$

with $\Delta S(\phi', \phi) = S(\nu, \phi_y) - S(\phi_x, \phi_y)$. In concordance with the above arguments, the proposed state refers to a change in the real part of the field. The complex acceptance term leads together with Eq. (3.32) to update dynamics for the complex field. These can be specified separately for the real and the imaginary part. For the real part, we obtain

$$\phi'_x = \phi_x + \frac{(\nu - \phi_x)}{N} \exp\left(-\frac{\Delta S_{\mathrm{Re}}(\phi', \phi)}{2}\right) \cos\left(\frac{\Delta S_{\mathrm{Im}}(\phi', \phi)}{2}\right), \tag{3.42}$$

and for the imaginary part

$$\phi'_y = \phi_y + \frac{(\nu - \phi_x)}{N} \exp\left(-\frac{\Delta S_{\mathrm{Re}}(\phi', \phi)}{2}\right) \sin\left(\frac{\Delta S_{\mathrm{Im}}(\phi', \phi)}{2}\right). \tag{3.43}$$

The normalization factor N turns out to be one. This can be again justified at the end of the derivation by a correct normalization of complex Langevin dynamics itself. As before, we assume that this update dynamics is only correct for proposal distributions with an infinitesimal small step size.

The change in the action for infinitesimally small step sizes $\delta\phi_x$ entails simplifications resulting from the following Taylor expansions. The action difference can be written as

$$\begin{aligned}\Delta S &:= S(\nu, \phi_y) - S(\phi_x, \phi_y) \\ &= S(\phi_x + \delta\phi_x, \phi_y) - S(\phi_x, \phi_y) \simeq \delta\phi_x \frac{\delta S(\phi_x, \phi_y)}{\delta\phi_x} \\ &= \delta\phi_x \frac{\delta S_{\mathrm{Re}}(\phi_x, \phi_y)}{\delta\phi_x} + i\delta\phi_x \frac{\delta S_{\mathrm{Im}}(\phi_x, \phi_y)}{\delta\phi_x} \,. \end{aligned} \tag{3.44}$$

The exponential function in Eqs. (3.42) and (3.43) simplifies to

$$\exp\left(-\frac{\Delta S_{\mathrm{Re}}(\phi', \phi)}{2}\right) = 1 - \frac{\delta\phi_x}{2}\frac{\delta S_{\mathrm{Re}}(\phi_x, \phi_y)}{\delta\phi_x} + \mathcal{O}\left((\delta\phi_x)^2\right) \,. \tag{3.45}$$

Additionally, we obtain from an expansion of the sine and the cosine function,

$$\begin{aligned}\cos\left(\frac{\Delta S_{\mathrm{Im}}(\phi', \phi)}{2}\right) &= 1 + \mathcal{O}\left((\delta\phi_x)^2\right) \,, \\ \sin\left(\frac{\Delta S_{\mathrm{Im}}(\phi', \phi)}{2}\right) &= \delta\phi_x \frac{\delta S_{\mathrm{Im}}(\phi_x, \phi_y)}{\delta\phi_x} + \mathcal{O}\left((\delta\phi_x)^2\right) \,. \end{aligned} \tag{3.46}$$

Putting all the different pieces together, the above update rules (3.42) and (3.43) turn into

$$\begin{aligned}\phi'_x &= \phi_x + \delta\phi_x - \frac{(\delta\phi_x)^2}{2}\frac{\delta S_{\mathrm{Re}}(\phi_x, \phi_y)}{\delta\phi_x} + \mathcal{O}\left((\delta\phi_x)^3\right) \,, \\ \phi'_y &= \phi_y - \frac{(\delta\phi_x)^2}{2}\frac{\delta S_{\mathrm{Im}}(\phi_x, \phi_y)}{\delta\phi_x} + \mathcal{O}\left((\delta\phi_x)^3\right) \,. \end{aligned} \tag{3.47}$$

By replacing the $\delta\phi_x$ terms in the same manner as for Langevin dynamics, we finally obtain

$$\begin{aligned}\phi'_x &= \phi_x - \varepsilon\frac{\delta S_{\mathrm{Re}}(\phi_x, \phi_y)}{\delta\phi_x} + \sqrt{2\varepsilon}\eta \,, \\ \phi'_y &= \phi_y - \varepsilon\frac{\delta S_{\mathrm{Im}}(\phi_x, \phi_y)}{\delta\phi_x} \,. \end{aligned} \tag{3.48}$$

The dynamics coincides in the limit of $\varepsilon \to 0$ and with the identifications

$$\begin{aligned}\frac{\delta S_{\mathrm{Re}}}{\delta\phi_x} &= \mathrm{Re}\left[\frac{\delta S}{\delta\phi}\bigg|_{\phi_x + i\phi_y}\right] \,, \\ \frac{\delta S_{\mathrm{Im}}}{\delta\phi_x} &= \mathrm{Im}\left[\frac{\delta S}{\delta\phi}\bigg|_{\phi_x + i\phi_y}\right] \,. \end{aligned} \tag{3.49}$$

with complex Langevin dynamics, cf. Eq. (2.10). Interestingly, we can derive complex Langevin dynamics without any necessary further assumptions on the dynamics.

The normalization can again be obtained by a comparison to complex Langevin dynamics and a respective consideration of the transition probability of the update equation of the real part of the field.

CHAPTER 4

Spiking neural networks on neuromorphic hardware

This chapter is based on Ref. [1].

The present chapter concentrates on the potential of Langevin dynamics for discrete systems, cf. Chapter 3, for a more accurate implementation of Boltzmann distributed systems on the neuromorphic hardware. Due to the spiking character of the system, the focus is on a successful implementation of the statistics of discrete systems with two possible states per neuron. Numerical results are studied for the Boltzmann machine and the Ising model. The term *spiking character* refers to an effective mapping of the continuous membrane potential to an active or an inactive state of a neuron states in an interacting system.

For scientific computations on neuromorphic hardware systems, an exact representation of the activation function of the Boltzmann machine is necessary to obtain accurate statistics. In the present chapter we show in a detailed numerical analysis that small deviations in the activation function of a neuron propagate if a rectangular refractory mechanism of the neurons or interactions between neurons are taken into account. These small deviations have a large impact on the resulting correlation functions and observables. The numerical results demonstrate that a reliable estimation, an understanding and a control of different sources of errors are essential to correctly compute Boltzmann distributed systems in the future.

Following up on the discussion in Sec. 2.3.1, a finite accuracy for the representation of Boltzmann machines can be and even is sufficient for most of the applications. This is in particular the case for applications in the field of deep learning, cf. Chapter 6. In contrast to this, the focus of this chapter is on performing scientific computations on neuromorphic hardware systems implementing LIF dynamics. For this kind of application, it is important to reach a certain accuracy, to know about possible systematic errors and to be able to reliably estimate error bars for the sampled statistics.

Non-linear interaction kernels between the neurons and the spiking character of the single neurons introduce a high complexity to the membrane dynamics on the BrainScaleS-2 chip. Because of this, deriving a closed form for the outcome of simulations with LIF dynamics is infeasible, cf. Sec 2.3. An accurate qualitative and quantitative analysis of Langevin dynamics for discrete systems on the neuromorphic device turns out to be difficult. To overcome this difficulty, we study simplified model descriptions of LIF sampling. This allows us to analyze the impact of particular hardware related properties and sources of resulting errors on different levels of abstraction.

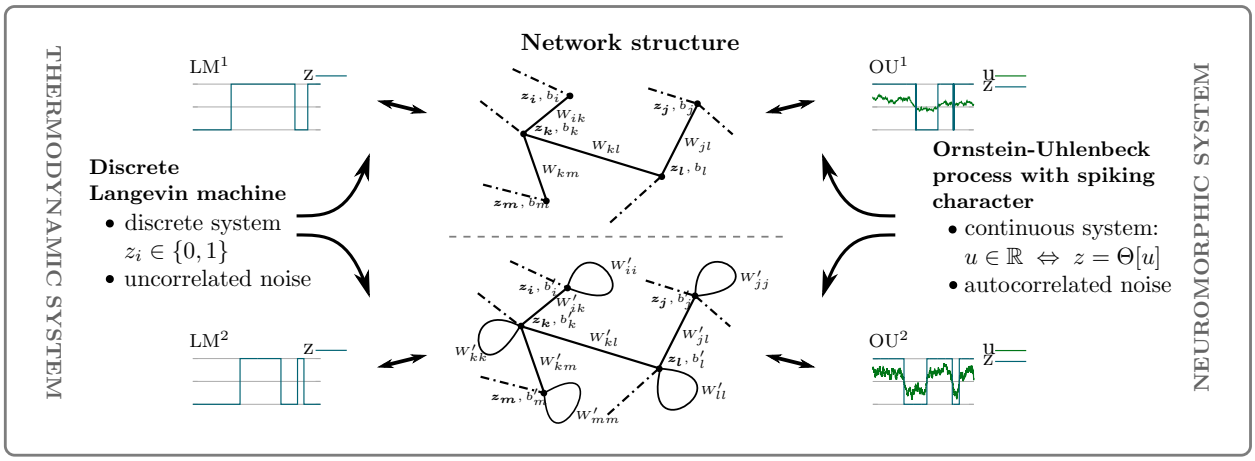

Figure 4.1: Comparison of the commonly used network structure (upper row) and the presented architecture with a self-interacting contribution (lower row). Both network structures can be considered as systems of two discrete states with an uncorrelated noise contribution, which corresponds to different implementations of the discrete Langevin machine. Their continuous counterpart is represented by an Ornstein-Uhlenbeck process with spiking character. The dynamics is based on the temporal evolution of a membrane potential $u := u_{\text{eff}}(t)$. The interaction of neurons relies on a projection of the potential onto two states and enables a comparison with the Langevin machine. The processes on the right-hand side are already very close to the fundamental dynamics of LIF sampling. The different types of dynamics will be introduced throughout this chapter, see Eqs. (4.19) and (4.22) for the LM^2 and the OU^2 process and Eqs. (4.16) and (4.14) for the LM^1 and the OU^1 process, the standard Ornstein-Uhlenbeck process.

The implementation of Langevin dynamics for discrete systems leads to an architecture of neurons based on a self-interacting contribution. The self-interacting term manifestly changes the dynamics of the neural network. For simplified models, this results in activation functions which are much closer to a logistic distribution, the activation function of a Boltzmann machine, than existing approaches. The architecture can be applied to both, discrete two-state systems and neuromorphic hardware systems with a continuous membrane potential and a spiking character. The dynamics differ in their kind of noise contribution that is uncorrelated in the former case and autocorrelated in the latter case. Throughout this chapter, the term autocorrelated noise refers to additive noise contributions to a stochastic process whereas uncorrelated noise is used in the update dynamics but not directly contributing to the updated state. Fig. 4.1 compares the different network structures and gives an overview over existing dynamics and novel dynamics introduced in this chapter.

We start in Sec. 4.1 with a detailed specification of simplified model descriptions for implementing Boltzmann distributed systems by means of LIF dynamics. Sec. 4.2 reviews standard implementations for emulating Boltzmann machines by means of these model descriptions. In Sec. 4.3, we introduce two new sampling processes for the representation of Boltzmann machines by utilizing the Langevin equation for discrete systems. After a short discussion about a possible integration of the refractory mechanism of spiking neural networks in Sec. 4.4, we present in Sec. 4.5 numerical results of the different dynamics introduced in this chapter. The chapter ends with a discussion of related stochastic processes in Sec. 4.6 and a summary in Sec. 4.7.

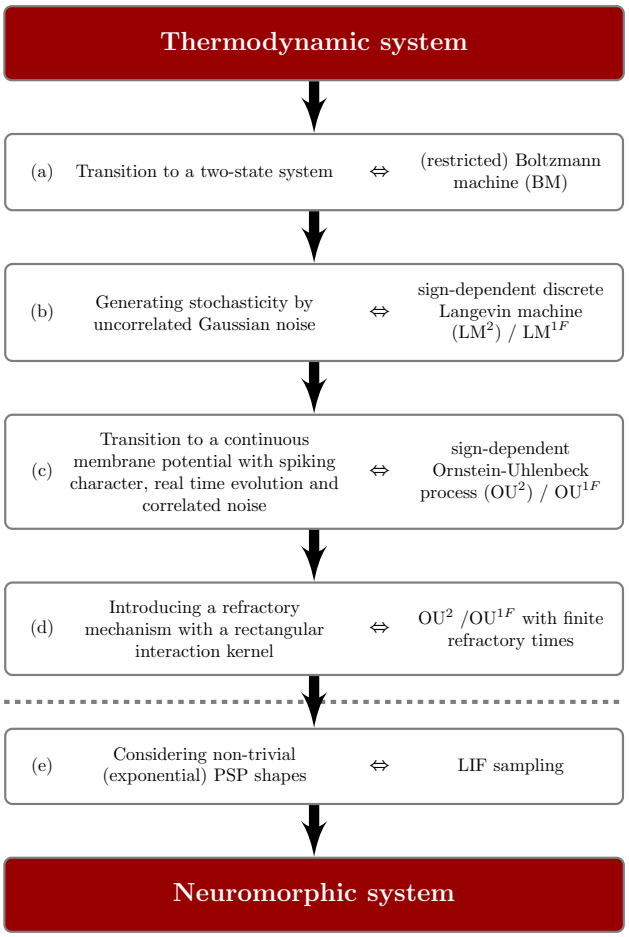

Figure 4.2: Illustration of a step-by-step approach to map thermodynamic systems on a neuromorphic hardware system. The dashed line indicates the progress of the paper. The paper proposes dynamics which have the potential to exactly preserve the properties of the Boltzmann machine up to this line.

4.1 Hardware Abstractions

Several mechanisms of the BrainScaleS-2 chip can be neglected to simplify the underlying dynamics. Having the initial motivation to simulate physical models in mind, we analyze a stepwise mapping of a thermodynamic, Boltzmann distributed system onto the neuromorphic system, as indicated in Fig. 4.2. We choose a step-by-step approach to have control over introduced sources of errors for each of the mechanisms. Each step describes a different level of abstraction of a neuromorphic system. Note that the chosen mechanisms and their order is not the only possible approach for such a projection.

The choice of the different levels was made based on the following properties of LIF sampling:

(1) A description of the microscopic state of a neuron by a continuous membrane potential,

(2) An autocorrelated noise contribution to the membrane potential,

(3) A spiking system with a refractory mechanism and

(4) Non-trivial and nonconstant interaction kernels between neurons.

Inspired by the first two properties of LIF sampling, we distinguish between the so-called discrete Langevin machine and an Ornstein-Uhlenbeck process with spiking character. The two kinds of dynamics differ in their microscopic representation and their noise contribution, cf. Fig. 4.1. Their characteristics are used throughout the chapter as a basis for comparing approaches to represent Boltzmann machines at the different levels of abstraction of LIF dynamics.

4.1.1 Ornstein-Uhlenbeck process with spiking character

Ornstein-Uhlenbeck processes with a spiking character represent the first kind of simplified dynamics. In Fig. 4.2, they refer to the levels of abstraction (c) and (d). The dynamics of the membrane potential of a neuron i is the same as for LIF dynamics in the high-conductance state, cf. Sec. 2.3.1,

$$\frac{du_{i,\mathrm{eff}}(t)}{dt} = \theta\left[\mu_i(t) - u_{i,\mathrm{eff}}(t)\right] + \sigma\tilde{\eta}_i(t)\,, \tag{4.1}$$

with hardware-related parameters θ and σ and with a time-dependent mean value $\mu_i(t) = \mu_i^{\mathrm{leak}} + \mu_i^{\text{average noise}} + \mu_i^{\text{interaction}}(t)$, incorporating synaptic interactions of the neural network. The noise is uncorrelated and has zero mean and unit variance

$$\langle\tilde{\eta}_i(t)\,\tilde{\eta}_j(t')\rangle_{\tilde{\eta}} = \delta(t'-t)\delta(j-i)\,, \quad \langle\tilde{\eta}_i(t)\rangle_{\tilde{\eta}} = 0\,, \tag{4.2}$$

If the refractory mechanism of neurons is neglected, a neuron i can be interpreted as active ($z_i = 1$) if the effective membrane potential is above a certain threshold, and as inactive ($z_i = 0$) otherwise:

$$z_i(t) := \Theta\left[u_{\mathrm{i,\ eff}}(t) - \vartheta\right]\,, \tag{4.3}$$

where ϑ refers to the threshold above which the neuron spikes and $\Theta[\cdot]$ denotes the Heaviside function. The resulting dynamics corresponds to the level of abstraction (c) in Fig. 4.2. For a finite refactory time τ_{ref}, the neuron is considered to be active within this time span, covered by the level of abstraction (d). Different types of this dynamics are depicted on the right-hand side of Fig. 4.1. To preserve the spiking character of the system, interactions between the neurons are also implemented on the basis of the projected neuron states instead of their actual effective continuous membrane potential.

4.1.2 Discrete Langevin machine

The so-called *discrete Langevin machine*, introduced in the following, can be interpreted as a discrete counterpart to the spiking systems discussed in the previous section. The noise is uncorrelated and the dynamics takes place within the two possible discrete states of each neuron. The discrete Langevin machine implements dynamics in the discrete states z_i,

$$z_i' = \Theta\left[\mu_i + \tilde{\eta}_i\right]\,, \tag{4.4}$$

with a Gaussian noise term $\tilde{\eta}_i$ and where μ_i is determined again by interacting contributions and a possible scalar offset. In the flow diagram in Fig. 4.2, this corresponds to dynamics at the level of abstraction (b). The studied dynamics of this kind are shown on the left-hand side of Fig. 4.1. The time scale of the dynamics is given by the computer time. The term *Langevin machine* is chosen because of the respective Gaussian noise term, characteristic for Langevin dynamics, and the latter

observed similar properties of the Langevin equation for discrete systems with a Boltzmann machine, cf. Sec. 4.3. The adjective *discrete* is added to avoid confusion with the Langevin machine presented in Ref. [258].

4.1.3 Mappings between different levels of abstractions

Dynamics of the different levels of abstractions can be mapped onto each other. We will discuss this for different realisations of network architectures for the implementation of Boltzmann distributed systems. In particular, we focus on a mapping of continuous dynamics of the membrane potential, cf. Eq. (4.1), onto dynamics in discrete states, cf. Eq. (4.4) and vice versa.

Denoting the neuromorphic hardware system as $\mathrm{HW}(p)$, with parameters $p = \{W_{ij}, b_i, z_i, z_j, \epsilon\}$, and the discrete Langevin machine as $\mathrm{LM}(h)$, with $h := h(p)$, we suggest the important relation,

$$\{\mathrm{LM}(h(p))\} = \{\mathrm{HW}(p)\}\,, \tag{4.5}$$

i.e., we assume that there exists a discrete two-state system for each set of parameters p of the hardware which emulates the dynamics of the spiking system in a discrete space.

In terms of update dynamics this corresponds to the mapping of the dynamics

$$\frac{du_{i,\mathrm{eff}}(t)}{dt} = \theta\left[\mu_i(p) - u_{i,\mathrm{eff}}(t)\right] + \sigma\tilde{\eta}_i(t)\,, \tag{4.6}$$

and $z_i(t) = \Theta\left[u_{i,\mathrm{eff}}(t) - \vartheta\right]$, onto dynamics in discrete neuron states $z_i \in \{0, 1\}$

$$z_i' = \Theta\left[\mu_i(h(p)) + \tilde{\eta}_i\right]\,, \tag{4.7}$$

for all realisations of p. The information about the network of neurons, parametrized by p is encoded in the mean values $\mu_i(p)$ and $\mu_i(h(p))$.

Relation (4.5) and the formal introduction of the discrete Langevin machine represent a theoretical framework for describing dynamics at different levels of abstractions of LIF sampling. For an exact mapping $h(p)$, the magnitudes of the sources of error have to be matched. Tab. 4.1 gives an overview of the now presented theoretical models for an implementation of Boltzmann distribution systems at different levels of abstractions. Respective systems are given by the Boltzmann machine and the Ising model, whereas the latter can be obtained by a rescaling of the neuron states to spins [259].

In the following, we neglect the finiteness of the refractory time. Therefore, we consider mostly simplified theoretical models of the hardware system at the level of abstractions (b) and (c). A discussion with respect to a refractory mechanism, as a process of the level of abstraction (d), is given in Sec. 4.4.

4.2 Representing Boltzmann machines

In this section, we discuss the implementation of Boltzmann machines (BM) at different levels of abstraction of the neuromorphic hardware. The consideration are similar to the studies in Refs. [77, 234, 259]. A short reminder on Boltzmann machines is given in Sec. 2.3.2. Samples from

	Gibbs sampling (BM)	**LM^{1F}**	**LM^{2}**	**OU^{1F}**	**OU^{2}**	**LM^{1}**	**OU^{1}**
Activation function	Logistic distribution	$\approx$ Logistic distribution	$\approx$ Logistic distribution	$\approx$ Logistic distribution	$\approx$ Logistic distribution	Cumulative Gaussian distribution	Cumulative Gaussian distribution
Microscopic representation	Discrete	Discrete	Discrete	Continuous	Continuous	Discrete	Continuous
Timescale	Computer time	Computer time	Computer time	Real time	Real time	Computer time	Real time
Deviations (free case)	Exact	Small	Small	Small	Small	Exact	Exact
Extrapolation to exact solution?	-	No	Yes	No	Yes	-	-
Deviations (interacting case)	Exact	Medium	Small	Large	Small	Exact	Exact
Extrapolation to exact solution?	-	No	Yes	No	Yes	-	-
Control of a refractory mechanism	Exact [260]	Constant shift $\tau(\tau')$	Constant shift $\tau(\tau')$	Constant shift $\tau(\tau')$	Constant shift $\tau(\tau')$	Nontrivial shift $\tau(\tau', W_{ij}, b_i)$	Nontrivial shift $\tau(\tau', W_{ij}, b_i)$

Table 4.1: Comparison of the different analysed dynamics throughout this chapter. An extrapolation to the exact solution and, therefore, a control of sources of errors is possible for the LM^2 and the OU^2 for both with and without a refractory mechanism.

the Boltzmann distribution, cf. Eq. (2.27),

$$p(\vec{z}) = \frac{1}{Z} \exp(-E(\vec{z})) \tag{4.8}$$

with a partition function Z and

$$E(\vec{z}) = -\sum_{i<j} W_{ij} z_i z_j - \sum_i b_i z_i \,, \tag{4.9}$$

can be drawn by means of a logistic distribution as activation function and an accurate implementation of interactions between neurons,

$$P(z_i = 1) = \sigma \left(\sum_j W_{ij} z_j + b_i \right) \tag{4.10}$$

where $\sigma(\cdot)$ denotes the sigmoid function $\sigma(x) = [1 + \exp(-x)]^{-1}$.

4.2.1 Dynamics in continuous states

We start by considering a mapping onto an Ornstein-Uhlenbeck process with spiking character, cf. Eq. (4.1), as a simplified model at the level of abstraction (c) in Fig. 4.2.

The activation function of the free (without interactions) membrane potential can be computed by means of the equilibrium distribution P_{eq} of the Ornstein-Uhlenbeck process

$$P_{\text{eq}}(u_{i,\text{eff}}) = \sqrt{\frac{\theta}{\pi\sigma^2}} \exp\left(-\frac{\theta(u_{i,\text{eff}} - \mu_i)^2}{\sigma^2} \right) . \tag{4.11}$$

It refers to the probability of the membrane potential to be above the threshold and can be expressed in terms of a cumulative Gaussian distribution $\Phi(\cdot)$,

$$P_{\text{eq}}(z_i = 1) = \int_\vartheta^\infty P_{\text{eq}}(u_{i,\text{eff}}) \mathrm{d}u_{i,\text{eff}} = \Phi\left(\frac{\sqrt{2\theta}}{\sigma} (\mu_i - \vartheta) \right) , \tag{4.12}$$

Fitting the activation function (4.12) to the logistic distribution represents a possible approach to sample Boltzmann distributed neuron states. This approximation can be implemented by a scaling parameter r and a shifting parameter μ^0 according to $P_{\text{eq}}(z_i = 1) = \Phi\left(\frac{\sqrt{2\theta}}{\sigma} \frac{\mu_i - \mu^0}{r} \right)$ [33, 77, 234, 259, 261]. Interactions are taken into account by absorbing their contributions into the mean value μ_i of the Ornstein-Uhlbenbeck process according to: $\mu_i \to \mu_i + \mu_i^{\text{interaction}}$. To be able to make use of this approximation in a network of interacting neurons, we assume that the time to equilibrium is negligible after a change of an interacting neuron.

Note that the more accurate expression of the activation function, defined in Eq. (2.21), takes the finite refractory time into account. The actually measured activation function is somewhere between a logistic distribution and the cumulative Gaussian distribution [259].

A mapping of the Boltzmann machine can be performed straightforwardly by the following identifications:

$$\begin{aligned} \mu_i^{\text{average noise}} &= 0\,, \\ \mu_i^{\text{leak}} &= b_i\,, \\ \mu_i^{\text{interaction}} &= \sum_{\text{syn}j} W_{ij} z_j(t)\,, \end{aligned} \tag{4.13}$$

i.e., by setting the average noise contribution to zero, adjusting the leak potential to b_i, and by taking the interacting contributions into account. Then, based on the dynamics of equation (4.1), the following Ornstein-Uhlenbeck process with spiking character is considered:

$$\frac{du_{i,\text{eff}}(t)}{dt} = \frac{\theta}{r^2}\left[\sum_{\text{syn}j} W_{ij} z_j(t) + b_i - \mu_i^0 - u_{i,\text{eff}}(t)\right] + \sigma\tilde{\eta}_i(t)\,, \tag{4.14}$$

with $z_j(t) = \Theta\left[u_{j,\text{eff}}(t) - \vartheta\right]$ and where $W_{ij} = \frac{A_{ij}}{\alpha}$. For simplicity, the threshold potential ϑ is set to zero in further considerations. The weights need to be adjusted by a scaling factor α in dependence on a certain hardware related interaction factor A_{ij}, as discussed in more detail in Sec. 2.3.1. With simplified parameters, $\sigma = \sqrt{2}$, $\theta = 1$, one obtains the following activation function:

$$P_{\text{OU}^{1\text{F}}}(z_i = 1) = \Phi\left(\frac{\sum_{\text{syn}j} W_{ij} z_j + b_i - \mu_i^0}{r}\right)\,. \tag{4.15}$$

The process is abbreviated in the following by $\text{OU}^{1\text{F}}$. The "1" in the exponent indicates that the process takes place in one regime, i.e., the process does not fluctuate between two fundamental dynamics as it is the case for the sign-dependent processes discussed in Sec. 4.3. The fitting of the activation function to the logistic distribution is indicated by the additional "F". The process without any fitting parameters ($r = 1$, $\mu_i^0 = 0$) is denoted as OU^1.

4.2.2 Dynamics in discrete states

We can also formulate an update rule with the same activation function as in the previous section for a discrete two-state system, i.e., a system without a membrane potential and dynamics in the neurons states z_i. Therefore, the system is built upon an immediate representation of the neuron states. The dynamics represents a possible realisation of a discrete Langevin machine, cf. Sec. 4.1.2. The resulting process corresponds to a transition from the level of abstraction (c) to the level (b) and is driven by uncorrelated Gaussian noise.

The update rule for the level of abstraction (b) is given by,

$$z_i' = \Theta\left[\frac{\sum_{\text{syn}j} W_{ij} z_j + b_i - \mu_i^0}{r} + \tilde{\eta}\right]\,, \tag{4.16}$$

where the updates take place in computer time and with $z_i \in \{0, 1\}$. The corresponding transition probability reads,

$$W_{\mathrm{LM}^{1\mathrm{F}}}(z_i \to 1) = \Phi\left(\frac{\sum_{\mathrm{syn}j} W_{ij} z_j + b_i - \mu_i^0}{r}\right). \tag{4.17}$$

The update rule is studied in Ref. [262] in more detail and introduced in Ref. [31] as an approximation of the so-called *Synaptic Sampling Machine*. In compliance with the OU^1 process, we use the abbreviation LM^1 for the process with $r = 1$ and $\mu_i^0 = 0$. When the activation function is fitted to the logistic distribution, $\mathrm{LM}^{1\mathrm{F}}$ is the corresponding acronym. In the latter case, sources of errors are resulting deviations due to an imperfect fit and finite times to equilibrium if an interacting neuron changes its macroscopic state [262].

Properties and similarities of the two presented processes, i.e., the Ornstein-Uhlenbeck process with spiking character, given in this case by OU^1 / $\mathrm{OU}^{1\mathrm{F}}$, cf. Eq. (4.14), and the discrete Langevin machine, given by LM^1 / $\mathrm{LM}^{1\mathrm{F}}$, cf. Eq. (4.16), are numerically investigated in Sec. 4.5.

4.3 Representing Boltzmann machines by self-interacting neurons

As already pointed out several times, the introduction of a Langevin equation for discrete systems, cf. Chapter 3 was motivated by the similarity of Langevin dynamics and the underlying dynamics of the neuromorphic hardware of the BrainScaleS-2 chip. Finally, we are able to analyse a possible implementation of this kind of dynamics. Similar to the previous section, we investigate models at different levels of abstraction of the neuromorphic hardware system. We focus again on an implementation of the statistics of a Boltzmann distributed system with two possible states per neuron.

4.3.1 Sign-dependent discrete Langevin machine

The Langevin equation for discrete systems (3.12) turns into a rather simple expression for a two-state system. The resulting dynamics is introduced in the following as *sign-dependent discrete Langevin machine* (LM^2) and represents a particular realisation of the *discrete Langevin machine*, cf. Sec. 4.1.2. The LM^2 is implemented by an architecture with the characteristic feature of self-interacting neurons, see also Fig. 4.3. The derived network structure results in dynamics with different weights and biases compared to the Boltzmann machine. It has the unique property that the equilibrium distribution converges in the limit $\epsilon \to 0$ to a logistic distribution which is the activation function of the Boltzmann machine.

We start our derivation by considering the energy function of the Boltzmann machine

$$E(\vec{z}) = -\sum_{i<j} W_{ij} z_i z_j - \sum_i b_i z_i \,, \tag{4.18}$$

with symmetric weights W_{ij} and biases b_i.

For applying the generalized update rule (3.12) of Langevin dynamics for discrete systems, we need the following identifications: $S \to E$ and $\phi_i \to z_i$. As discussed in App. A.4, the following simplified

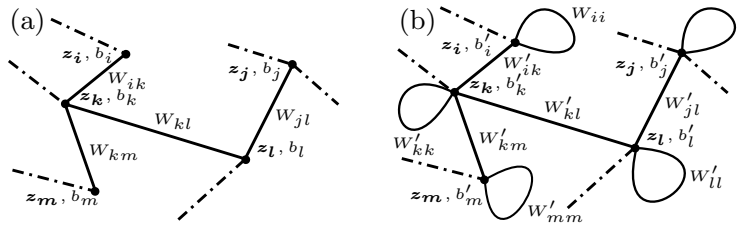

Figure 4.3: Comparison of the structure of a Boltzmann machine (a) and that of the sign-dependent discrete Langevin machine (b). The LM^2 has a self-interacting term and rescaled weights and biases. Nevertheless the dynamics leads in equilibrium to a Boltzmann distribution.

update rule can be derived for the LM^2:

$$z'_i = \Theta\left[W'_{ii} z_i + \sum_j W'_{ij} z_j + b'_i + \tilde{\eta}^T\right] , \tag{4.19}$$

where the transformed parameters are defined as follows: $W'_{ii} = \frac{2}{\sqrt{\epsilon}}$, $W'_{ij} = \frac{\sqrt{\epsilon}}{2\lambda_\epsilon} W_{ij}$, and $b'_i = \left(\frac{\sqrt{\epsilon}}{2\lambda_\epsilon} b_i - \frac{1}{\sqrt{\epsilon}}\right)$. Fig. 4.3 illustrates a comparison between the structure of the Boltzmann machine and the novel update dynamics. The activation function of the LM^2 is given in the limit of $\epsilon \to 0$ by the logistic distribution,

$$\lim_{\epsilon \to 0} P_{\mathrm{LM}^2}(z_i = 1) = \frac{1}{1 + \exp\left[-\sum_j W_{ij} z_j - b_i\right]} . \tag{4.20}$$

This holds since the equilibrium distribution of Langevin dynamics for discrete sytsems is represented by the Boltzmann distribution, the equilibrium distribution of the Boltzmann machine, cf. Eqs. (4.8) and (4.10).

The noise term in the dynamics can be chosen according to equation (3.10), i.e., it can be a Gaussian noise or a truncated Gaussian noise. The self-interaction term $W'_{ii} \in \{0, 2/\sqrt{\epsilon}\}$ and the contribution $-1/\sqrt{\epsilon}$ of the bias b'_i lead in dependency of the state of the neuron for small values of ϵ to a strong shift of the mean value into a positive or negative direction. Respectively, the neuron stays very long in an active regime or in an inactive regime in the case of Gaussian noise. The process fluctuates between two different fundamental descriptions. We emphasize this property by the term *sign-dependent* in the name of dynamics. The exponent "2" in the abbreviation LM^2 indicates the fluctuation between the two regimes.

Instead of performing an implicit update, it is also possible to explicitly compute the probability for an activation of the neuron in the next step. This probability is given by

$$W_{\mathrm{LM}^2}(z_i \to 1) = \Phi\left(W'_{ii} z_i + \sum_j W'_{ij} z_j + b'_i\right) . \tag{4.21}$$

In contrast to the Boltzmann machine, the transition probability is not the same probability as the activation function (4.20). However, the functional from is the same as for the LM^1 and the OU^1 process, which is of advantage for an implementation of Boltzmann distributed statistics on the neuromorphic device, as we will discuss in the next section.

The LM^2 exhibits a totally different dynamics than the Boltzmann machine. The dynamics is characterised by a Gaussian noise term as stochastic input, a self-interacting term and its simplicity in terms of utilized mathematical functions. Transition probabilities and correlation times can be easily controlled by usage of truncated Gaussian noise, cf. Sec. 3.1. As discussed in Chapter 3, finite values of ϵ lead to small deviations in numerically computed observables for the Langevin dynamics for discrete systems. This issue can be resolved by performing simulations for several finite values of ϵ and a subsequent extrapolation of the observables to $\epsilon \to 0$, as verified numerically in Sec. 4.5.

4.3.2 Sign-dependent Ornstein-Uhlenbeck process

Similar to Sec. 4.2, a mapping of the LM^2 can be performed onto an Ornstein-Uhlenbeck process with spiking character, i.e., from the level of abstraction (b) to (c). The resulting process represents a continuous counterpart to the sign-dependent discrete Langevin machine and is referred to as *sign-dependent Ornstein-Uhlbenbeck process* (OU^2). The activation function of the OU^2 process converges in the limit $\epsilon \to 0$ also to a logistic distribution, allowing a sampling of Boltzmann distributed statistics. As illustrated in Fig. 4.1, the two processes differ in their microscopic representation and their timescales. The LM^2 corresponds to a process with two discrete states and the computer time as timescale. In contrast, the OU^2 process describes the temporal evolution of a membrane potential in real time, whereas interactions between neurons are again implemented by spikes, referring here to the information whether the interacting neuron is active or inactive.

The mean value of the Ornstein-Uhlenbeck process in Sec. 4.2.1 is exchanged for a mapping by the associated mean of the sign-dependent discrete Langevin machine: $W'_{ii} z_i + \sum_j W'_{ij} z_j + b'_i$. This leads to the following dynamics:

$$\frac{du_{i,\mathrm{eff}}(t)}{dt} = \theta \left[W'_{ii} z_i(t) + \sum_{\mathrm{syn}j} W'_{ij} z_j(t) + b'_i - u_{i,\mathrm{eff}}(t) \right] + \sigma \tilde{\eta}(t) \,, \tag{4.22}$$

with $z_i(t) = \Theta\left[u_{i,\mathrm{eff}}(t) - \vartheta\right]$. The additional scaling factor of λ_ϵ is omitted since the dynamics implements the activation function of a Boltzmann machine in a slightly different way, as discussed further below. Accordingly, it holds: $W'_{ii} = \frac{2}{\sqrt{\epsilon}}$, $W'_{ij} = \frac{\sqrt{\epsilon}}{2} W_{ij}$, and $b'_i = \left(\frac{\sqrt{\epsilon}}{2} b_i - \frac{1}{\sqrt{\epsilon}}\right)$. The term *sign-dependent* in the name of the process reflects again the property of the neuron to stay very long in an active regime ("+"), or in an inactive regime ("−"), as can be observed in the trajectories of the dynamics in the lower parts of Fig. 4.4. If the membrane potential randomly crosses the threshold $\vartheta = 0$, it perceives a strong drift towards the other regime, due to the changing mean value of the process, i.e., due to the transition $z_i = 0 \to z_i = 1$. An immediate return to its initial regime gets rather unlikely because of the dependence of the mean value on $\frac{1}{\sqrt{\epsilon}}$. Therefore, the self-interaction term in the OU^2 process has significant impact on the resulting equilibrium distribution.

The equilibrium distribution of the process is derived in App. A.3. It is given by

$$P_{\mathrm{eq}}(u_{i,\mathrm{eff}}) \propto \exp\left[K(u_{i,\mathrm{eff}})\right] \,, \tag{4.23}$$

with

$$K(u_{i,\mathrm{eff}}) = \frac{|u_{i,\mathrm{eff}}|}{\sqrt{\epsilon}} + \frac{\sqrt{\epsilon}}{2} \left(\sum_{\mathrm{syn}j} W_{ij} z_j + b_i \right) u_{i,\mathrm{eff}} - \frac{u_{i,\mathrm{eff}}^2}{2} \,. \tag{4.24}$$

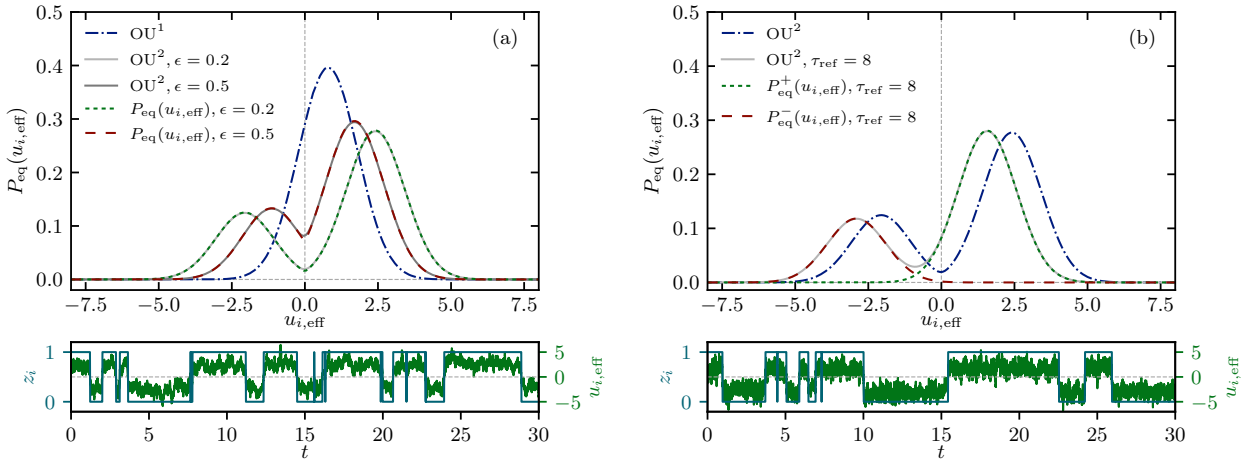

Figure 4.4: Upper plots: Comparison of equilibrium distributions $P_{\mathrm{eq}}(u_{i,\mathrm{eff}})$ for the continuous processes for the free case with bias $b = 0.8$ without a refractory mechanism (a) and with a refractory mechanism with refractory time $\tau_{\mathrm{ref}} = 8$ (b). Lower plots: Corresponding example trajectories of the OU^2 processes with and without a refractory mechanism for $\epsilon = 0.2$. The timescale of the lower plot is rescaled according to the transition probabilities to coincide. The threshold $\vartheta = 0$ is indicated by the dashed horizontal gray line.

The exponent $K(u_{i,\mathrm{eff}})$ contains the absolute value of the effective potential. This property causes a change of the sign of the otherwise Gaussian distribution in dependence of the sign of $u_{i,\mathrm{eff}}$. Therefore, the equilibrium distribution can be considered as the concatenation of two Gaussian distributions $P^+_{\mathrm{eq}}(u_{i,\mathrm{eff}})$ and $P^-_{\mathrm{eq}}(u_{i,\mathrm{eff}})$ with different mean values. In the active regime, the resulting mean value is $\mu_i^+(t) = -\frac{1}{\sqrt{\epsilon}} - \frac{\sqrt{\epsilon}}{2}\left(\sum_{\mathrm{syn}j} W_{ij} z_j(t) + b_i\right)$ and in the inactive regime, it holds $\mu_i^-(t) = -\frac{1}{\sqrt{\epsilon}} + \frac{\sqrt{\epsilon}}{2}\left(\sum_{\mathrm{syn}j} W_{ij} z_j(t) + b_i\right)$. The two distribution are weighted implicitly by the dynamics, resulting in the observed stationary probability distribution. The left part of Fig. 4.4 compares numerically found stationary distributions with the analytically found probability distribution of Eq. (4.23).

The stationary distribution of the process as a function of z_i can be obtained by an integration of $P_{\mathrm{eq}}(u_{i,\mathrm{eff}})$ over $u_{i,\mathrm{eff}}$ with respect ot the threshold ϑ. As derived in App. A.3, this results in the following activation function of the sign-dependent Ornstein-Uhlenbeck process,

$$P_{\mathrm{OU}^2}(z_i = 1) = \frac{1}{1 + \exp\left[\alpha_\epsilon(m_i) \times m_i\right]}\,, \tag{4.25}$$

with $\alpha_\epsilon(m_i)$ being a correction factor. Here, $m_i := -\sum_{\mathrm{syn}j} W_{ij} z_j(t) - b_i$ corresponds to the total input of the neuron. Since $\lim_{\epsilon\to 0} \alpha_\epsilon(m_i) = 1$, the activation function converges in the limit of $\epsilon \to 0$ to the logistic distribution,

$$\lim_{\epsilon\to 0} P_{\mathrm{OU}^2}(z_i = 1) = \frac{1}{1 + \exp\left[-\sum_{\mathrm{syn}j} W_{ij} z_j - b_i\right]}\,, \tag{4.26}$$

the activation function of the Boltzmann machine. The correction factor $\alpha_\epsilon(m_i)$ is different to the scaling factor λ_ϵ of the LM^2 since the convergence to the logistic distribution is caused by different characteristics for the two processes. Fig. 4.5 compares numerically found scaling and correction

factors of the LM^2 and OU^2 process with the theoretical factors λ_ϵ and $\alpha_\epsilon(m_i)$ for different biases b in the free case, i.e., for the activation function without interacting neurons.

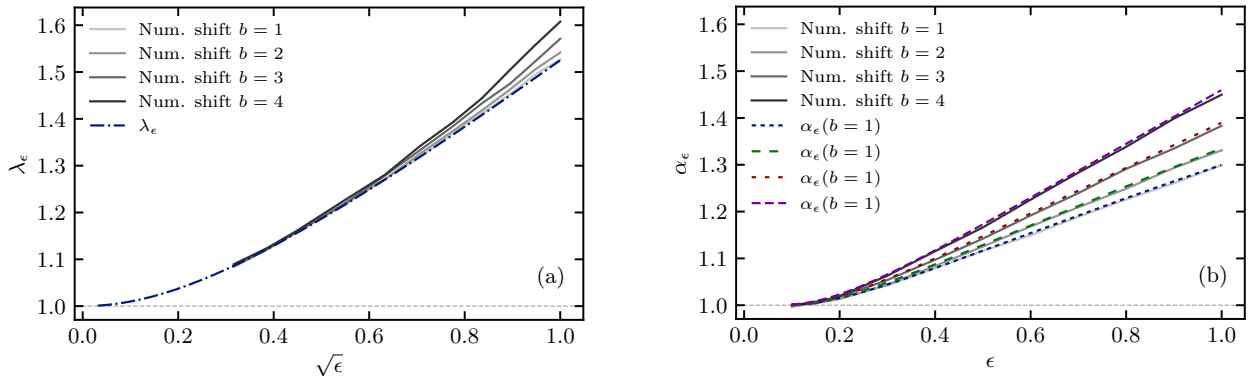

Figure 4.5: Comparison of scaling and correction factors for the sign-dependent processes for a better convergence to the logistic distribution: (a) The numerical and the analytic scaling factor $\lambda_\epsilon = \lambda(\sqrt{\epsilon})$ for the sign-dependent Langevin machine. (b) The numerical and analytic correction factors α_ϵ for the sign-dependent Ornstein-Uhlenbeck process.

The overlap of the tails of the two shifted Gaussian distributions is the reason for necessity of a correction factor. The two distributions do not overlap at all in the limit of $\epsilon \to 0$. Hence, the activation function corresponds to the logistic distribution only in this case. For larger values of ϵ, the distributions are closer together and crossings between the active and the inactive state take place more often. This property results in the observed deviations to the logistic distribution.

An advantage of the sign-dependent Ornstein-Uhlenbeck process is the possibility to extrapolate results for different values of ϵ to the limit of $\epsilon \to 0$, i.e., to exact results of the Boltzmann machine. A disadvantage of the process is that smaller values of ϵ lead to larger correlation times and therefore to a higher simulation cost. This results from the limitation that it is not possible to accelerate the dynamics by an adaptation of the noise source, as it is the case for the LM^2. From another perspective, this property might even help to straighten out problems related to the hardware, like nontrivial postsynaptic shapes, for example.

4.4 Refractory mechanism

A possible further step towards LIF sampling is to take into account a refractory mechanism of the neurons, as indicated by the level of abstraction (d) in Fig. 4.2.

Beyond that, a refractory mechanism can also be considered for dynamics in a discrete system, for example, for a standard implementation of a Markov chain Monte Carlo algorithm for a Boltzmann machine, as discussed in Ref. [260]: A neuron stays active for the refractory time τ_{ref}, after it got activated, whereby the refractory time is measured by the number of Monte Carlo sweeps. The introduced refractory mechanism causes an imbalance between the active and the inactive state. The resulting asymmetry can be compensated by reducing the transition probability to become active by a factor of $1/\tau_{\text{ref}}$, cf. Ref. [260]. The factor can be absorbed into the membrane potential by a shift of the activation function by $\log(\tau_{\text{ref}})$, i.e., by $b_i \to b_i - \log(\tau_{\text{ref}})$. The Markov chain Monte Carlo algorithm performs updates based on the logistic distribution (4.10) as activation function.

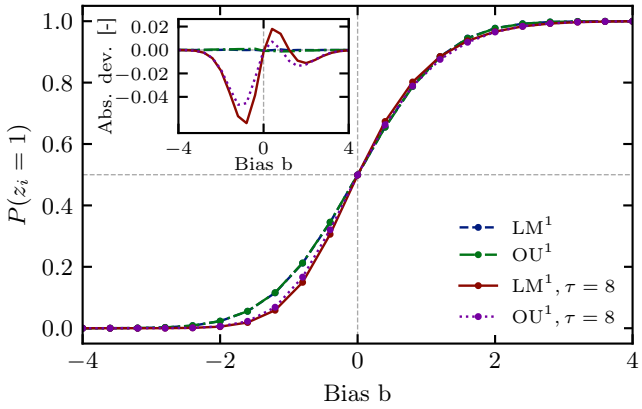

Figure 4.6: Illustration of the nonsymmetric deformation of the LM^1 and the OU^1 process for larger refractory times. The activation functions are shifted to comply: $P(z_i = 1)\big|_{b=0} = 0.5$. The small plot contains the absolute deviation to the cumulative Gaussian distribution.

In light of this approach, we want to point out that such a compensation no longer works for the LM^1 and the OU^1 process. By following the proof for the case of a logistic distribution in Ref. [260], it is easy to see that such an absorption of the refractory time is non-trivial or not possible at all for the cumulative Gaussian distribution. Instead, the activation function with a finite refractory time is deformed, as shown in Fig. 4.6. The deformation gets larger for larger refractory times.

The impact of the refractory mechanism on possible approximations of the activation function of the Boltzmann machine is analysed for the proposed dynamics in more detail in Sec. 4.5.2.

For completeness, we want to shortly comment the last level of abstraction of Fig. 4.2. At this level, interactions between neurons are in general not constant. The interaction is governed by the postsynaptic potential (PSP), the received input signal of an interacting neuron [77], cf. Sec. 2.3.1. In Sec. 2.3.1, the relation between a correct implementation of the weights based on a non-linear interaction kernel is discussed in more detail. Non-linear PSP shape induce further complexity to the system of neurons. An investigation of exponential PSP shapes is postponed to future work. A similar analysis with respect to the dynamics presented in Ref. [260] can be found in Ref. [259].

4.5 Numerical results: neuromorphic hardware versus Langevin machine

Numerical results are discussed for the introduced representation of Boltzmann distributed systems at different levels of abstraction of the neuromorphic hardware system. In Secs. 4.5.1 and 4.5.2, we start with a comparison of the dynamics and equilibrium distributions of the free membrane potential for the discrete Langevin machine and for abstractions of the neuromorphic hardware system, according to Fig. 4.2 and Tab. 4.1. The focus is on a correct implementation of the logistic distribution of the Boltzmann machine and on a detailed analysis of the impact of different sources of errors. The systems are considered with and without an asymmetric refractory mechanism with a rectangular postsynaptic shape. The section ends with a computation of the Ising model by a projection of the model on the Boltzmann machine and with a numerical investigation of a Boltzmann machine with three neurons in Sec. 4.5.3. Both models serve as a benchmark for Boltzmann distributed systems with interacting neurons.

4.5.1 Free membrane potential

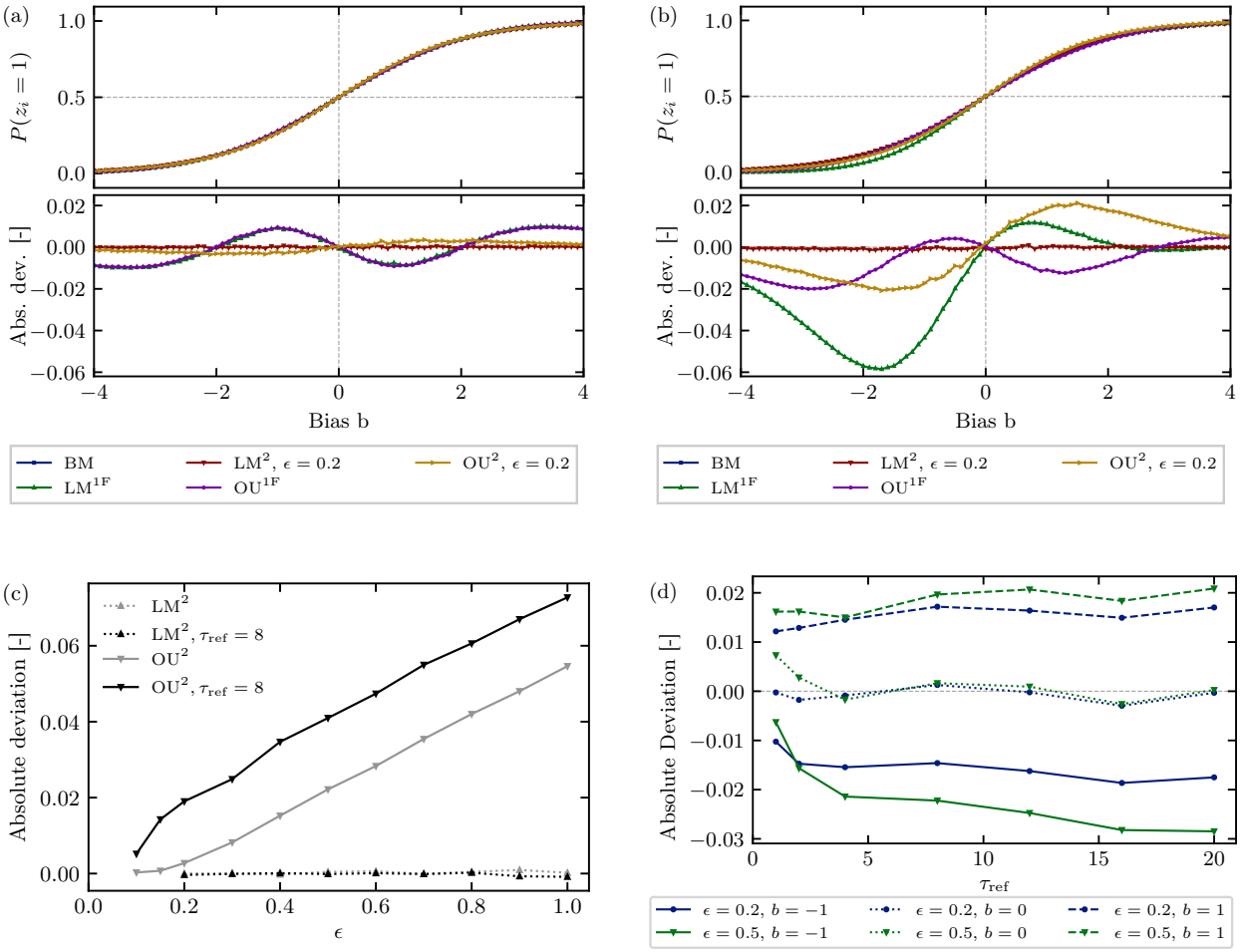

Figure 4.7: Comparison of different properties of the activation function for the different processes with and without a refractory mechanism: (a) Activation function and absolute deviation to the logistic distribution without a refractory mechanism. (b) The same as in (a), but with a refractory mechanism with refractory time $\tau = 8$. (c) Absolute deviation of the exact results of the activation for $b = 1$ for the sign-dependent processes in dependence of ϵ. The deviation converges in the limit $\epsilon \to 0$ for all processes to zero, with and without a refractory mechanism. (d) Absolute deviation of the results for the OU^2 process without a refractory mechanism to the results with a refractory mechanism and different values of τ_{ref}. The deviations are compared for $b = \{-1, 0, 1\}$.

We numerically analyze the mapping between dynamics of the discrete Langevin machine and the continuous dynamics according to relation (4.5) by an explicit consideration of transition probabilities. It is discussed the impact of deviations in the transition probabilities as well as a mapping of the temporal evolution of the different processes onto each other.

Differences of the two dynamics which are given by construction are illustrated in Fig. 4.1. The processes correspond to the levels of abstraction (b) and (c) of a neurmorphic hardware system in Fig. 4.2. The essential differences are the source of noise, which is for the Langevin machine uncorrelated and for the Ornstein-Uhlenbeck process correlated, as well as the representation of a microscopic state. The dynamics is described in the former case by two discrete states in computer time and in the latter one by the evolution of a continuous membrane potential with spiking

character in real time. In particular, we analyze deviations to the expected logistic distribution for the sign-dependent and for the fitted processes: LM^2, OU^2, $\mathrm{LM}^{1\mathrm{F}}$, and $\mathrm{OU}^{1\mathrm{F}}$.

Activation function

Fig. 4.6 and the upper left part of Fig. 4.7 compare numerical results of the activation functions of the free membrane potential in dependency of the bias b in the network for the different presented dynamics. The results of the LM^1 and the OU^1 process in Fig. 4.6 coincide exactly and their deviation to the cumulative Gaussian distribution emerges from numerical errors. In concordance to these observations, the fitted $\mathrm{LM}^{1\mathrm{F}}$ and $\mathrm{OU}^{1\mathrm{F}}$ process have the same deviations to the logistic distribution, cf. Fig. 4.7. In the case of the LM^2 and the OU^2 process, the observed deviations mirror the theoretically derived errors for finite values of ϵ. As shown in the lower left part of Fig. 4.7, both activation functions converge in the limit of $\epsilon \to 0$ to the expected probability. The rate of convergence of the OU^2 process is much smaller than the one of the LM^2 for equal values of ϵ. This can be reasoned by the different sources of errors for the two processes, as discussed in detail at the end of Sec. 3.2 for the Langevin equation for discrete systems and at the end of Sec. 4.3.2 for the sign-dependent Ornstein-Uhlenbeck process. In contrast to the $\mathrm{LM}^{1\mathrm{F}}$ and $\mathrm{OU}^{1\mathrm{F}}$, the only limiting factor for deviations to the logistic distribution are resulting larger correlation times for smaller values of ϵ for the OU^2 process. For the LM^2, these can be suppressed by truncated Gaussian noise, cf. Sec. 3.1.

Dynamics: time evolution

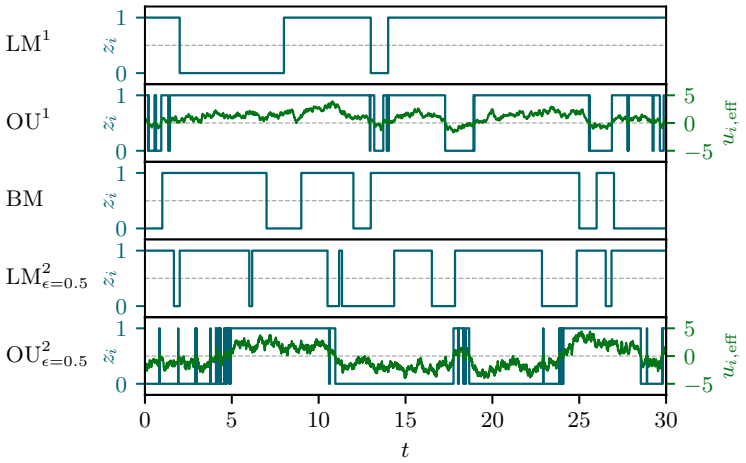

Figure 4.8: Trajectories of the neuron state and the membrane potential in computer time for the different processes with a uniform timescale.

It has been found numerically that the computer time and the real time coincide for the LM^1 and the Ornstein-Uhlenbeck process. All simulations in real time are performed with finite time steps of 0.02. All processes in computer time are computed with a random sequential update formalism and in real time by a parallel update scheme. The timescale in all figures are chosen in units of the computer time.

Fig. 4.8 compares trajectories of the different discussed processes with respect to a uniform timescale. It can be observed in the evolution of the membrane potentials that there occur fast changes if the

	$W_{\mathrm{BM}}(0 \to 1)$	$W_{\mathrm{LM}^2}(0 \to 1)$	$W_{\mathrm{OU}^2}(0 \to 1)$
$\tau_{\mathrm{ref}} = 1$	$\sigma(-m_i)$	$\Phi\left(-\frac{1}{\sqrt{\epsilon}} - \frac{\sqrt{\epsilon}}{2\lambda_\epsilon} m_i\right)$	$\varphi\left(-\frac{1}{\sqrt{\epsilon}} - \frac{\sqrt{\epsilon}}{2} m_i\right)$
$\tau_{\mathrm{ref}} > 1$	$\sigma(-(m_i + \log(\tau_{\mathrm{ref}})))$	$\Phi\left(-\frac{1}{\sqrt{\epsilon}} - \frac{\sqrt{\epsilon}}{2\lambda_\epsilon}(m_i + \log(\tau_{\mathrm{ref}}))\right)$	$\varphi\left(-\frac{1}{\sqrt{\epsilon}} - \frac{\sqrt{\epsilon}}{2}(m_i + \log(\tau_{\mathrm{ref}}))\right)$

Table 4.2: Transition probabilities from an inactive state to an active state for the different considered dynamics with ($\tau_{\mathrm{ref}} > 1$) and without ($\tau_{\mathrm{ref}} = 1$) a refractory time. m_i corresponds to the total input for a neuron i, according to App. A.4.

membrane potential is close to the threshold value $\vartheta = 0$. These perturbations seem to have no influence on the time evolution and the equilibrium distribution.

As discussed in Sec. 3.1, a scaling factor a can be found for a correct mapping of the temporal evolution of two processes A and B if both processes exhibit the same equilibrium distribution. According to Eq. (3.15), the scaling factor can be computed by means of the transition probabilities

$$a = \frac{W_{\mathrm{A}}(0 \to 1)}{W_{\mathrm{B}}(0 \to 1)}\,. \tag{4.27}$$

Analytic expressions for the transition probabilities of the considered sign-dependent processes are given in Tab. 4.2. The given transition probabilities have been validated numerically. For that purpose we have mapped the temporal ensemble evolution of the different dynamics onto the evolution of the Boltzmann machine with respect to the computed scaling factors. A scaling factor $a \neq 1$ reflects an increase/decrease of the correlation time for processes with different transition probabilities. In Fig. 4.9, the dependency of the scaling factor a on ϵ is plotted for the sign-dependent processes.

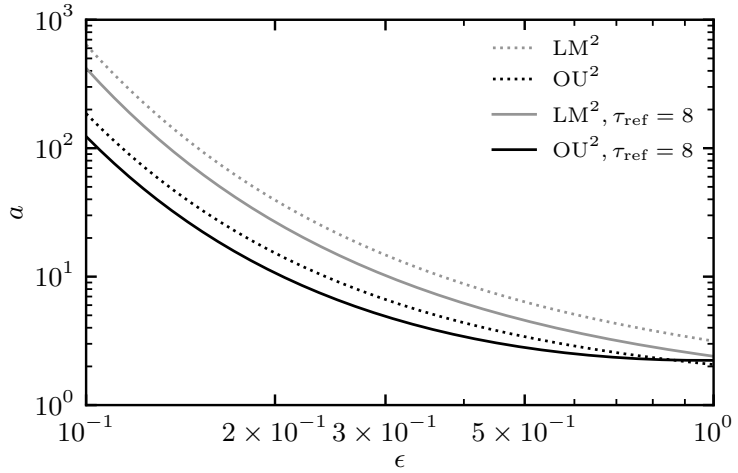

Figure 4.9: Scaling factors a in dependence of ϵ for a mapping of the transition probabilities of the sign-dependent processes onto the transition probability of the Boltzmann machine and, hence, of the temporal evolution on the computer time. The scaling factors are computed for the free case with a bias $b = 0$.

The considerations of the time evolution reinforce that the relation (4.5) corresponds to an exact mapping of the dynamics of a discrete system with uncorrelated noise to a continuous system with correlated noise. This property is not self-evident. However, the dependency $h(p)$ is in some cases non-trivial, due to different occurring sources of errors of the considered models.

4.5.2 Refractory mechanism

In this section we investigate the impact of a refractory mechanism of a neuromorphic system with a rectangular PSP shape, cf. Sec. 4.4.

For an analysis of the activation functions of the self-interacting processes, it is important to keep in mind what kind of impact the refractory mechanism has on the dynamics. Firstly, it corresponds to the property of the neuron to stay active for a certain refractory time after a spike was triggered and, secondly, it has in impact on the synaptic input to a connected neuron. The latter needs only to be taken into account if interactions between neurons are considered. In particular, this entails for the free case that only the activation functions of the sign-dependent processes are affected by the PSP shape, due to their self-interacting contribution.

In the following, we analyse how the logistic function can be correctly represented with a refractory mechanism for the presented dynamics to sample Boltzmann distributed statistics. The upper right part of Fig. 4.7 compares the impact of a refractory mechanism on the different dynamics regarding their deviations to the logistic distribution.

As discussed in Sec. 4.4, the imbalance between the inactive and the active state can in general not be compensated entirely by a trivial shift of the bias by $\log(\tau_{\text{ref}})$ for the cumulative Gaussian distribution. However, as long as the activation function is sufficiently close to a logistic distribution, a shift of the bias according to

$$b_i \to b_i - \log(\tau'_{\text{ref}}) \tag{4.28}$$

can still be used to compensate the refractory mechanism in the activation function. The optimal value for τ'_{ref} of the activation function is determined by a numerical simulation based on the constraint that $p(z_i = 1)\big|_{b=0} = 0.5$. We introduced a further time constant τ'_{ref} to distinguish clearly between the refractory time τ_{ref} of a neuron and the resulting optimal shift $\log(\tau'_{\text{ref}})$ for an approximation of the logistic distribution. Ideally, one can derive a dependency $\tau'_{\text{ref}}(\tau_{\text{ref}})$ to obtain a consistent approximation of the activation function for different refractory times.

For the $\text{LM}^{1\text{F}}$ and the $\text{OU}^{1\text{F}}$ process, the resulting shifts $\log(\tau'_{\text{ref}})$ are slightly different to $\log(\tau_{\text{ref}})$ as a consequence of a non-symmetric deformation of the cumulative Gaussian distribution for larger refractory times, cf. Fig. 4.6 and the discussion in Sec. 4.4. Resulting deviations in the activation function of the LM^1 and the OU^1 process can be compensated to a certain extent by an adaptation of the variance, i.e., of the scaling parameter r, cf. upper right part of Fig. 4.7. By contrast, it holds for the LM^2 process that $\tau_{\text{ref}} \simeq \tau'_{\text{ref}}$ since the deviation of the activation function to the logistic distribution is nearly symmetric around $b = 0$. Nevertheless, the shift amplifies deviations of the transition probability for finite values of ϵ, leading to a worse statistics, as discussed below.

The numerical results in the upper right part of Fig. 4.7 show that deviations of the LM^2 and the OU^2 process have increased, as expected by the introduced asymmetry of the refractory mechanism. Nevertheless, the error vanishes for $\epsilon \to 0$, as illustrated in the lower left part of Fig. 4.7. Furthermore, the lower right part of Fig. 4.7 shows that a further increase of the refractory time has a very low impact on the deviations which encourages an applicability for large refractory times, in practice.

For the OU^2 process, the shift of the bias by $\log(\tau'_{\text{ref}})$ is much larger than $\log(\tau_{\text{ref}})$. Fig. 4.10 illustrates numerically found relations $\tau'_{\text{ref}}(\tau_{\text{ref}})$ and $\tau'_{\text{ref}}(\epsilon)$. The large differences in τ_{ref} and in τ'_{ref} can be traced back to a differing impact of the refractory mechanism on the dynamics of the membrane potential, caused by the self-interacting term in the process. This can be explained as follows.

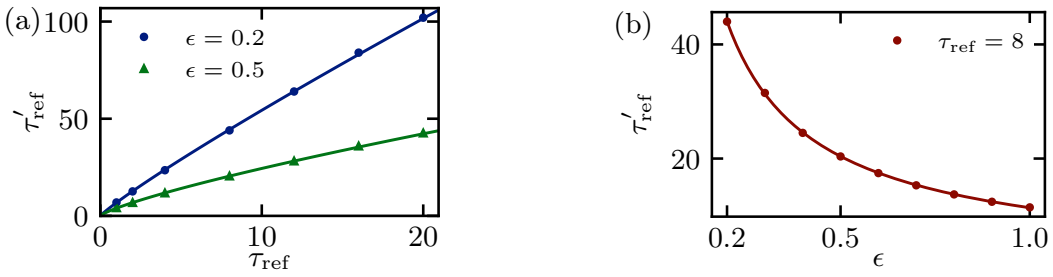

Figure 4.10: τ'_{ref} in dependence on: (a) the refractory time τ_{ref} for $\epsilon = 0.2$ and $\epsilon = 0.5$ and (b) ϵ for $\tau_{\text{ref}} = 8$ for the OU^2 process. Both dependencies obey a power law.

In contrast to the OU^1 process, the dynamics of the OU^2 process fluctuates between the active and the inactivate regime as a result of the self-interacting contribution. By integrating a refractory mechanism, changes from the active to the inactive regime are suppressed as long as the neuron is captured in its refractory mode. This can be seen in the lower right plot of Fig. 4.4, where the membrane potential crosses from time to time the threshold $\vartheta = 0$ while the neuron stays active. Consequently, the tails of two Gaussian distributions, $P^+_{\text{eq}}(u_{i,\text{eff}})$ and $P^-_{\text{eq}}(u_{i,\text{eff}})$, have a dissimilar impact around the threshold on the resulting distribution of $P(u_{i,\text{eff}})$. The refractory time biases the lower tail distribution of $P^+_{\text{eq}}(u_i)$, i.e., the part of the distribution for $u_{i,\text{eff}} < \vartheta$, within the refractory time. In contrast, the upper tail distribution of $P^-_{\text{eq}}(u_{i,\text{eff}})$ does not affect $P(u_{i,\text{eff}})$, since the dynamics always changes for $u_{i,\text{eff}} > \vartheta$ from the inactive to the active regime. The local minimum of $P(u_{i,\text{eff}})$ around $u_{i,\text{eff}} = \vartheta$ as well as the entire distribution $P(z_i)$ are shifted to smaller values, as a result of this asymmetry, illustrated in the upper right part of Fig. 4.4. Furthermore, the absolute value of the minimum is larger than the one for the process without a refractory mechanism. We will see that this imbalance between $P^+_{\text{eq}}(u_{i,\text{eff}})$ and $P^-_{\text{eq}}(u_{i,\text{eff}})$ is the reason for larger deviations of the activation function for the OU^2 process with a refractory mechanism.

Note that the underlying dynamics of the OU$^{1\text{F}}$ process is not affected by the refractory mode due to the absence of a self-interacting term. Therefore, in this case, the only purpose of the shift by $\log \tau'_{\text{ref}}$ is to fix the transition probabilities to correctly compensate the emerging asymmetry of the refractory mechanism.

4.5.3 Interacting systems

The Ising model [263] and the Boltzmann machine [243] are suitable systems to investigate the presented abstractions of a neuromorphic hardware system with interactions between neurons. The Ising model can be easily mapped onto the Boltzmann machine. A numerical analysis can be understood as a proof of concept that the presented processes also work in a more complex network setup. As a second model, we study a Boltzmann machine with three neurons. We compare the results for all presented models with and without a refractory mechanism with a rectangular PSP shape.

The Ising model describes a two-state spin system. The spin states are $s_i \in \{-1, +1\}$, which are likewise also referred to as spin up and spin down $s_i \in \{\downarrow, \uparrow\}$. The Hamiltonian is defined as

$$H = -J \sum_{\langle i,j \rangle} s_i s_j - h \sum_i s_i \,. \tag{4.29}$$

The external magnetic field h is set to zero in the following numerical analysis and $J = 1$ is the coupling constant for interacting term. For this particular case, we can consider the averaged absolute value of the magnetization per spin as an order parameter, given by

$$m = \frac{1}{N}\left|\sum_{i}^{N} s_i\right|, \tag{4.30}$$

where the sum runs over all spins of the lattice for a given configuration. From theoretical considerations, an exact expression for the inverse critical temperature β_c of the model can be obtained in the case of a vanishing external field [264],

$$J\beta_c = \frac{\ln\left(1+\sqrt{2}\right)}{2}. \tag{4.31}$$

For a computation with the presented algorithms, we need a mapping between the Boltzmann machine and the Ising model onto the correct domain of definition. The mapping of $s_i = -1 \to z_i = 0$ and $s_i = 1 \to z_i = 1$ can be obtained by the following identifications between J and h and the weights W_{ij} and biases b_i of a Boltzmann machine [234, 265]:

$$\begin{aligned} W_{ij} &= 4J\,, \\ b_i &= 2h - 2Jd\,, \end{aligned} \tag{4.32}$$

where d corresponds to the dimension of the system. The spin state can be computed by $s_i = 2z_i - 1$.

The Boltzmann machine can have an arbitrarily complex network structure. Particular implementations like the restricted Boltzmann machine turn the Boltzmann machine to an interesting class of networks, which has many applications in different areas of research; see, e.g., Refs. [37–39, 266]. To study the impact of systems with a higher possible variability, we consider a Boltzmann machine with three neurons and different weights and biases around zero. The Kullback-Leibler divergence [267] serves as a measure to numerically classify the quality of the presented processes. We compute the Kullback-Leibler divergence based on the history of a process, starting from a random initial state according to

$$D_{\mathrm{KL}}\left(P_{\mathrm{BM}}||P_{\mathrm{AM}}\right) = -\sum_{c\in\Omega} P_{\mathrm{BM}}(c)\log\frac{P_{\mathrm{AM}}(c)}{P_{\mathrm{BM}}(c)}\,. \tag{4.33}$$

The index BM indicates the exact probability distribution of the Boltzmann machine and the AM refers to the resulting approximated probability distribution of the presented dynamics. The sum runs over all possible neuron configurations c. The probabilities are approximated by respective histograms of the history of the trajectory in the configuration space.

The upper row in Fig. 4.12 shows the absolute value of the magnetization for the Ising model with a vanishing external magnetic field for the dynamics without and with a refractory mechanism. The absolute magnetization is computed for the LM^2 and the OU^2 process for different values of ϵ as well as for the $\mathrm{LM}^{1\mathrm{F}}$ and the $\mathrm{OU}^{1\mathrm{F}}$ process. In addition, the deviation of the derived inverse critical temperatures is plotted as a function of ϵ for the processes without a refractory mechanism in Fig. 3.2. Fig. 4.11 confirms a convergence of the considered processes for vanishing values of ϵ. The deviations for finite values of ϵ show that errors in the representation of the activation function affect the dynamics of interacting systems in a similar, or even worse, way as in the free case.

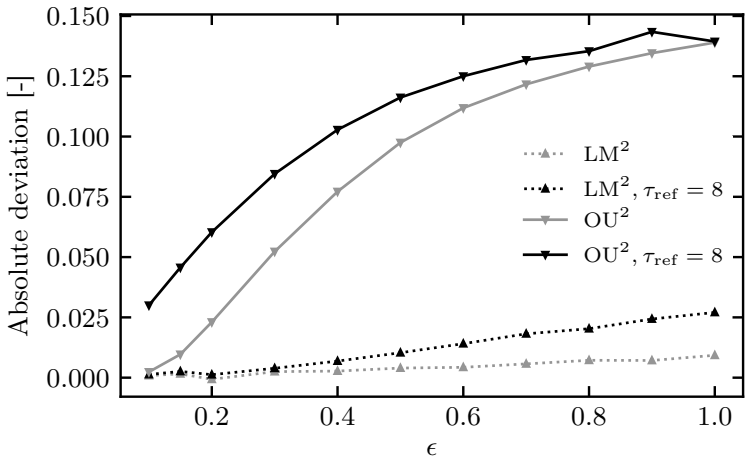

Figure 4.11: Absolute deviation of the exact results of the Ising model for the absolute magnetization m at the inverse critical temperature for the sign-dependent processes in dependence of ϵ. The curves converge for all methods and all dynamics for smaller values of ϵ to the results of the Metropolis algorithm. The rate of convergence differs and signifies dependencies on the properties of the model, the intrinsic parameters and the update dynamics. Minor differences of the OU^2 process occur due to finite time steps in the simulation.

The comparison of the Kullback-Leibler divergence of the different processes in the lower row in Fig. 4.12 reinforces the better representation of the logistic distribution by the sign-dependent processes and illustrates again the dependency on ϵ.

Finally, it is interesting to note that the deviations in the magnetization show for the sign-dependent processes the same tendency as the results for the 4-state clock model. The equilibrium distributions are shifted to smaller values of β as described in the discussion of Sec. 3.2. As before, the shift grows with larger values of β and of ϵ. The similar behavior of the OU^2 process can be justified by the similar trend in the deviation of the activation functions of the two processes. The higher rate of convergence of the LM^2 process is a result of a different source of errors.

4.6 Relations to further stochastic processes

Due to its novelty, there are plenty of open questions regarding the fundamental properties and characteristic of the proposed sign-dependent dynamics. These concern, for example, investigating different kinds of sources for noise or inspecting a transferability on other underlying neuron models. Another important aspect is the integration and embedding of the introduced dynamics into similar existing dynamics, also with respect to other areas of applications. This allows a deeper understanding of interrelations and similarities of the developed dynamics with regard to systems of stochastic differential equations, in general. The modelling of population dynamics represents one of such areas of application. Multiplicative noise sources are often introduced in those models to imitate the stochastic impact of the environment. For example, the Verhulst model is considered in Ref. [268] as a Langevin equation with a model-related drift term and multiplicative noise. In Refs. [269–271], noisy systems of Lotka-Volterra equations are studied in a time-discrete description on a coupled map lattice as well as in their continuous form in time. Counterintuitive and interesting phenomena originate from the additional noise sources that range from the formation of spatiotemporal patterns of species to noise delayed spatial extinction [269–271] and noise-induced phase transitions [268].

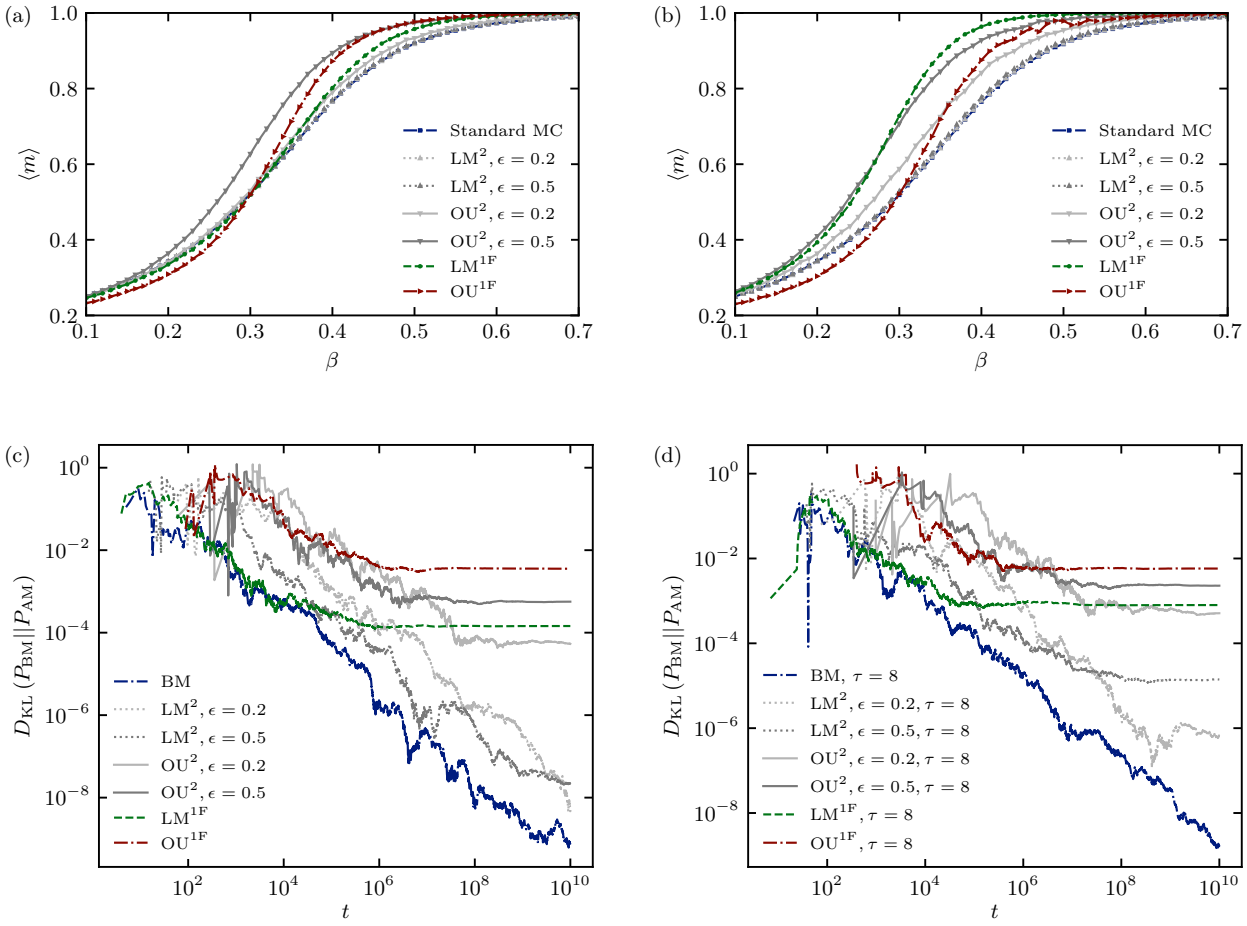

Figure 4.12: Comparison of numerical results for the different models without a refractory mechanism (left column) and with a rectangular PSP shape: (a) Ising model without a refractory mechanism (4×4 lattice). (b) Ising model with a rectangular PSP ($\tau = 8$, 4×4 lattice). (c) Kullback-Leibler divergence without a refractory mechanism. (d) Kullback-Leibler divergence with a rectangular PSP ($\tau = 8$). (a), (b): The absolute magnetization of the Ising model is plotted against the inverse temperature β. The deviations of the different models mirror the observed deviations of the activation function. The results show that small deviations in the activation function can have a large impact on the resulting observables. (c), (d): Illustration of the evolution of the Kullback-Leibler divergence for a Boltzmann machine with three neurons based on their history. In contrast to other plots, the time is not rescaled with respect to the transition probabilities, i.e., the correlation times. This causes a shift of the curves of the sign-dependent processes to larger times. The observed levels of convergence of the Kullback-Leibler divergence of the different models are in concordance to the results of the Ising model. The different levels of convergence for the fitted processes indicates are large dependency of the accuracy of resulting correlation functions on errors in the representation of the activation function and respectively on corresponding weights and biases within the network.

Furthermore, the mathematical structure behind coupled map lattices is very similar to the evolution of the membrane potential of LIF neurons in discrete time with additional non-trivial interacting terms, which are based on the interactions of different neurons (see Eqs. (4.14) and (4.22)). Coupled map lattices [272] describe a dynamical system in discrete space and discrete time, but with a continuous state variable. In future work, we want to analyze whether there exist similar mappings as derived in this work (see Eq. (4.5)) for the dynamics of coupled map lattices. It might also be possible to transfer findings from these research areas onto the here considered dynamics and vice versa. A brief history of excitable map-based neurons and neural networks is given in Ref. [273], for example. Analogies might also be found in more related dynamics like the FitzHugh-Nagumo neuron model [274, 275] with an additional stochastic noise source. A representation of LIF neurons based on a stochastic Fitzhugo-Nagumo neural model is considered in Ref. [276]. In Ref. [277], collective dynamics of a noisy FitzHugh-Nagumo oscillator are studied. Critical phenomena and noise-induced phase transitions on classical random networks are provoked by shot noise in Ref. [278]. We expect that a detailed analysis of all these approaches together with the derived dynamics in this work will result in many interesting phenomena. This holds for simulations as well as for actual implementations on the neuromorhic hardware system.

4.7 Summary and outlook

In this chapter, we introduced the sign-dependent discrete Langevin machine (4.19) and the sign-dependent Ornstein-Uhlenbeck process (4.22). We studied their properties in the context of implementing Boltzmann statistics on the BrainScaleS systems with respect to possible mappings between dynamics with different underlying representations.

The numerical results of different abstractions of a LIF network in Sec. 4.5 demonstrate that the network architecture of the sign-dependent discrete Langevin machine (LM^2) and the OU^2 process is suitable for emulating Boltzmann distributed systems. This applies to both, a discrete two-state system with uncorrelated noise and a continuous system with autocorrelated noise. The numerical results show that an exact implementation of the logistic distribution or at least a correct estimation of errors is crucial to obtain quantitative exact observables.

It remains to be seen whether this statement is also sufficient and valid for nontrivial PSP shapes, as a last step towards LIF sampling. In particular, one has to analyze the impact of marginal deviations to the activation function on observables of larger and more complex systems than the one considered in this work. Moreover, one may ask whether an exact representation of an activation function with a self-interacting term is sufficient to also obtain reliable and accurate results for interacting neurons. In other words: Is it possible to extend findings for a single self-interacting neuron to a general complex interacting system. These questions are postponed to future work. Either way, we expect that the existence of a self-interacting contribution in the OU^2 process helps to better compensate arising non-linearities of the neuromorphic hardware.

Currently, the considered dynamics are restricted to additive Gaussian noise and are built upon sampling with leaky integrate-and-fire neurons as a neuron model. This applies also to the discussed mapping of a process with a discrete state space onto a system with continuous dynamics, as discussed in Sec. 4.1.3. An approach for sampling-based Bayesian spiking inference has been introduced recently in Ref. [75]. The proposed sampling algorithm works without any source of external noise but is driven instead by activities of neighboring sampling spiking neurons. It is interesting whether similar

results can be observed for Bayesian inference based on the introduced sign-dependent dynamics in this work.

So far, the statistical properties of the introduced sign-dependent processes rely on an exact implementation of the underlying equations on a neuromorphic hardware system. Further, the sign-dependent Ornstein-Uhlenbeck process might suffer from large correlation times. These properties limit a broad application of the discovered processes in a wider class of models in biology and further stochastic dynamics at first sight. Furthermore, it is unclear which impact additional non-linearities have on the characteristics of the processes. These non-linearities include, for example, non-trivial PSP shapes or a different kind of noise. Therefore, it is subject to future work to study properties of the introduced dynamics also with respect to a wider class of stochastic processes, see also Sec. 4.6.

In summary, the presented dynamics and the studied abstractions of LIF sampling offer novel tools for bridging the gap between thermodynamics and neuromorphic systems. The potentially more accurate implementation of Boltzmann machines by the dynamics (4.19) and (4.22) represents a further step towards an integrating of deep learning in neuroscience [30, 33, 55].

CHAPTER 5

Towards implementing Langevin dynamics on neuromorphic hardware (non-spiking)

This chapter refers to Ref. [6].

The Hagen mode, see Sec. 2.3.4, allows an implementation of Langevin dynamics on the BrainScaleS-2 chip, different to the ones discussed so far. Designed for a fast computation of matrix-vector multiplications, the non-spiking operation mode can be used for hybrid computations by means of regular read and write operations from a digital computer. For this case, the chip runs without the spiking formalism of neurons. Accordingly, in contrast to the previous chapter, the membrane potential of the neuron can be interpreted to represent a continuous state. We discuss in this chapter limitations and potential properties of a one-to-one implementation of Langevin dynamics in the Hagen mode. In Sec. 5.2, we suggest a path way for such an implementation and point out possible improvements for an easier, more effective implementation. Furthermore, we discuss several hardware-related as well as conceptional restrictions of the BrainScaleS-2 chip. Sec. 5.3 presents another possible implementation of Langevin dynamics. In this implementation, the dynamics takes place in the synaptic weights.

5.1 Langevin dynamics as a set of ordinary differential equations

As a reminder, we start by specifying in more detail what kind of dynamics we aim to implement. The overall goal is to sample field configurations $(\phi_1, \dots, \phi_n)_i$ of the Boltzmann distribution

$$p(\phi_1, \dots, \phi_n) \propto e^{-S(\phi_1, \dots, \phi_n)} \,. \tag{5.1}$$

The action $S(\phi_1, \dots, \phi_n)$ depends on the simulated system and can, in principle, contain arbitrary mathematical relations. Often, it consists of an interaction term and an additional potential. Both terms are often represented by polynomials of linear or higher order. The sampled configurations are used in a second step to numerically compute observables of the type given in Eq. (1.1).

As described in more detail in Sec. 2.1.2, Langevin dynamics is a numerical method to sample such field configurations. In contrast to most of the Markov chain Monte Carlo algorithms, samples are drawn based on a continuous evolution of the fields $(\phi_1, \dots, \phi_n)$. The similarity to the evolution of a membrane potential renders a potential implementation on a neuromorphic hardware system very appealing. As discussed in the introduction, it motivated several chapters of this work.

Restricting ourselves to scalar fields, the sampling process is given by the following set of coupled stochastic differential equations

$$\begin{aligned}
\frac{\mathrm{d}\phi_1(t)}{\mathrm{d}t} &= -\frac{\partial S(\phi_1,\ldots,\phi_n)}{\partial \phi_1} + \eta_1(t)\,, \\
&\vdots \\
\frac{\mathrm{d}\phi_i(t)}{\mathrm{d}t} &= -\frac{\partial S(\phi_1,\ldots,\phi_n)}{\partial \phi_i} + \eta_i(t)\,, \\
&\vdots \\
\frac{\mathrm{d}\phi_n(t)}{\mathrm{d}t} &= -\frac{\partial S(\phi_1,\ldots,\phi_n)}{\partial \phi_n} + \eta_n(t)\,,
\end{aligned} \tag{5.2}$$

where η_i is again Gaussian white noise,

$$\langle \eta_i(t), \eta_j(t') \rangle_\eta = 2\delta(t'-t)\delta(j-i)\,, \quad \langle \eta_i(t) \rangle_\eta = 0 \tag{5.3}$$

and $\langle \cdot \rangle_\eta$ represents the expectation value with respect to the Gaussian distribution

$$\varphi(\eta) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\eta^2}{2}\right)\,. \tag{5.4}$$

5.2 Langevin dynamics in neurons

5.2.1 An abstract model

In the following, we introduce an idealized model of LIF dynamics, see Sec. 2.3.1, in a non-spiking mode and point out conceptional restrictions of this idealized model. In particular, it is assumed that interacting signals can be transferred by means of the membrane potential instead of spikes. The simplification is motivated by other neurmorphic computing devices or newly developed chips designed specifically for fast matrix-vector multiplications. The BrainScaleS chips do not feature this property. In Sec. 5.2.3, current hardware-related restrictions of the chip as well as possible improvements are discussed with respect to an implementation in the Hagen mode.

We start by defining a directed graph $G = (V, E)$ of neurons/vertices V and synapses/edges E. Every neuron i has a membrane potential u_i. The neurons represent LIF neurons that do not spike. They can be approximated by an Ornstein-Uhlenbeck process, cf. Eq. (2.19) in Sec. 2.3.1:

$$\frac{\mathrm{d}u_i(t)}{\mathrm{d}t} = \theta\left(\mu_i(t) - u_i(t)\right) + \sigma_i \eta(t)\,, \tag{5.5}$$

with time-independent parameters θ and σ_i. The additive Gaussian noise contribution can be generated by Poisson noise from multiple other neurons or by an external current, see Sec. 2.3.

The time-dependent mean value $\mu_i(t)$ consists of the following terms:

$$\mu_i(t) = \mu^{\text{leak}} + \mu^{\text{avg. noise}} + \mu_i^{\text{ext.}} + f_i\left(\mu_i^{\text{interaction}}(t) + b_i\right)\,. \tag{5.6}$$

The first two terms are hardware-related. The potential $\mu^{\text{ext.}}$ facilitates a calibration of the membrane potential by means of an external source. The interaction potential $\mu_i^{\text{interaction}}(t)$ and the bias b_i are

defined as the input to an activation function f_i. The bias can be changed during training. The interaction potential $\mu_i^{\text{interaction}}(t)$, defined as

$$\mu_i^{\text{interaction}}(t) = \sum_{E_{ij}} W_{ij} u_j(t)\,, \tag{5.7}$$

captures interactions between neuron i and connected neurons j defined by edges of the considered graph G. They represent the synapses of the neural network. The sum in Eq. (5.7) runs over all edges E_{ij}, accumulating the incoming signals of all neurons j connected with neuron i.

In summary, the dynamics of n neurons V can be described by the following set of coupled non-linear ordinary differential equations:

$$\begin{aligned}
\frac{\mathrm{d}u_1(t)}{\mathrm{d}t} &= \theta\left(\mu^{\text{leak}} + \mu^{\text{avg. noise}} + \mu_1^{\text{ext.}} + f_1\left(\sum_{E_{1j}} W_{1j} u_1(t) + b_1\right) - u_1(t)\right) + \sigma_1 \eta(t)\,, \\
&\vdots \\
\frac{\mathrm{d}u_i(t)}{\mathrm{d}t} &= \theta\left(\mu^{\text{leak}} + \mu^{\text{avg. noise}} + \mu_i^{\text{ext.}} + f_i\left(\sum_{E_{ij}} W_{ij} u_j(t) + b_i\right) - u_i(t)\right) + \sigma_i \eta(t)\,, \\
&\vdots \\
\frac{\mathrm{d}u_n(t)}{\mathrm{d}t} &= \theta\left(\mu^{\text{leak}} + \mu^{\text{avg. noise}} + \mu_n^{\text{ext.}} + f_n\left(\sum_{E_{nj}} W_{nj} u_j(t) + b_n\right) - u_n(t)\right) + \sigma_n \eta(t)\,.
\end{aligned} \tag{5.8}$$

The synaptic weights W_{ij} and the biases b_j represent trainable or configurable parameters. The activation functions f_i and the noise parameter σ_i can be partly tweaked. The remaining parameters are fixed during calibration of the hardware. Hardware effects can be partially imitated by sampling the system parameters around a mean value.

5.2.2 Conceptional restrictions

A comparison of Langevin dynamics (5.2) and the simplified model description of LIF dynamics (5.8) shows that there are certain conceptional restrictions. These concern, in particular, a successful implementation of the drift term $\partial S(\phi_1, \ldots, \phi_n)/\partial\phi_i$.

First, the activation functions f_i might introduce non-linear dependencies that do not coincide with the functional form of the drift term. Second, an implementation of higher-order polynomials in u_i is not possible, or, at least, non-trivial.

We want to illustrate the second restriction based on the simple example of a feed-forward network. For simplicity, the activation function is assumed to be the identity map and the biases are set to zero. In this case, neural input propagates according to

$$u_i = \sum_{E_{ij}} W_{ij} u_j\,, \tag{5.9}$$

where all the u_j's are in the input layer and the u_i's in the output layer. A composition of an additional layer results in

$$u_k = \sum_{E_{ki}} W_{ki} \left(\sum_{E_{ij}} W_{ij} u_j \right) = \sum_{E_{ki}} \sum_{E_{ij}} W_{ki} W_{ij} u_j \,. \tag{5.10}$$

As can be seen, the expression is still only linear in u_j and can interpreted as one linear layer by a redefinition of the weights. This is also the reason why artificial neural networks are built with non-linear activation functions. The introduced non-linearity allows a representation of a wide variety of functional dependencies [279, 280],

$$u_k = f_k \left(\sum_{E_{ki}} W_{ki} f_i \left(\sum_{E_{ij}} W_{ij} u_j \right) \right) . \tag{5.11}$$

However, the resulting composition of several layers is hard to interpret in terms of polynomials, making an implementation of specific higher-order polynomials very difficult.

This restriction also applies to the dynamics of the neuromorphic hardware in Eq. (5.8). We conclude that a simulation of Langevin dynamics based on LIF dynamics is only non-trivial if the drift term of the action contains only zero- or first-order polynomials and if the activation is given by a linear function. The latter restriction also holds for other alternative computing devices optimized for the evaluation of matrix-vector multiplications.

5.2.3 Current hardware restrictions

The dynamics in Eq. (5.8) is not yet realized completely on the BrainScaleS chip, as pointed out in Sec. 5.2.1. In concordance with the human brain, interactions between neurons take place via spikes. Therefore, a transmission of a continuous constant signal between two neurons is not feasible.

A possible workaround is to limit the utilization of the hardware to computing matrix-vector multiplications between weights and input neurons. Especially large systems can benefit from a respective energy-efficient and highly parallelized implementation in the Hagen mode, see also Sec. 2.3.4. A drawback of this approach is that intermediate neuron states need be stored externally. Furthermore, the notion of time changes. The simulation of Eq. (5.8) takes no longer place in real time, but is determined by a discretization of the update scheme in time. Additionally, ones needs to take into account a finite precision for reading and writing signals and weights as well as hardware effects like possible non-linearities in certain regimes and further sources for perturbations.

5.3 Langevin dynamics in synaptic weights

Implementing Langevin dynamics in the weights of the network is a possible approach to resolve the difficulty of a feed-forward multilayer neural network to represent higher-order polynomials based solely on its underlying mathematical operation. The approach is inspired by the functional form of relation (5.10). It is perfectly suited for devices designed for a fast computation and update of feed-forward multilayer neural networks. The approach is another possible way to use the BrainScaleS-2 chip in the non-spiking operation mode.

The weights are considered as dynamical quantities of a feed-forward network. The input of the feed-forward network are the prefactors of the monomials of the action. In this setting, the output can be identified with the action of the system for the current set of weights. Higher-order terms in the weights can be realized by choosing linear activation functions and the same weight for consecutive layers in the network, cf. Eq. (5.10).

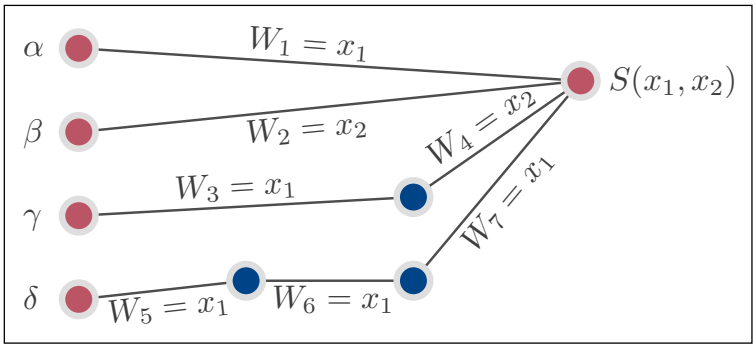

Figure 5.1: Network architecture computing the action (5.12) for implementing Langevin dynamics in the weights of a feed-forward neural network.

As an example, we consider a simple toy model with two state variables x_1 and x_2 and an action of the form,

$$S(x_1, x_2) = \alpha x_1 + \beta x_2 + \gamma x_1 x_2 + \delta x_1^3 \,. \tag{5.12}$$

The resulting feed-forward neural network is shown in Fig. 5.1.All the weights are defined according to the action by the two state variables. The input is given by the prefactors and stays constant. The Hagen mode of the BrainScaleS2-chip is perfectly suited for this kind of task since the computation only involves matrix-vector multiplications.

The drift term $\partial S/\partial x_i$ of Langevin dynamics can be computed by backpropagation. It is given by the sum of all gradients related to the same state variable x_i:

$$\frac{\partial S}{\partial x_i} = \sum_{j, \forall j \, (W_j \leftrightarrow x_i)} \frac{\partial S}{\partial W_j} \,. \tag{5.13}$$

Afterwards, the weights are updated according to an update step of Langevin dynamics in discrete time, cf. Eq. (A.1):

$$W_j' = W_j - \epsilon \frac{\partial S}{\partial x_i} + \sqrt{2\epsilon}\eta_j \,, \quad \forall j \, (W_j \leftrightarrow x_i) \,, \tag{5.14}$$

where ϵ denotes a finite time increment in Langevin time and η_j is Gaussian white noise, cf. Eq. (5.3).

In an alternative setup, the feed-forward network is used to compute the drift term, instead of the action. In this case, a larger network is required since the drift needs to be computed for every state x_i. The presented way for implementing Langevin dynamics can also be applied to any other device allowing the computation of feed-forward networks and featuring backpropagation in the described manner.

Similar to the approach of the previous section, the Langevin time cannot be identified with the real time of evolution of the membrane potential in this kind of implementation. Instead, time is again discrete and the neuromorphic hardware is used as a parallel computing device for matrix-vector multiplications. An advantage towards the sampling based on dynamics in the neurons

is that polynomials of higher-order can be realized. A drawback is the necessity to perform the backpropagation step and to explicitly update the weights.

5.4 Summary

In Sec. 5.2, we analyse an implementation of Langevin dynamics by identifying the membrane potential with the physical state variables of the considered model. Two important limitations are pointed out. The first one is the limitation to a restricted set of interaction terms, namely, zero or first-order terms in the drift. The second limitation refers to the property that on-chip interactions between neurons are only possible via spikes, prohibiting a direct transfer of a continuous, constant signal between neurons. The later limitation entails that a reasonable implementation of Langevin dynamics is only feasible if the dynamics is discrete in the Langevin time. In the Hagen mode, the states are updated based on a computation of the drift term on the neuromorphic device and a subsequent Langevin update on a digital computer. Implementing Langevin dynamics in the weights of the neural network, cf. Sec. 5.3, comes with the benefit of a possible implementation of higher-order polynomials. However, the dynamics still needs to be considered in a discrete Langevin time.

We conclude that a one-to-one identification of the membrane potentials and the physical state variables, i.e., a mapping between the dynamics in Eqs. (5.2) and (5.8), is not feasible given the current properties of the hardware. However, a computation based on the other two methods might still profit from a low energy consumption and a fast evaluation of the action or the drift term in the Hagen mode. A successful integration allows the computation of large physical systems in a parallel computing scheme.

CHAPTER 6

Learning entangled quantum states on a spiking neuromorphic chip

This chapter is based on Ref. [4].

The approximation of quantum states with artificial neural networks has gained a lot of attention during the last years [37–39, 87, 242]. Meanwhile, analog neuromorphic chips show a high energy efficiency in running artificial neural-network architectures for the profit of generative applications [17–19, 30, 33, 250]. This encourages employing such hardware systems as platforms for simulations of quantum systems or quantum state tomography. Here we report on the realization of a prototype using the BrainScaleS hardware. The approximate, probabilistic representation of quantum states is achieved through Bayesian sampling by the spiking neurons, see Sec. 2.3 for a brief introduction to the emulation of Boltzmann distributed statistics. The all-or-nothing nature of spikes represents a blessing in disguise. On the one hand, it does have an apparent drawback by making the computation of gradients – and thus, training – more demanding than in classical deep neural networks [19]. However, it also allows us to use a spiking neuromorphic substrate in the first place, the speed-up of which we harness for efficient Hebbian learning [235].

We show in this chapter that high fidelities can be reached by training the hardware-encoded network to represent maximally entangled quantum states of up to four qubits. Extracted Bell correlations for pure and mixed two-qubit states convey that non-classical features are captured by the analog hardware, demonstrating the feasibility of and an important building block for simulating quantum systems with spiking neuromorphic chips.

The chapter starts with a description of the utilized set-up for an encoding of quantum states on a neuromorphic device in Sec. 6.1. We continue with the encoding of a pure and noisy Bell states as an example for the representation of entangled quantum states with the help of a spiking neural network in Sec. 6.2. In Sec. 6.3, we provide more details about the representation accuracy. After a short discussion of extended network architectures in Sec. 6.4, we conclude with a summary of the results in Sec. 6.5.

6.1 Neuromorphic encoding of quantum states

Here, we encode quantum states by using the hierarchical spiking network architecture, illustrated in Fig. 6.1a, for emulating a restricted Boltzmann machine on the BrainScaleS-2 system, depicted in Fig. 6.1b. The network consists of N visible and M hidden leaky integrate-and-fire neurons arranged in a bipartite graph with a symmetric connectivity matrix. Such a network can be tuned

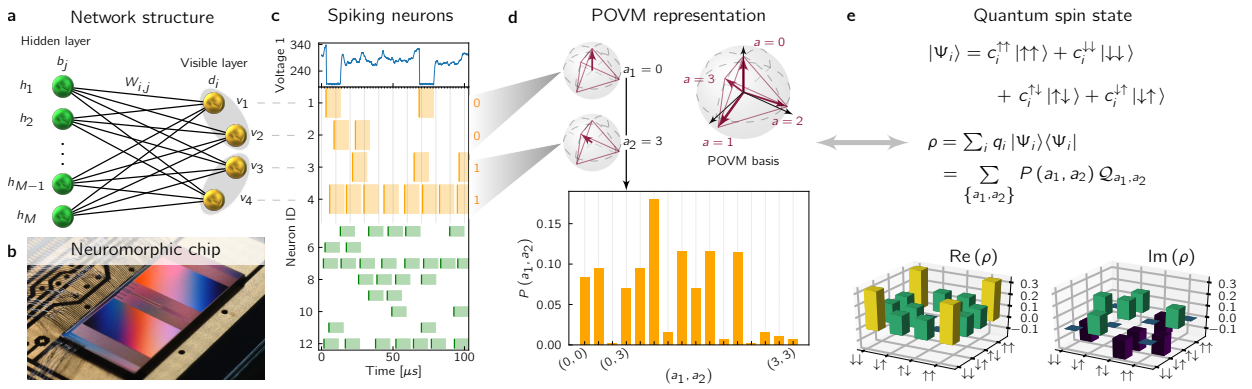

Figure 6.1: **Neuromorphic representation of quantum states.** **a**, Two-layer spiking network architecture with weight parameters $W_{i,j}$ between the visible (orange) and hidden (green) neurons and biases d_i (b_j) for the binary visible (hidden) neurons. **b**, Photograph of the BrainScaleS-2 chip used as a substrate for the experiments in this work [28]. **c**, Dynamical evolution of the spiking network. Upper panel: membrane potential evolution of a single LIF neuron integrating synaptic input. Whenever the potential crosses a threshold a spike is generated and the potential is clamped to prevent immediate refiring (refractory period). Lower panels: Spikes (solid lines) for 4 visible (orange) and 8 hidden (green) neurons with associated $z = 1$ time frames (shaded regions). The network state is observed periodically (gray lines showing only every fifth observation time for visibility reasons). Each observation results in a binary vector corresponding to a sample drawn from the underlying distribution. **d**, The 4-state positive-operator-valued measure (POVM) representation of a qubit state can be encoded by a pair of visible neurons. A combination of N such neuron pairs thus serves to represent an N-qubit system. The frequency of occurrence of neuron configurations drawn from a trained network encodes the POVM probability distribution of a quantum state (lower panel). **e**, Any quantum state can be represented as a density matrix ρ, which can be a statistical mixture of states $|\Psi_i\rangle$. For the example of two qubits shown here, the complex-valued entries of ρ can be reconstructed linearly from the sampled probabilities $P(a_1, a_2)$. For the definition of the operators $\mathcal{Q}_{a_1,a_2}$, see App. D.1.

to approximate the probability of the visible neurons to be in state $\vec{v} = (v_1, \ldots, v_N)$, $v_i \in \{0, 1\}$, as the marginal

$$p\left(\vec{v}; \mathcal{W}\right) = \frac{1}{Z\left(\mathcal{W}\right)} \sum_{\{\vec{h}\}} \exp\left[-E\left(\vec{v}, \vec{h}; \mathcal{W}\right)\right] , \tag{6.1}$$

over all hidden states $\vec{h} = (h_1, \ldots, h_M)$, where $h_j \in \{0, 1\}$, of the joint Boltzmann distribution $p(\vec{v}, \vec{h}; \mathcal{W}) = \exp[-E(\vec{v}, \vec{h}; \mathcal{W})]$ [234, 237]. The network energy $E(\vec{v}, \vec{h}; \mathcal{W}) = -\sum_{i,j} v_i W_{i,j} h_j - \sum_i v_i d_i - \sum_j h_j b_j$ depends on the set of network parameters $\mathcal{W} = (W, \vec{b}, \vec{d})$ including the weights $W_{i,j}$ and biases b_j and d_i. The partition sum $Z(\mathcal{W}) = \sum_{\{\vec{v},\vec{h}\}} p(\vec{v}, \vec{h}; \mathcal{W})$ ensures normalization. Note that the network represents a particular implementation of the Boltzmann machine discussed in Sec. 2.3.2 characterized by the respective constraints on the network connectivity and a distinction of visible and hidden neurons. Statistical samples are collected in a histogram by observing the state of each neuron at regular time intervals, as visualized in Fig. 6.1c and Fig. 6.1d; see also Sec. 2.3.3 for more details.

A pure quantum state is described by a vector in Hilbert space and can be represented by a hermitian density matrix with complex entries. Density matrices can also encode mixed states and thus

account for a possible coupling to an environment, which is relevant for a realistic description of experiments. Fig. 6.1e shows an example of a density matrix for a system of two spin-1/2 degrees of freedom (qubits) corresponding to a Hilbert-space dimension $d = 4$. The corresponding probability distribution which we encode in our network is obtained from a so-called tomographically complete measurement [86]. Such a measurement has d^2 possible outcomes. Mathematically, these outcomes are represented by a set of operators $\{M_{\vec{a}}\}_{\vec{a}}$, forming a so-called positive-operator-valued measure (POVM). A detailed description of the encoding of Bell states by means of POVM probability distribution and the subsequent representation in a spiking neural network is provided in App. D. Here, we continue with a summary of the method and of the utilized training algorithm.

Based on our approach, the density matrix can be reconstructed uniquely as $\rho = \sum_{\{\vec{a}\}} P(\vec{a})\mathcal{Q}_{\vec{a}}$ from probabilities $P(\vec{a}) = \mathrm{Tr}\,[\rho M_{\vec{a}}]$ for obtaining outcome $\vec{a}$ according to Born's rule. The operators $\mathcal{Q}_{\vec{a}}$ are given by $\mathcal{Q}_{\vec{a}} = \sum_{\{\vec{a}'\}} T^{-1}_{\vec{a},\vec{a}'} M_{\vec{a}'}$, with $T_{\vec{a},\vec{a}'} = \mathrm{Tr}\,[M_{\vec{a}} M_{\vec{a}'}]$ [87]. In our two-qubit example (Fig. 6.1d) we chose $M_{\vec{a}} = M_{a_1} \otimes M_{a_2}$, where M_{a_i} $(a_i = 0, \ldots, 3)$ are projection operators onto the single-qubit states represented as the four corners of a tetrahedron on the Bloch sphere. As each a_i can take four different values, the encoding of the probabilities $P(\vec{a})$ by a spiking network is realized by representing each qubit by a pair of binary neurons in the visible layer, defined over the neuron states $\vec{v}$ (cf. gray shadings in Fig. 6.1a). This results in the distribution $p^*(\vec{v})$ over the visible neurons, see App. D.1 for more details.

To approximate $p^*(\vec{v})$ through spike-based sampling, the parameters of the spiking network were adjusted in an iterative training procedure. We used the Kullback-Leibler divergence

$$D_{\mathrm{KL}}\left(p^* \| p\right) = \sum_{\{\vec{v}\}} p^*(\vec{v}) \ln\left[\frac{p^*(\vec{v})}{p(\vec{v};\mathcal{W})}\right] \tag{6.2}$$

to measure the quality of the sampled marginal $p(\vec{v};\mathcal{W})$. In each training epoch, the synaptic weights were updated along the gradient of the D_{KL}, see also App. D.2:

$$\Delta W_{i,j} \propto \langle v_i h_j \rangle_{\mathrm{target}} - \langle v_i h_j \rangle_{\mathrm{model}} \,. \tag{6.3}$$

Pairwise correlations $\langle v_i h_j \rangle_{\mathrm{model}}$ in the network were directly estimated from the sampled distribution $p(\vec{v}, \vec{h}; \mathcal{W})$. Target correlations were also obtained from the sampled distribution by renormalization to the target marginal distribution:

$$\langle v_i h_j \rangle_{\mathrm{target}} = \left\langle \frac{p^*(\vec{v})}{p(\vec{v};\mathcal{W})} v_i h_j \right\rangle_{p(\vec{v},\vec{h};\mathcal{W})} . \tag{6.4}$$

A similar scheme was used for the neuronal biases b_j and d_i. This otherwise prohibitively compute-intensive method was made possible by the accelerated hardware dynamics and allows a much better approximation of the D_{KL} gradient than the more conventional contrastive divergence update scheme [235]. Moreover, it does not rely on layer-wise conditional independence, allowing the exploration of network topologies other than bipartite graphs.

6.2 Encoding an entangled Bell state

To demonstrate that a spiking neural network can learn to represent entangled quantum states we focus on a maximally entangled two-qubit state, the Bell state $|\Psi^+\rangle = \left(|\uparrow\uparrow\rangle + |\downarrow\downarrow\rangle\right)/\sqrt{2}$. This

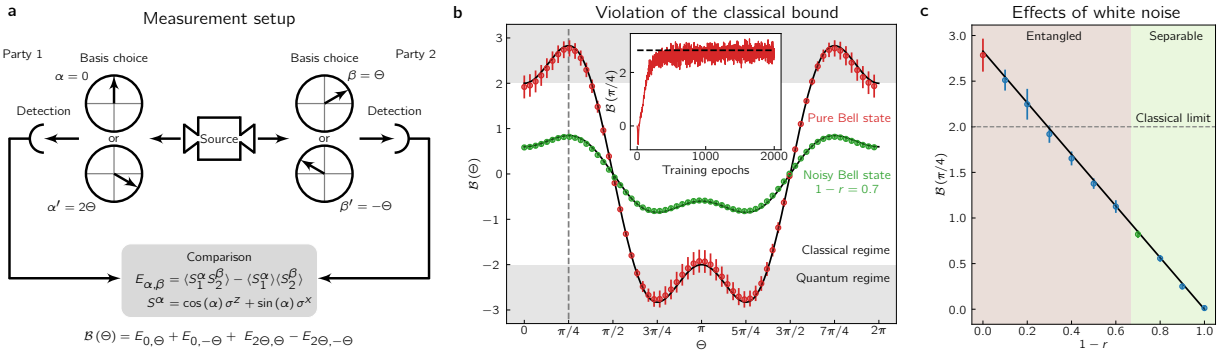

Figure 6.2: **Encoding Bell states and Werner states.** **a**, Illustration of a typical Bell-test scenario. Two correlated qubits emerging from a source are distributed between two parties. Each of the parties is allowed to choose between two different measurements each characterized by a single common angle Θ. The measurement outcomes indicate genuine quantum correlations if the combination $\mathcal{B}(\Theta)$ of the correlations violates the inequality $|\mathcal{B}(\Theta)| \leq 2$ obeyed by classical states. **b**, Observable $\mathcal{B}(\Theta)$ evaluated on the learned encoding of the Bell state $\rho_\mathrm{B} = |\Psi^+\rangle\langle\Psi^+|$ on the neuromorphic hardware, with $M = 20$ hidden neurons. Red symbols depict the observable for different angles Θ, averaged over the last 200 training epochs, where errorbars here and in the following denote the standard deviation. Note that these data points have been obtained from the same trained network and the same set of neuron states sampled from it by evaluating the observable for different angles Θ on this sample set. Werner states $\rho_\mathrm{W} = r\rho_\mathrm{B} + (1 - r)\,\mathbb{1}/4$ are obtained by adding white noise to the pure Bell state. Green points correspond to $r = 0.3$. In both cases, the data capture the exact values (black lines) well, including the violation of the classical bound in the pure case $r = 1$. The inset shows the evolution of the Bell-correlation witness $\mathcal{B}(\Theta = \pi/4)$ during training (red line) and the convergence towards the expected value (black dashed line). **c**, Bell-correlation witness $\mathcal{B}(\Theta = \pi/4)$ for a Werner state as a function of the noise strength $1 - r$. The exact solution (black line) is captured well for both entangled and separable states.

state is a prototypical example exhibiting quantum mechanical correlations [281, 282]. We trained a network of four visible and 20 hidden neurons to encode the POVM probability distribution corresponding to $\rho_\mathrm{B} = |\Psi^+\rangle\langle\Psi^+|$. For calculating the weight updates in each epoch of the training procedure, as well as for evaluating expectation values, we drew 125000 samples of neuron states. This number is sufficient for the saturation of the D_KL as can be seen in Fig. 6.3b and was used for all experiments, if not specified otherwise.

To characterize the learned quantum state, we used the observable $\mathcal{B}(\Theta)$, which can signal genuine quantum correlations and is experimentally accessible via measurements as illustrated and defined in Fig. 6.2a: The two qubits are distributed to two parties who independently perform one of two possible measurements on their respective qubit. We choose the standard parametrization of the different measurements by a single angle Θ. For a Bell state this procedure yields correlations violating the inequality $|\mathcal{B}(\Theta)| \leq 2$, which is obeyed by classical systems [282]. At $\Theta = \pi/4$ this inequality is maximally violated for the Bell state ρ_B and thus yields an experimentally accessible witness for Bell correlations [281, 283].

The correlations encoded by the trained spiking network clearly exceed the classicality bound $|\mathcal{B}(\Theta)| = 2$ (red points in Fig. 6.2b) and are in agreement with their exact Θ-dependence (black line). The inset shows how the Bell correlation witness $\mathcal{B}(\Theta = \pi/4)$ develops during the training, converging after less than 1000 iterations.

To illustrate the generality of our neuromorphic encoding scheme we consider mixed quantum states by adding white noise to the pure Bell state resulting in the Werner state $\rho_\mathrm{W} = r\rho_\mathrm{B} + (1-r)\,\mathbb{1}/4$ with noise strength $0 \leq 1-r \leq 1$ [284]. Increasing the noise reduces $|\mathcal{B}(\Theta)|$ and eventually confines it within the classical regime (cf. green data in Fig. 6.2b). For $1-r > 1/\sqrt{2}$ the Bell correlation witness fails to detect entanglement, and for $1-r > 2/3$ the state becomes separable (unentangled). The resulting mixed states are faithfully represented by our system for any value of r as shown in Fig. 6.2c. The fluctuations in the experimental data decrease with increasing noise contribution, allowing a more accurate learning of mixed states. This counterintuitive effect is due to additional noise leading to an increase in entropy, which is synonymous with sampling from more uniform distributions. These, in turn, are realized by weaker weights, thus decreasing the influence of imperfect synaptic interactions in the neuromorphic substrate.

6.3 Learning performance

We analyzed in detail the convergence of the learning algorithm using the classical Kullback-Leibler divergence D_KL as defined in (6.2). In addition, we use the quantum fidelity

$$\mathcal{F}(\rho_\mathrm{B}, \rho_\mathrm{N}) = \mathrm{Tr}\left[\sqrt{\sqrt{\rho_\mathrm{B}}\rho_\mathrm{N}\sqrt{\rho_\mathrm{B}}}\right], \tag{6.5}$$

to quantify the distance between the target state ρ_B and the network-encoded state ρ_N, which, for pure states, reduces to the state overlap. As shown in Fig. 6.3a, the learning converges after 1000 training epochs. Increasing the number of hidden neurons we find that the fidelity reaches $\approx 98\%$ (correspondingly $D_\mathrm{KL} \lesssim 10^{-2}$) for $M \gtrsim 20$ hidden neurons. The limited reachable fidelity is a result of many different factors of the physical implementation of the spiking neural network on the BrainScaleS-2 platform. The synaptic connections are implemented with 6-bit resolution, limiting the achievable precision of approximating the probability distribution. Also, uncontrolled environmental changes such as temperature variations or host-to-system effects influence the performance of the

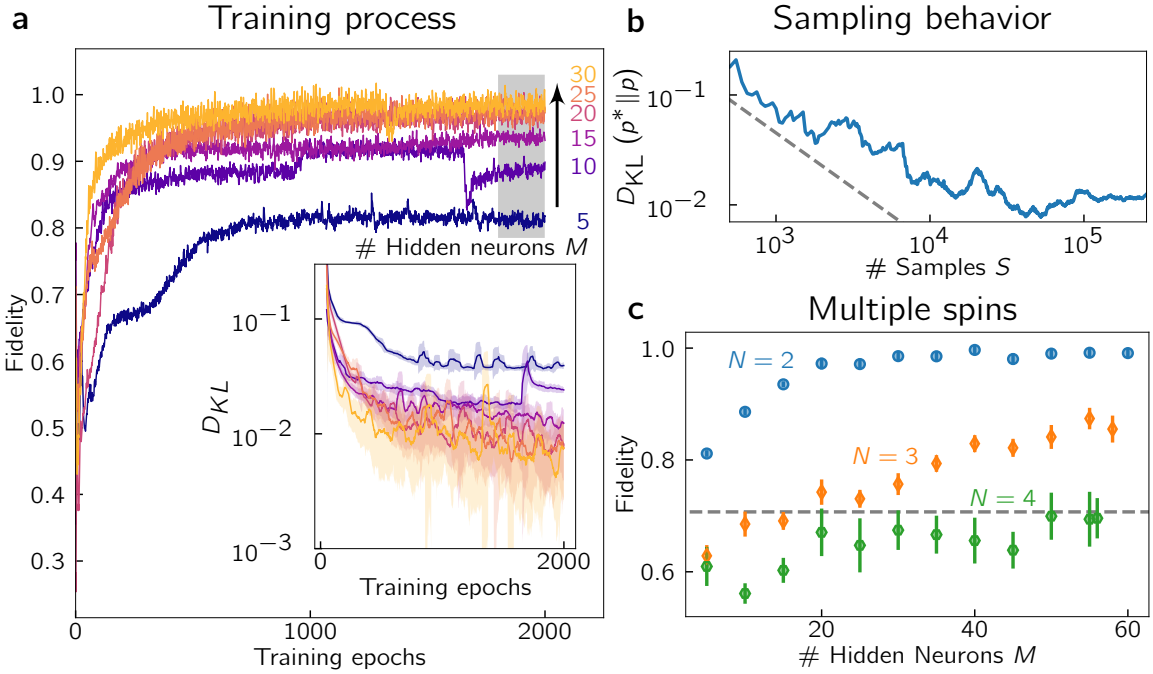

Figure 6.3: **Training performance.** **a**, Dynamics of the learning procedure for the pure Bell state ρ_B. The quality of the network-encoded state is measured by the quantum fidelity, (6.5) (main frame), and by the Kullback-Leibler divergence, (6.2) (inset), for different numbers of hidden neurons. For better visibility, the running average over 50 epochs is shown in the inset as solid lines, with the shaded areas indicating the corresponding standard deviation. **b**, Kullback-Leibler divergence in a fixed trained network with $M = 20$ hidden neurons as a function of the number of samples drawn. The dashed line shows the expected trend for exact sampling from the target distribution. **c**, Quantum fidelity as a function of the number of hidden neurons for GHZ states $|\Psi\rangle = (|\uparrow\rangle^{\otimes N} + |\downarrow\rangle^{\otimes N})/\sqrt{2}$ with $N = 2$ (Bell state), 3, and 4 qubits. We show the averages over 200 training epochs after convergence (gray shaded area in **a**). The dashed line shows the bound for genuine N-partite entanglement.

hardware. This manifests itself in the jumps of fidelity occurring during learning, as well as in strong noise in the fidelity after the learning process has saturated, as can be seen in Fig. 6.3a. These instabilities exceed the anticipated noise level due to finite sample statistics used for evaluating observables and calculating gradients in each epoch. These factors degrade the correspondence between the model assumption underlying the employed learning rule and the actual dynamics of the hardware. Many of the issues mentioned above can be resolved in future hardware generations.

To ensure that the learning performance is not limited by finite sample statistics, we evaluated the Kullback-Leibler divergence as a function of the number of samples in a trained network with fixed network parameters. Fig. 6.3b shows the expected convergence towards a minimum value determined by the quality with which the spiking network approximates the POVM distribution. Typically, for $>10^5$ samples the statistical error is negligible compared to the errors due to hardware noise and limited representational power of the network, causing the saturation of the DKL observed in Figure 6.3b. This justifies our choice of training with 125000 samples per epoch.

Having demonstrated high-fidelity emulation of two-qubit entangled states, we investigated whether states of multiple qubits can also be encoded by our spiking sampling network. Fig. 6.3c shows the fidelity achieved in learning Greenberger-Horne-Zeilinger (GHZ) states [285], i. e. N-qubit generalizations of a Bell state, as a function of the number of hidden neurons M. The underlying probability distribution covers a larger state space of the visible neurons, requiring us to increase the number of samples to 225000 to reach convergence in the D_{KL}. In all cases the fidelity of the learned state to the perfect GHZ state increases with M, reaching values of close to 90% and about 70% for three and four qubits, respectively. As layered network architectures are known to require a large number of neurons for representing GHZ states [87], we assume that larger chip sizes will allow to increase these values further. Note that a GHZ-state fidelity above $\mathcal{F} = 1/\sqrt{2} \approx 70\%$ means that the state exhibits genuine N-partite entanglement (cf. dashed line in Fig. 6.3c) [286].

6.4 Deep and partially restricted networks

Our flexible learning scheme allows the training of network architectures beyond simple bipartite graphs. To explore network architectures with potentially larger representational power we added connections between the visible neurons, resulting in a more densely connected network. Figure 6.4a shows that a Bell state can be encoded successfully with this architecture, reaching similar fidelities as the two-layer fully restricted spiking network. We also explored deeper network architectures by adding an additional hidden layer, see Fig. 6.4b. Again, the Bell state was learned successfully reaching similar fidelities as in the bipartite case. We note that the learning performance is not monotonic at small M for $M_2 = 10$ neurons in the second hidden layer. This is expected, since the intermediate layer constitutes an information bottleneck towards the visible layer, which makes learning more difficult. Therefore, the greater representational power offered by additional depth [287] does not necessarily translate into a higher fidelity for $M < M_2$. The overall non-monotonic dependence of the fidelity on the number of hidden neurons is caused by hardware noise leading to fluctuating training performance.

The fact that the learning performance does not improve when using different architectures indicates that the reachable fidelity is currently limited by technical imperfections rather than the representational power of the ansatz. Larger-scale systems may be able to exploit the greater representational power of these deeper and more complex architectures.

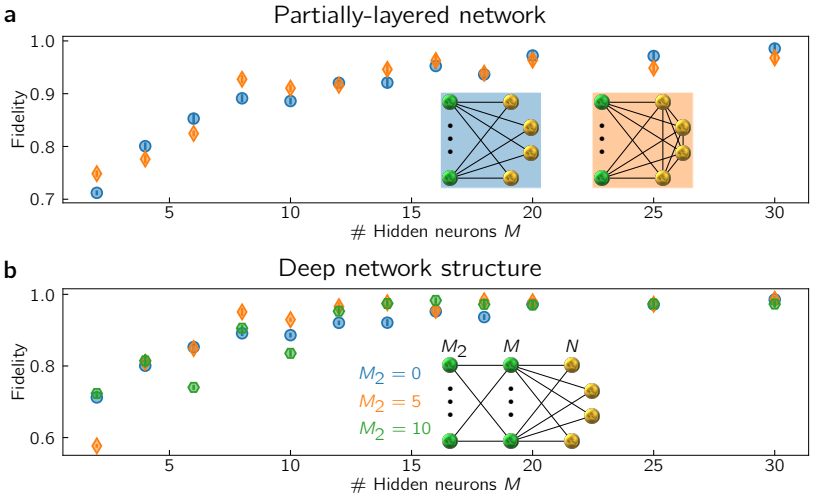

Figure 6.4: **Extending the network architecture. a**, Fidelity between network-encoded and perfect Bell state, Eq. (6.5), as a function of the number of hidden neurons for strictly layered network (blue) and an architecture with additional connections between the visible neurons (orange). **b**, Quantum fidelity for states encoded in a deep network architecture with a second hidden layer containing 5 (orange) and 10 (green) neurons compared to the restricted two-layer network (blue).

6.5 Summary

We have shown that a spiking neural network implemented on a neuromorphic chip can approximate entangled quantum states of few particles with high quantum fidelity. In particular non-classical Bell correlations can be encoded faithfully, demonstrating that intrinsic quantum features can be captured by a classical spiking network.

The fidelities and system sizes achieved in this first study on neuromorphic quantum state encoding should be regarded as a proof of principle. The experienced restrictions are only technical in nature and can be improved in future generations of spiking neuromorphic devices. Specifically for the BrainScaleS-2 system, both the hardware and its surrounding software framework are in an ongoing maturation process. The size and fidelity of the approximated quantum states can be significantly improved upon by optimizing the usage of hardware real-estate, the signal-to-noise ratio of the analog circuitry and the calibration of the chip. Judging from the current pace of progress in neuromorphic engineering, significantly larger systems, both digital and analog, can be expected to become available in the near future [18].

Furthermore, runtime improvements are anticipated, as the current bottleneck is the calculation of the weight updates of the network parameters, which is done "offline" on a conventional computer, and only the sampling itself is performed on the chip, see also App. D.4. Using the on-chip plasticity processor to update synaptic weights has the potential of drastically reducing the training time by removing the cumbersome chip-host loop [252].

One key advantage of this neuromorphic system as compared with simulated generative models is that scaling to larger network sizes does not increase the time needed to collect a desired number of samples. We illustrate this property by comparing the sampling time on the neuromorphic chip with a CPU implementation, see App. D.5, showing a gain through neuromorphic sampling already at moderate system sizes. Given the efficient learnability [288] and representability of important

classes of quantum states [38, 242, 289], and the availability of sampling schemes for neuromorphic devices [1, 290], we thus expect favorable scaling properties for our approach, see also App. D.3. Thus our work opens up a path towards applications of neuromorphic hardware in quantum many-body physics.

CHAPTER 7

Towards sampling complex actions

This chapter is based on Ref. [5].

Path integrals with complex actions are encountered in many physical systems, including spin- or mass-imbalanced atomic gases, graphene, as well as quantum chromo-dynamics at finite density and the non-equilibrium evolution of quantum systems [79, 88, 92, 94, 99, 102, 103, 106, 109, 111], see also Chapter 1. Many computational approaches have been developed for tackling the sign problem emerging for complex actions [97, 115, 116, 118]. Among these, complex Langevin dynamics has the appeal of general applicability [15, 117]. One of its key challenges is the potential convergence of the dynamics to unphysical fixed points. The statistical sampling process at such a fixed point is not based on the physical action and hence leads to wrong predictions [80, 119–121, 225]. Moreover, its unphysical nature is hard to detect due to the implicit nature of the process.

In the present chapter we set up a general approach based on a Markov chain Monte Carlo scheme in an extended state space. In this approach we derive an explicit real sampling process for generalized complex Langevin dynamics. Subject to a set of constraints, this sampling process is the physical one. These constraints originate from the detailed-balance equations satisfied by the Monte Carlo scheme. This allows us to re-derive complex Langevin dynamics from a new perspective and establishes a framework for the explicit construction of new sampling schemes for complex actions. The resulting framework rests on a reformulation of the path integral (1.1) as a one-dimensional[1] stochastic integral in the complex plane. The approach is sketched together with an outline of the chapter in Sec. 7.1.

7.1 Summary of main results

7.1.1 Motivation

The standard approach in formulating complex Langevin dynamics consists of inserting the complex action into the Langevin equation and proving the validity of the solutions by a comparison with the associated Fokker-Planck equations, cf. Eq. (2.7) in Sec. 2.1 for the case of real Langevin. Here, we take a different route and derive complex Langevin dynamics from first principles.

[1] One dimensional here refers to the important fact that, even if the integration variable ϕ is allowed to become complex, one still integrates over ϕ only. This in contrast to, e.g., a coherent-state path integral $\sim \int \mathrm{d}\phi\, \mathrm{d}\phi^\star \exp(-S)$, which, in that sense, is two dimensional.

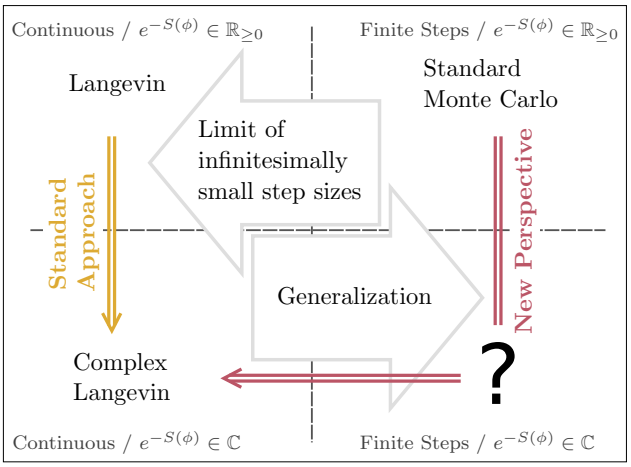

Figure 7.1: Comparison of the standard approach to deriving complex Langevin dynamics and of the perspective provided in this work. By taking the limit of infinitesimally small step sizes, the Markov chain turns into a continuous evolution in state space. This kind of dynamics corresponds to the left-hand side of the graphics. We pursue the goal to generalize complex Langevin dynamics as a Monte Carlo algorithm for complex measures that also works with finite step sizes in configuration space.

While complex Langevin dynamics is thus identified as a valid means for evaluating Eq. (1.1) for complex-valued actions S, it nevertheless still suffers from the numerical problem of runaway processes as well as convergence to unphysical solutions [97, 116, 217, 220, 222–225, 227, 228, 291, 292]. Moreover, the continuous evolution of Eq. (2.10) cannot be straightforwardly applied to models of discrete-valued fields ϕ such as of spin systems.

With the framework introduced in the following, we aim to pave the way for two long-term goals:

- A generalization of complex Langevin dynamics that allows developing numerically more stable sampling algorithms.
- A numerical computation of expectation values for discrete systems with a sign problem which does not rely on reweighting but entails sampling in an extended state space.

In Fig. 7.1, we relate known techniques and their derivations with the chosen path of our approach to achieve these two goals.

7.1.2 Key insights

Our central task is to evaluate expectation values of observables $\mathcal{O}(\phi)$ with respect to the complex distribution $\rho(\phi)$ defined in Eq. (2.3), depending, for the first, on a real-valued field ϕ,

$$\langle \mathcal{O}(\phi) \rangle_\rho = \int_a^b \mathrm{d}\phi \, \mathcal{O}(\phi) \rho(\phi) \,, \tag{7.1}$$

with integral boundaries a and b.

We substitute the field variable ϕ by

$$\phi = \phi(\phi_x) = \phi_x + i\phi_y \,, \quad \mathrm{d}\phi = \mathrm{d}\phi_x \,, \tag{7.2}$$

where the integration variable is now the real field ϕ_x, and ϕ_y is, for the moment, just a constant. The integral turns into

$$\langle \mathcal{O}(\phi) \rangle_\rho = \int_{a-i\phi_y}^{b-i\phi_y} \mathrm{d}\phi_x \, \mathcal{O}(\phi_x + i\phi_y)\rho(\phi_x + i\phi_y) \,. \tag{7.3}$$

If the integral is invariant with respect to a shift of its boundaries by $i\phi_y$, we can reset the integral bounds back to a and b, such that the integral reads

$$\langle \mathcal{O}(\phi) \rangle_\rho = \int_{a}^{b} \mathrm{d}\phi_x \, \mathcal{O}(\phi_x + i\phi_y)\rho(\phi_x + i\phi_y) \tag{7.4}$$

and, as a result, becomes independent of ϕ_y. We will later identify ϕ_y with the imaginary part of the field in the complex Langevin evolution. Under these conditions, we can express the expectation value as the mean over multiple, arbitrary values of ϕ_y,

$$\langle \mathcal{O}(\phi) \rangle_\rho = \frac{1}{N} \sum_{i=1}^{N} \int_{a}^{b} \mathrm{d}\phi_x \, \mathcal{O}(\phi_x + i\phi_{y;i})\rho(\phi_x + i\phi_{y;i}) \,, \tag{7.5}$$

and it will be sufficient to assume that the above invariance holds for the range of values of ϕ_y appearing in this sum. We furthermore assume that there exists a numerical method for sampling ϕ_x from ρ. In this case, we can express the integral on the right-hand side as the mean value

$$\langle \mathcal{O}(\phi) \rangle_\rho = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{M} \sum_{j=1}^{M} \mathcal{O}(\phi_{x;ij} + i\phi_{y;i}) \,. \tag{7.6}$$

Note that ϕ_x depends on both i and j, meaning that one draws the samples $\{\phi_{x;ij}\}$ for a fixed $\phi_{y;i}$.

In the last step, we argue that we can mix $\phi_{x;ij}$'s belonging to different values of $\phi_{y;i}$ as long as changes in $\phi_{y;i}$ do not introduce further correlations between the updated ϕ_y and the sampled ϕ_x, as will be discussed in more detail below. This can be implemented by allowing only infinitesimally small changes in ϕ_y, independent of the sampling probability for ϕ_x.

Under the above condition, we can keep the sampling index i as the only one in the sum over, still, MN samples, implying that we no longer consider separate evolutions for a fixed $\phi_{y;i}$, but smoothly mix the respective evolutions in ϕ_x. So the index i counts the combined update step of both ϕ_x and ϕ_y, performed, e.g., in complex Langevin dynamics,

$$\langle \mathcal{O}(\phi) \rangle_\rho = \frac{1}{MN} \sum_{i=1}^{MN} \mathcal{O}(\phi_{x;i} + i\phi_{y;i}) \,. \tag{7.7}$$

This expression is eventually to be understood as a numerical expectation value determined from samples (ϕ_x, ϕ_y) which a stochastic process generated according to the complex distribution $\rho(\phi_x + i\phi_y)$, i.e.:

$$\langle \mathcal{O}(\phi_x + i\phi_y) \rangle_\rho = \frac{1}{MN} \sum_{i=1}^{MN} \mathcal{O}(\phi_{x;i} + i\phi_{y;i}) \,. \tag{7.8}$$

We conclude that we can reinterpret the computation of the integral in Eq. (7.1) as that of the expectation value of a combined process in ϕ_x and ϕ_y, where it needs to be guaranteed that stochastic changes according to some transition probability take place only in the ϕ_x direction. This ensures

that we independently compute a mean value with respect to the imaginary part of the field. As a result, the expectation value in the extended state space of complex ϕ still reflects the degrees of freedom of the integral (7.1).

In summary, we can write

$$\langle \mathcal{O}(\phi) \rangle_\rho = \langle \mathcal{O}(\phi_x + i\phi_y) \rangle_\rho \,, \tag{7.9}$$

as long as the invariance of the integral (7.1) under imaginary shifts of the boundaries holds and the underlying stochastic process has a vanishing variance along the ϕ_y direction. Note that the expectation value on the right-hand side is considered to be computed by means of a Markov chain Monte Carlo algorithm with respect to $\rho(\phi_x + i\phi_y)$, but constrained by the conditions provided above.

These conditions imply that the eventually obtained higher-dimensional probability distribution, denoted as $P(\phi_x, \phi_y)$, is different from $\rho(\phi_x + i\phi_y)$. This point of view is in strong contrast to standard considerations of complex Langevin dynamics where a Fokker-Planck equation in both ϕ_x and ϕ_y and, therefore, an expectation value with respect to a distribution $P(\phi_x, \phi_y)$ is analysed [97, 122, 219]. Furthermore, the restriction to an infinitesimal step size in the ϕ_y direction underscores findings of a higher numerical stability for vanishing imaginary noise [216, 219]. It also corroborates the finding that a distribution $P(\phi_x, \phi_y)$ which decays sufficiently fast in the imaginary direction ensures correct convergence [217, 223].

Due to the restrictions discussed above, we can compute the expectation value on the right-hand side of Eq. (7.9) only by an extrapolation to a vanishing change in ϕ_y. In practice, this can be achieved by performing multiple simulations with small step sizes and an extrapolation of the resulting observables.

At first sight, evaluating an expectation value with respect to the complex distribution $\rho(\phi_x + i\phi_y)$ over complex fields instead of $\rho(\phi)$ over real ϕ does not improve a numerical sampling due to difficulties in defining real-valued transition probabilities. However, the imaginary part ϕ_y of the field introduces an additional degree of freedom. In contrast to that of the real part ϕ_x, the dynamics of ϕ_y is only constraint by the proposed restriction to an infinitesimally small update. We point out once more that this is the reason for a different distribution $P(\phi_x, \phi_y)$ observed after sampling.

Hence, it is important to understand that there is a difference between sampling from a higher-dimensional probability distribution defined in ϕ_x and ϕ_y and a sampling of the complex distribution $\rho(\phi_x + i\phi_y)$ subject to the above restrictions. In this work, we refer only to the latter case and discuss how to set up a Markov chain Monte Carlo algorithm with respect to this numerical sampling procedure.

7.1.3 Key results

We will show that, in the case of complex Langevin dynamics, the introduced additional degree of freedom can be used to render the transition probabilities of a corresponding Markov chain Monte Carlo algorithm real and positive. As a result numerical sampling from the complex distribution $\rho(\phi_x + i\phi_y)$ becomes possible.

In the following sections,

- we set the ideas laid out in Sec. 7.1.2 on firm grounds and compare a numerical sampling of the expectation value in Eq. (7.5) with other algorithms (Restricted Boltzmann machine and

Hamiltonian Monte Carlo algorithm) where the underlying dynamics also takes place in an extended higher-dimensional state space, cf. Sec. 7.2.3;

- we give a reminder on general aspects of Monte Carlo sampling in higher dimensions, cf. Sec. 7.2;
- we introduce *Substitution Sampling* as a possible approach to constructing transition probabilities of a Markov chain Monte Carlo algorithm from first principles, which allows sampling from the equilibrium distribution subject to the constraints given above, cf. Sec. 7.3;
- we show that complex Langevin dynamics satisfies these first principles, cf. Sec. 7.3.2;
- we provide numerical evidence in support of our approach by use of other algorithms that are built on the same first principles as complex Langevin dynamics, for the cases of a complex action and a real action, cf. Secs 7.4 and 7.5 for the derived algorithms and Sec. 7.6 for numerical results.

Furthermore, a systematic approach for the derivation of substitution sampling algorithms is presented in Chapter 8. The algorithms in Sec. 7.4 are derived based on this approach.

Our work provides a framework for deriving Markov chain Monte Carlo algorithms that are, in principle, suitable to sample from a distribution given in terms of a complex action. We expect the framework to be useful for developing new algorithms for theories with a sign problem.

7.2 Markov chain Monte Carlo sampling in auxiliary dimensions

In this section, we introduce a formal framework for developing Monte Carlo sampling algorithms on state spaces including additional auxiliary dimensions. By this we mean algorithms that work in a higher-dimensional representation space while sampling from a lower dimensional probability distribution. In Sec. 7.2.3, we generalize and embed the perspective on computing observables given in the previous section into this framework.

7.2.1 Extended state space

Examples of algorithms, in which the dimension of the state space is extended by additional auxiliary dimensions include the Hamiltonian Monte Carlo algorithm (HMC) [293], introducing momenta for each state, or the restricted Boltzmann machine (RBM) [294], with a distinction between visible and hidden layers of neurons. Both algorithms are recapitulated in App. B. In the case of the Hamiltonian Monte Carlo algorithm, the extra dimensions lead to a faster exploration of the original state space. For the restricted Boltzmann machine, the introduced hidden layers are essential for the representation of a larger class of in general non-Gaussian probability distributions.

Complex Langevin dynamics can also be attributed to this class of algorithms. The state space is complexified and the imaginary part represents an auxiliary variable, cf. Sec. 7.1.2.

Inspired by the RBM, we introduce auxiliary dimensions by distinguishing *visible* state variables v and *hidden* state variables w. For RBMs, the visible variables are given by the neuron states in the visible layer and the hidden variables by the ones in the hidden layer. Fig. 7.2 depicts the distinction of visible (red) and hidden (blue) variables for the different algorithms. In the case of the

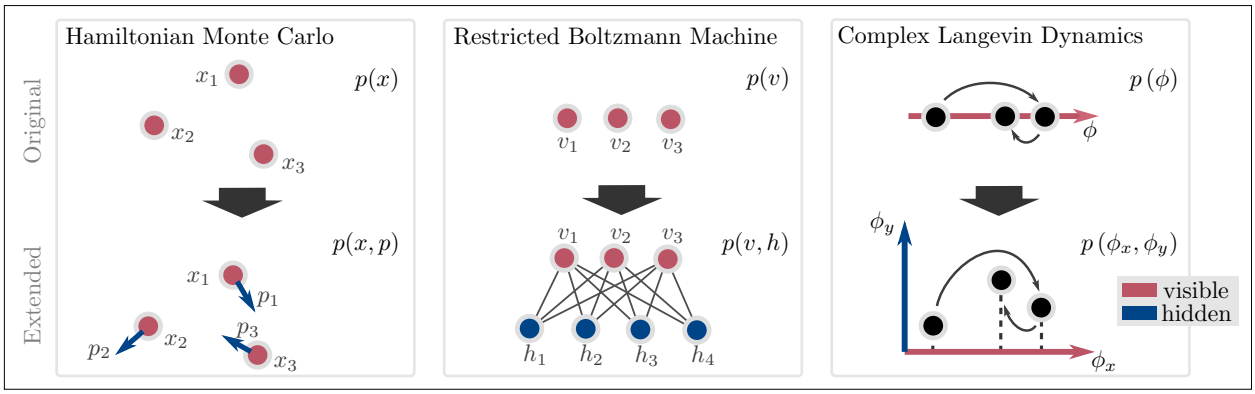

Figure 7.2: Comparison of different algorithms that all make use of the introduction of additional hidden variables / auxiliary dimensions. The respective Markov chain is realized in a new set of visible and hidden variables (v, w). The target distribution of the different algorithms is indicated in the upper row. In the lower row, the newly introduced hidden variables are marked in blue and the visible variables in red. The probability distributions are functions of this new set of variables. In the case of complex Langevin dynamics, the field ϕ is promoted to a complex field where the visible variable is given by the real part ϕ_x and the hidden variable by the imaginary part ϕ_y. Accordingly, the field ϕ is parametrized by $\phi_x + i\phi_y$. Observables in the original set of variables can be obtained for the Hamiltonian Monte Carlo algorithm and the restricted Boltzmann machine by marginalizing the higher-dimensional distributions. This is different for complex Langevin dynamics, where observables are expressed according to Eq. (7.16) in terms of the hidden and visible variables. Details on the algorithms can be found in different sections of this work and in the appendix.

HMC algorithm the variables x are considered as visible and the momenta p as hidden variables. For complex Langevin dynamics, the higher-dimensional representation is given by the complex field. The real part of the field is identified as the visible and the imaginary part as the hidden variable.

The distinction between visible and hidden variables can be formally understood as follows: The original distribution $\rho(x)$ is defined over a set of variables x. Therefore, expectation values need to be computed by integrating over x. The visible variables encode the probabilistic nature of this set of original variables. Accordingly, v has the same dimension as x, and the subspace, spanned by the visible variables, reflects the degrees of freedom of the original state space. The hidden state variables are used to improve the sampling procedure itself. The diversity of the discussed algorithms demonstrates the flexibility that originates from the introduction of additional auxiliary dimensions.

7.2.2 Master equation and detailed balance

A master equation, see Sec. 2.2.1, can be formulated for the set of visible and hidden variables introduced in the previous section in the same manner as for x in Eq. (2.13):

$$\frac{\mathrm{d}p(v,w,\tau)}{\mathrm{d}\tau} = \sum_{v',w'} \Big[p(v',w',\tau)\, W(v',w' \to v,w) - p(v,w,\tau)\, W(v,w \to v',w') \Big] . \tag{7.10}$$

In contrast to the original state x, the evolution is governed by transition probabilities $W(v, w \to v', w')$. They determine how the probability distribution $p(v, w, \tau)$, defined over the higher-dimensional representation space, evolves in time.

In the higher-dimensional state space, the condition for equilibrium reads

$$p(v', w', \tau) \stackrel{!}{=} \sum_{v,w} p(v, w, \tau)\, W(v, w \to v', w')\,. \tag{7.11}$$

In App. B.2, we provide an example of how this relation is fulfilled by the equilibrium distribution of the restricted Boltzmann machine. The detailed-balance equation

$$p(v, w)W(v, w \to v', w') = p(v', w')W(v', w' \to v, w)\,. \tag{7.12}$$

represents again a stronger condition preventing the occurrence of limit cycles, cf. Sec. 2.2.1. The Hamiltonian Monte Carlo algorithm is discussed as an example for satisfying this condition in App. B.1.

7.2.3 Complex Langevin versus HMC / RBM

At this point, it is interesting to have a closer look at the use of auxiliary dimensions in the different algorithms in more detail. We will, in particular, point out differences between complex Langevin dynamics, the Hamiltonian Monte Carlo algorithm, and the restricted Boltzmann machine.

For the latter two algorithms, the visible state v can be identified with the state x in the originally considered problem. This is an important property since it allows the numerical computation of observables in x by considering just the visible states v. Therefore, $v = x$ and thus

$$\langle \mathcal{O}(x) \rangle_\rho = \langle \mathcal{O}(v) \rangle_p = \frac{1}{N} \sum_i^N \mathcal{O}(v_i)\,. \tag{7.13}$$

The auxiliary, hidden variables can be ignored for the computation of observables. The mathematical argument behind this is a possible marginalization of the joint probability distribution $p(x, w)$ according to

$$\rho(x) = \int \mathrm{d}w\, p(x, w)\,. \tag{7.14}$$

This is, however, different for complex Langevin dynamics, which we show by generalizing the way expectation values are computed for this kind of dynamics.

Following the line of arguments in Sec. 7.1.2, the visible states v are no longer identified with the original state x, but are related to them through a linear shift by the hidden variable, cf. Eq. (7.2). The original integral is effectively computed for different substitutions $v \to v + w_i$ in terms of the hidden state variables,

$$\langle \mathcal{O}(x) \rangle_\rho = \frac{1}{N} \sum_{i=1}^N \int_a^b \mathrm{d}v\, \mathcal{O}(v + w_i)\rho(v + w_i)\,, \tag{7.15}$$

where it is assumed that the integral is invariant under the linear shifts of the integral bounds by $-w_i$. This allows using the same integral bounds a and b for all different values of the hidden variables w_i.

As a result, the auxiliary variables contribute to the numerical computation of observables, in which it is summed over samples v_i instead of the continuous integrals,

$$\langle \mathcal{O}(x) \rangle_\rho = \langle \mathcal{O}(v,w) \rangle_\rho = \frac{1}{MN} \sum_{i=1}^{MN} \mathcal{O}(v_i, w_i)\,. \tag{7.16}$$

In the case of complex Langevin dynamics, one may identify v_i with $\phi_{x;i}$ and w_i with $i\phi_{y;i}$, cf. Eq. (7.8).

This a valid approach since we demand that the hidden variables w do not undergo any stochastic evolution. In the case of complex Langevin dynamics, this is realized by a missing noise term and an extrapolation to a vanishing step size in the direction of the hidden states. Therefore, Eq. (7.16) does not compute the expectation value of a joint distribution of both the visible and the hidden states,

$$\langle \mathcal{O}(v,w) \rangle_\rho \neq \int \mathrm{d}v \int \mathrm{d}w\, p(v,w)\mathcal{O}(v,w)\,. \tag{7.17}$$

Instead, only the visible states incorporate the degrees of freedom of the originally considered expectation value.

We note that, mathematically, this is clear in the case of complex Langevin dynamics for a single complex field $\phi = \phi_x + i\phi_y$. The original integral over ϕ is one-dimensional rather than a two-dimensional surface integral over the complex plane. As a result, the sum in Eq. (7.16) returns the mean of the integral for different values of the hidden variables and thus an expectation value with respect to the original distribution $\rho(x)$.

This computation of expectation values with auxiliary variables differs significantly from existing ones. We emphasise that $v = x$ does not hold and a marginalization over w is absent. In the following, we discuss, besides complex Langevin dynamics, several algorithms that implement the above principles and satisfy all of the given constraints for this approach. Keeping all the constraints in mind, one can make use of general relations and methods for sampling from high-dimensional probability distributions, such as Markov chain Monte Carlo methods.

7.3 Substitution sampling

In this section, we formulate in Sec. 7.3.1 the general constraints a sampling algorithm needs to fulfil which serve to compute expectation values of the kind defined in Eq. (7.16). We will refer to this kind of sampling algorithm as *Substitution Sampling* to reflect that it is built on the key insights in Sec. 7.1.2. Additionally, we identify complex Langevin dynamics as such an algorithm and provide a guide for constructing substitution sampling algorithms in Sec. 7.3.3.

7.3.1 General definition

The proposed substitution sampling algorithm generates dynamics in the set of variables (v, w) as a Markov process with transition probabilities $W(v, w \to v', w')$. It distinguishes between an update step that only affects the visible variables and one that only changes the hidden variables. This is implemented by splitting the transition probability into conditional probabilities T and g for visible and hidden states, respectively. The splitting can be done in two ways, with an update first of the

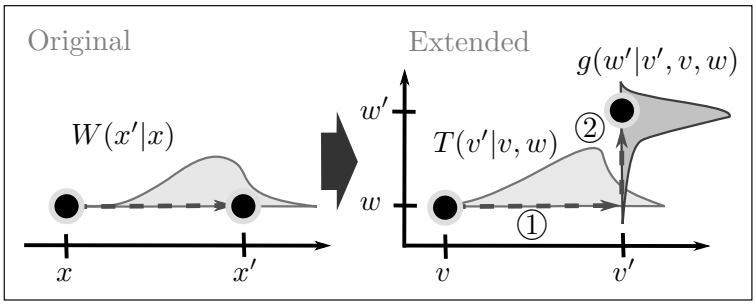

Figure 7.3: Schematic illustration of the transition probabilities (left) in the original and (right) the extended representation space. The visible variables v are updated according to the transition probability $T(v'|v,w)$. The new hidden states w' are obtained in a second step, involving the transition probability $g(w'|v',v,w)$.

visible variables, followed by a conditional update of the hidden ones,

$$W(v,w \to v',w') = g(w'|v',v,w)T(v'|v,w)\,, \tag{7.18}$$

or vice versa,

$$W(v,w \to v',w') = T(v'|v,w',w)g(w'|,v,w)\,. \tag{7.19}$$

In the following, we will only use the first splitting, although both variants are possible. The differences between an update step in the original state space and one in the higher-dimensional state space are schematically shown in Fig. 7.3. One update step consists of a sequential update of the visible and the hidden states.

A substitution sampling algorithm needs to satisfy, in the large-time limit, the following constraints:

1. Satisfaction of the following detailed-balance equation for a fixed hidden state w:

$$p(v,w)g(w'|v',v,w)T(v'|v,w) = p(v',w)g(w'|v,v',w)T(v|v',w)\,. \tag{7.20}$$

2. The hidden states w are updated with an infinitesimal step size.

3. The mean values (7.15) are invariant under shifts of the boundaries a and b by any of the sampled hidden state variables w_i. This is satisfied, for example, if $p(v,w)$ converges sufficiently fast to zero near the integral boundaries.

4. The distribution $p(v,w)$ and the transition probabilities T and g need to satisfy the constraint, cf. Eq. (7.11),

$$p(v',w',\tau) \overset{!}{=} \int \mathrm{d}v \int \mathrm{d}w\, p(v,w,\tau)\, g(w'|v',v,w)T(v'|v,w)\,. \tag{7.21}$$

The first two constraints ensure that the hidden states do not introduce any stochastic behaviour with respect to the distribution $p(v,w)$ and that only the visible states incorporate the degrees of freedom of the originally considered expectation value. In numerical simulations, it can also be sufficient if the stochastic behaviour in the visible direction dominates the one in the hidden direction. This is, for example, the case for complex Langevin dynamics with imaginary noise [216, 219] and for the algorithms discussed in Sec. 7.4.3 and Sec. 7.5.

The last constraint enforces that the substitution sampling algorithm, at long times, formally, samples from the equilibrium distribution $p(v, w)$. As pointed out above, it is feasible to make use of the condition (7.21) since relations of Monte Carlo sampling algorithms in higher dimensions can be used for a computation of observables according to Eq. (7.16) as long as the hidden variables introduce no stochastic contribution to the computed expectation value. Because of this, the actually observed distribution differs from $p(v, w)$. Instead, numerical observables coincide with expectation values with respect to the underlying distribution $\rho(x)$:

$$\langle \mathcal{O}(x) \rangle_\rho = \langle \mathcal{O}(v + w) \rangle_p \,. \tag{7.22}$$

In the case of a complex probability measure $p(v, w)$ the transition probabilities T and g need to be real-valued and positive to allow an actual sampling. In the next section, we show how this is implemented for complex Langevin dynamics.

7.3.2 Complex Langevin as a substitution sampling algorithm

We show in this section that complex Langevin dynamics can be attributed to the class of substitution sampling algorithms. But before that, we want to point out that the complex Langevin equations can also be systematically derived by imposing the respective constraints for complex actions, as worked out explicitly in Chapter 8. The algorithms discussed in Sec. 7.4 are derived in the same way.

The approach allows deriving transition probabilities for complex Langevin dynamics. We use these transition probabilities in the following to prove a satisfaction of constraints no. 1 to no. 4.

Transition probabilities

In concordance with the discussion in Sec. 7.1 and in the previous section, our goal is to show that complex Langevin dynamics, formally, samples from the complex distribution

$$\rho(\phi_x + i\phi_y) \propto \exp(-S(\phi_x + i\phi_y)) \tag{7.23}$$

while satisfying the constraints no. 1 to no. 4 as required for a substitution sampling algorithm. The constraints demand that the stochastic contribution in the ϕ_y direction vanishes in a certain limit. In the following, we will specify this limit for the case of complex Langevin dynamics for which it is reached with an evolution in the continuous Langevin time τ.

We thereby assume that constraint no. 3, namely an invariance under simultaneous shifts of the integration boundaries, is satisfied by the considered observables, which holds independently of the transition probabilities.

Driven by the motivation to view complex Langevin dynamics from the perspective of a Markov chain Monte Carlo algorithm, we start with a discretization of the Langevin time in Eq. (2.10),

$$\begin{aligned} \phi_x' &= \phi_x - \epsilon \,\mathrm{Re}\left[\frac{\delta S(\phi)}{\delta \phi}\bigg|_{\phi_x + i\phi_y}\right] + \sqrt{2\epsilon}\eta \,, \\ \phi_y' &= \phi_y - \epsilon \,\mathrm{Im}\left[\frac{\delta S(\phi)}{\delta \phi}\bigg|_{\phi_x + i\phi_y}\right] \,, \end{aligned} \tag{7.24}$$

where $\epsilon = \Delta\tau$ is the time step in which ϕ_x and ϕ_y evolve to ϕ'_x and ϕ'_y. This formulation allows a numerical implementation of the evolution. The continuous limit in the Langevin time ($\epsilon \to 0$) is evaluated by extrapolating the results of repeated simulations for different values of ϵ.

The update rule for the real part in Eq. (7.24) can be obtained by means of an expansion of the real part of the action difference $\Delta S_{\mathrm{Re}}(\phi', \phi) = S_{\mathrm{Re}}(\phi'_x + i\phi_y) - S_{\mathrm{Re}}(\phi_x + i\phi_y)$ in the transition probability,

$$T(\phi'_x|\phi_x, \phi_y) \propto \varphi\left(\frac{\phi'_x - \phi_x}{\sqrt{2\epsilon}}\right) \exp\left(-\frac{\Delta S_{\mathrm{Re}}(\phi', \phi)}{2}\right) . \tag{7.25}$$

This expression for the transition probability is derived in Chapter 8, cf. Eq. (8.27). Here, φ denotes the Gaussian distribution

$$\varphi(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) . \tag{7.26}$$

Hence, the transition probability T is given by the product of a proposal distribution φ for the new field value ϕ'_x and an acceptance probability that depends on the action difference $\Delta S_{\mathrm{Re}}(\phi', \phi)$.

Analogously, the update equation for the imaginary part of complex Langevin dynamics, see Eq. (7.24), involves the imaginary part of the action difference $\Delta S_{\mathrm{Im}}(\phi', \phi) = S_{\mathrm{Im}}(\phi'_x + i\phi_y) - S_{\mathrm{Im}}(\phi_x + i\phi_y)$, cf. Eq. (8.32),

$$\phi'_y = \phi_y - \epsilon \frac{\Delta S_{\mathrm{Im}}(\phi', \phi)}{\phi'_x - \phi_x} . \tag{7.27}$$

Since it does not contain any noise term, the respective conditional transition probability is a delta-distribution,

$$g(\phi'_y|\phi'_x, \phi_x, \phi_y) = \delta\left(\phi'_y - \phi_y + \epsilon \frac{\Delta S_{\mathrm{Im}}(\phi', \phi)}{\phi'_x - \phi_x}\right) . \tag{7.28}$$

The transition probability g defines the update rule for ϕ'_y, where we use that $x = \phi$, $v = \phi_x$ and $w = i\phi_y$. An expansion of the action difference to first order yields the update equation of the imaginary part of complex Langevin dynamics. In the limit $\epsilon \to 0$, constraint no. 2, demanding an infinitesimal step size into the ϕ_y direction, is thus obeyed.

Note that, in the derivation of both update rules, the action difference involves a change in ϕ_x only. This is in accordance with the condition that only the visible variables represent the degrees of freedom of the initially considered expectation value over x.

The derivation of the discrete update equations (7.24) is performed explicitly in App. C.1, starting from the transition probabilities (7.25) and (7.28).

Langevin symmetry

We point out that the transition probability (7.28) for the imaginary part is invariant under an exchange of ϕ'_x and ϕ_x,

$$g(\phi'_y|\phi'_x, \phi_x, \phi_y) = g(\phi'_y|\phi_x, \phi'_x, \phi_y) . \tag{7.29}$$

We will refer to this symmetry as *Langevin symmetry*, which will be a key ingredient for the construction of substitution sampling algorithms. See Fig. 7.4 for an illustration of the symmetry.

If the Langevin symmetry holds, constraint no. 1 reduces to

$$p(\phi_x, \phi_y) T(\phi'_x|\phi_x, \phi_y) \overset{!}{=} p(\phi'_x, \phi_y) T(\phi_x|\phi'_x, \phi_y) , \tag{7.30}$$

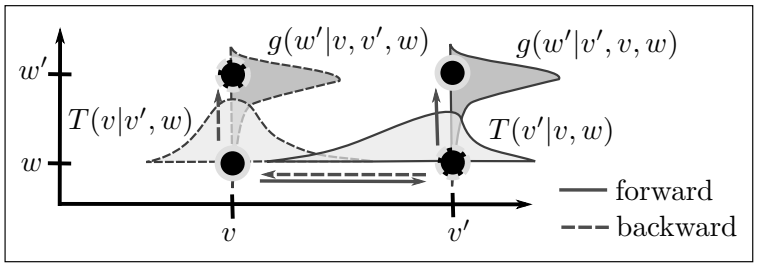

Figure 7.4: Illustration of the Langevin symmetry defined in Eqs. (7.29) and (7.36). The transition probabilities for the visible variables in the forward and the backward directions of the adapted detailed-balance equation (7.37) are different. In contrast, the transition probability for the hidden state w is invariant under an exchange of v' and v.

where $p(\phi_x, \phi_y) = \rho(\phi_x + i\phi_y)$, cf. Eq. (7.23).

In fact, the transition probability T, defined in Eq. (7.25), violates this modified detailed-balance equation, since:

$$p(\phi_x, \phi_y) T(\phi'_x|\phi_x, \phi_y) = p(\phi'_x, \phi_y) T(\phi_x|\phi'_x, \phi_y) \exp(-i\Delta S_{\mathrm{Im}}(\phi, \phi')) \,. \tag{7.31}$$

However, Eq. (7.31) is satisfied if the step size into the ϕ_x direction is also chosen to be infinitesimal. This is ensured in the limit $\epsilon \to 0$ since the proposal distribution converges to a delta-distribution around ϕ_x,

$$\lim_{\epsilon \to 0} \frac{1}{\sqrt{2\epsilon}} \varphi\left(\frac{\phi'_x - \phi_x}{\sqrt{2\epsilon}}\right) = \delta(\phi'_x - \phi_x) \,. \tag{7.32}$$

The infinitesimal step size in the ϕ_x direction justifies the previously performed expansion in the action difference and ensures constraint no. 4 to be fulfilled:

$$p(\phi'_x, \phi'_y, \tau) \overset{!}{=} \int \mathrm{d}\phi_x \int \mathrm{d}\phi_y \, p(\phi_x, \phi_y, \tau) \, g(\phi'_y|\phi'_x, \phi_x, \phi_y) T(\phi'_x|\phi_x, \phi_y) \,. \tag{7.33}$$

This is proven as follows. We start by inserting relation (7.31) into Eq. (7.33). We then make use of the symmetry (7.29) and finally expand the action difference $\Delta S_{\mathrm{Im}}(\phi, \phi')$ to first order around ϕ'_x, which gives

$$\begin{aligned} p(\phi'_x, \phi'_y, \tau) \overset{!}{=} \int \mathrm{d}\phi_x \int \mathrm{d}\phi_y \, p(\phi'_x, \phi_y, \tau) \, g(\phi'_y|\phi'_x, \phi_y) \\ \times \, T(\phi_x|\phi'_x, \phi_y) \, \exp\left(-i(\phi_x - \phi'_x) \frac{\delta S_{\mathrm{Im}}(\phi'_x + i\phi_y)}{\partial \phi'_x}\right) \,. \end{aligned} \tag{7.34}$$

Here, $g(\phi'_y|\phi'_x, \phi_y) \equiv g(\phi'_y|\phi_x, \phi'_x, \phi_y)$, i.e., the ϕ_x-dependence of the transition probability g can be dropped due to the expansion of $\Delta S_{\mathrm{Im}}(\phi, \phi')$ to first order around ϕ'_x. The expansion is justified in the limit $\epsilon \to 0$, where also ϕ_x changes by infinitesimal amounts only. In this limit, it is possible to absorb the exponential function in Eq. (7.34) into the transition probability $T(\phi_x|\phi'_x, \phi_y)$. See App. C.5 for further details.

We can now integrate over ϕ_x since the transition probability on the right-hand side is the only distribution depending on ϕ_x and, using its normalization, we are left with

$$p(\phi'_x, \phi'_y, \tau) \overset{!}{=} \int \mathrm{d}\phi_y \, p(\phi'_x, \phi_y, \tau) \, g(\phi'_y|\phi'_x, \phi_y) \,. \tag{7.35}$$

As a last step, we take the limit $\epsilon \to 0$. In this limit, by the definition of the conditional transition probability g, Eq. (7.35) is indeed satisfied by $p(\phi_x, \phi_y) = \lim_{\tau\to\infty} p(\phi_x, \phi_y, \tau)$. This completes the proof.

We conclude that the step sizes in configuration space need to be infinitesimal in both the ϕ_x and the ϕ_y directions in order to fulfil the constraints no. 1 to no. 4 required for a substitution sampling algorithm. Hence, for $\epsilon \to 0$, the transition probabilities (7.25) and (7.28) are equivalent to the discretized update rules (7.24) and thus to complex Langevin dynamics.

7.3.3 Constructing substitution sampling algorithms

In the following, we generalize the key concepts of complex Langevin dynamics discussed in the previous section to the case of general visible and hidden variables and thus also to finite step sizes in the visible direction. This generalization provides a possible approach to constructing transition probabilities that satisfy all of the constraints a substitution sampling algorithm must fulfil.

We start again by demanding that the transition probability for the hidden variables obeys the Langevin symmetry (see also Fig. 7.4)

$$g(w'|v', v, w) = g(w'|v, v', w)\,. \tag{7.36}$$

With this symmetry, the detailed-balance equation (7.20) can be written as

$$g(w'|v', v, w) \times \big[p(v, w)T(v'|v, w) - p(v', w)T(v|v', w)\big] \overset{!}{=} 0\,. \tag{7.37}$$

For non-vanishing g, the term in square brackets, referred to as adapted detailed-balance equation, must vanish, which constrains the transition probabilities $T(v'|v, w)$.

The meaning of the adapted detailed-balance equation becomes clearer when one takes a closer look at the equation: It can be viewed as a detailed-balance equation of a Markov chain that allows changes in the visible state variables v only, whereby w is fixed. The process is unaware of any dependence on the additional auxiliary variables w. Nevertheless, w will be updated based on $g(w'|v', v, w)$. This entails a transformation of the environment for the Markov chain in v after each update step since the action depends on w.

The above interpretation mirrors the important concept of the substitution sampling algorithm for computing observables by means of an integration over the visible variables only. In contrast, the hidden variables give rise to a continuous set of different substitutions of the dynamical variables in the originally considered integral and carry no stochastic behaviour, cf. Eq. (7.15).

It remains to derive transition probabilities g for the hidden states that are in concordance with the constraints no. 2 to no. 4.

In the following, we point out possible implications that result from constraint no. 4, Eq. (7.21). We insert the detailed-balance equation (7.20) into the right-hand side of Eq. (7.21),

$$\begin{aligned} p(v', w', \tau) &= \int \mathrm{d}v \int \mathrm{d}w\, g(w'|v', v, w)T(v'|v, w)p(v, w, \tau) \\ &= \int \mathrm{d}v \int \mathrm{d}w\, g(w'|v, v', w)T(v|v', w)p(v', w, \tau)\,. \end{aligned} \tag{7.38}$$

Inspired by the first-order expansion (7.34) in the case of complex Langevin dynamics, we here demand that g does not depend on v,

$$g(w'|v, v', w) \equiv g(w'|v', w)\,. \tag{7.39}$$

As for complex Langevin dynamics, this allows performing the integration over v in Eq. (7.38), resulting in

$$p(v', w', \tau) \overset{!}{=} \int \mathrm{d}w\, g(w'|v', w) p(v', w, \tau)\,. \tag{7.40}$$

Next, we make use of constraint no. 2, which suggests that g is of the form

$$g(w'|v', w) = \delta\left(w' - h(v', w; \epsilon)\right)\,, \tag{7.41}$$

where $\delta(\cdot)$ represents the delta-distribution and the function $h(v', w; \epsilon)$ has the property that

$$\lim_{\epsilon \to 0} h(v', w; \epsilon) = w\,. \tag{7.42}$$

Here, the parameter ϵ parametrizes the step size in the update process of the hidden states. With the above assumptions on g, one can take the limit $\epsilon \to 0$ and integrate over w, which confirms constraint (7.40) to hold and therefore constraint no. 4, cf. Eq. (7.21).

Note that for a transition from $(v, w) \to (v', w')$, one needs to replace v' by v in Eq. (7.41),

$$g(w'|v, w) = \delta\left(w' - h(v, w; \epsilon)\right)\,. \tag{7.43}$$

Complex Langevin dynamics deviates from this construction in the sense that the adapted detailed-balance equation (7.37) is only warranted when step sizes into the visible direction are infinitesimal, too.

In the next chapter, we introduce examples of algorithms that are constructed based on the same principles as complex Langevin dynamics. Sec. 7.5 then provides an example of an algorithm that satisfies the constraints of a substitution sampling algorithm in a different way.

7.4 Complex Langevin-type algorithms

The analysis of complex Langevin dynamics and the above guide for constructing substitution sampling algorithms can be combined to define a systematic approach to deriving transition probabilities T and g. The resulting algorithms differ in their proposal distributions and satisfy the constraints of substitution sampling in the same manner as complex Langevin dynamics.

This systematic approach is described in detail in Chapter 8. It is inspired by an alternative derivation of complex Langevin dynamics which recovers known results from a different point of view. The core concepts of the derivation are: an extension of the transition probabilities of Langevin dynamics to a higher-dimensional state space and a compensation of certain (here imaginary) contributions in the action by terms that emerge in the transition to the extended state space. The approach is built on the requirement that the Langevin symmetry as well as constraints no. 1 to no. 4 stated in Sec. 7.3.1 are obeyed. It then leads to the transition probabilities of, e.g., complex Langevin dynamics, cf. Eqs. (7.25) and (7.28).

The cancellation of imaginary contributions of the action is a crucial step in this derivation, cf. Eqs. (8.6) and (8.7) in Chapter 8. In the case of complex Langevin dynamics, the update of ϕ'_y of the imaginary field ϕ_y is used for this. This compensation leads to well-defined, real-valued transition probabilities and, therefore, allows an actual sampling of problems with a sign problem.

In the following, we present several algorithms resulting from the systematic approach. Detailed derivations of these algorithms are given in App. C.

7.4.1 Second-order complex Langevin

Second-order complex Langevin dynamics results as a refinement of complex Langevin dynamics. For this, also the second-order term of the Taylor expansion, cf. Eq. (C.3), of the action difference around ϕ_x is taken into account. The resulting update rule for the real part of the field is

$$\phi'_x = \phi_x - \left(\epsilon \frac{\delta S_{\mathrm{Re}}}{\delta \phi_x} + \sqrt{2\epsilon}\eta\right) \Bigg/ \left(1 + \frac{\epsilon}{2}\frac{\delta^2 S_{\mathrm{Re}}}{\delta \phi_x^2}\right), \tag{7.44}$$

and, for the imaginary part,

$$\phi'_y = \phi_y - \epsilon \frac{\delta S_{\mathrm{Im}}}{\delta \phi_x} - \frac{\epsilon}{2}\left(\phi'_x - \phi_x\right)\frac{\delta^2 S_{\mathrm{Im}}}{\delta \phi_x^2}, \tag{7.45}$$

where we defined $S_{\mathrm{Re}} := S_{\mathrm{Re}}(\phi_x + i\phi_y)$ and $S_{\mathrm{Im}} := S_{\mathrm{Im}}(\phi_x + i\phi_y)$. As before, the update rule samples from the desired equilibrium distribution in the limit of $\epsilon \to 0$ since detailed balance is satisfied only in this limit. Details on the derivation can be found in App. C.2. A numerical comparison to complex Langevin dynamics will be presented in Sec. 7.6.

7.4.2 Complex hat function algorithm

Complex Langevin dynamics uses a Gaussian distribution φ, cf. Eq. (7.26), in proposing states ϕ'. We demonstrate, in this section, that the systematic derivation of Langevin-type sampling algorithms does also work for other types of proposal distributions. In particular, we consider the triangular hat function,

$$\eta_\epsilon(\phi' - \phi) = \frac{1}{\epsilon}\begin{cases} 1 - \frac{\phi' - \phi}{\epsilon} & \text{for } 0 \leq \phi' - \phi < \epsilon\,, \\ 1 + \frac{\phi' - \phi}{\epsilon} & \text{for } -\epsilon < \phi' - \phi < 0\,, \\ 0 & \text{otherwise.} \end{cases} \tag{7.46}$$

as a proposal distribution. The limit $\epsilon \to 0$ facilitates the implementation of an infinitesimal step size in configuration space. This is a necessary condition to satisfy the constraints of the substitution algorithm, as worked out in Chapter 8.

We assume again a complex action, as defined for the polynomial model in Eq. (2.1), and define an update scheme that allows sampling despite a sign problem. The respective update rules for ϕ_x and ϕ_y are derived with the help of the systematic derivation in App. C.3.

The update rule for the imaginary field ϕ_y is given by:

$$\phi'_y = \phi_y + \left[\frac{\epsilon}{s} - \left(\phi'_x - \phi_x\right)\right] \tan\left(-\frac{\Delta S_{\mathrm{Im}}(\phi', \phi)}{2}\right), \tag{7.47}$$

where

$$s = \operatorname{sign}(\phi'_x - \phi_x)\,. \tag{7.48}$$

The update rule is invariant under an exchange of ϕ'_x and ϕ_x and thus possesses the Langevin symmetry, Eq. (7.36).

The update rule compensates the contributions from the imaginary part of the action as it is also the case for complex Langevin dynamics. This compensation leads to a real-valued transition probability for the real part of the field ϕ_x, namely,

$$T(\phi'_x|\phi_x,\phi_y) = \frac{1}{\epsilon N(\phi)} \exp\left(-\frac{\Delta S_{\mathrm{Re}}(\phi',\phi)}{2}\right)\left(1 - s\frac{\phi'_x - \phi_x}{\epsilon}\right)\cos^{-1}\left(-\frac{\Delta S_{\mathrm{Im}}(\phi',\phi)}{2}\right). \tag{7.49}$$

In contrast to complex Langevin dynamics, it is not trivial to translate this transition probability in an update rule for ϕ_x. Instead, we sample a new state ϕ'_x implicitly by numerically solving the transformation of the transition probability to a uniform distribution,

$$\int_{-\infty}^{\phi'_x} \mathrm{d}\tilde{\phi}_x\, T(\tilde{\phi}_x|\phi_x,\phi_y) \overset{!}{=} \int_0^r \mathrm{d}\tilde{r} = r\,. \tag{7.50}$$

In practice, one samples r from the uniform distribution and numerically solves the expression on the left-hand side for ϕ'_x, so that the equality is satisfied for the sampled r. It is important that the transition probability T represents a probability distribution. In the limit of $\epsilon \to 0$ this is indeed the case.

7.4.3 Uniform complex Langevin

A substitution sampling algorithm can also be formulated for a uniform proposal distribution. We achieve this by defining the proposal distribution by means of an integrated delta-distribution

$$q(\phi \to \phi') = \int_{-l}^{l} \frac{\mathrm{d}r}{2l}\,\delta\left(\phi' - (\phi + r)\right) = \frac{1}{2l}\left[\Theta\left(\phi' - \phi + l\right) - \Theta\left(\phi' - \phi - l\right)\right], \tag{7.51}$$

and implement it by sampling r uniformly from the interval $[-l, l]$.

The resulting update rules are

$$\begin{aligned} T(\phi'_x|\phi_x,\phi_y) \propto \int_{-l}^{l} \frac{\mathrm{d}r}{2l}\,\delta\left(\phi'_x - (\phi_x + r)\right) \\ \times \exp\left(-\frac{\Delta S_{\mathrm{Re}}(\phi',\phi)}{2}\right)\cos^{-1}\left(-\frac{\Delta S_{\mathrm{Im}}(\phi',\phi)}{2}\right) \end{aligned} \tag{7.52}$$

for the real part ϕ_x and

$$\phi'_y = \phi_y + \left(\tilde{\phi}'_x - (\phi_x + r)\right)\tan\left(-\frac{\Delta S_{\mathrm{Im}}(\phi',\phi)}{2}\right) \tag{7.53}$$

for the imaginary part ϕ_y of the field. Sampling a state ϕ'_x works in the same manner as for the complex hat function algorithm by a transformation of the transition probability, cf. Eq. (7.50). In contrast to the other approaches, two proposal states, ϕ'_x and $\tilde{\phi}'_x$ are sampled. This entails a finite

step size for ϕ_y. The algorithm satisfies all constraint of a substitution algorithm in the limit of $l \to 0$, see Chapter 8 for details.

In principle, any other proposal distribution can be used as long as constraint (7.41) for the imaginary update rule is satisfied and potentially introduced noise in the imaginary direction is dominated, in numerical simulations, by the noise in the real direction in the limit of infinitesimally small step sizes.

7.4.4 Metropolis-like sampling

In principle, it is also possible to define a Metropolis [253] accept/reject step based on the adapted detailed-balance equation (7.30). The acceptance probability is approximated by

$$A(\phi'_x|\phi_x,\phi_y) = \min\left[1, \exp\left(-(S_{\mathrm{Re}}(\phi'_x + i\phi_y) - S_{\mathrm{Re}}(\phi_x + i\phi_y))\right]\right. . \tag{7.54}$$

where a respective transition probability T is defined as the product of a symmetric proposal distribution for ϕ'_x and the acceptance probability according to:

$$T(\phi'_x|\phi_x,\phi_y) = q(\phi'_x|\phi_x)A(\phi'_x|\phi_x,\phi_y)\,. \tag{7.55}$$

The adapted detailed-balance equation is violated by this definition in the same way as for complex Langevin dynamics and the other complex Langevin-type algorithms in this section, cf. Eq. (7.31). Accordingly, the sampling algorithm works also only in the limit of infinitesimally small step sizes in ϕ_x. The imaginary part ϕ_y is updated based on the associated update equation g of the used proposal distribution, independent of an acceptance or a rejection of the proposed state.

7.5 Substitution Hamiltonian Monte Carlo sampling in auxiliary dimensions

We present, in this section, as a proof of concept, an alternative algorithm satisfying constraints no. 1 to no. 4 of a substitution sampling algorithm defined in Sec. 7.3.1. We call the algorithm *Substitution Hamiltonian Monte Carlo Sampling* (SHMCS). The algorithm is not derived within the systematic approach introduced in Chapter 8. Instead, it makes use of the basic idea of the Hamiltonian Monte Carlo algorithm to introduce an additional momentum as an auxiliary dimension. The SHMCS algorithm only works for real actions and cannot be applied to problems with a sign problem. However, it serves as a good example and provides numerical evidence in support of the general framework introduced in this work.

We consider a (real) probability distribution $\rho(x)$ and want to design an algorithm for computing observables according to Eq. (7.16),

$$\langle \mathcal{O}(x)\rangle = \langle \mathcal{O}(v+w)\rangle = \frac{1}{MN}\sum_{i=1}^{MN} \mathcal{O}(v_i + w_i)\,. \tag{7.56}$$

In contrast to complex Langevin dynamics, the hidden variables w are taken to be real and introduced by the substitution

$$x = x(v) = v + w\,, \quad \mathrm{d}x = \mathrm{d}v\,. \tag{7.57}$$

We assume that the probability distribution $p(v+w)$ satisfies the necessary constraints for a valid computation of expectation values by Eq. (7.56).

In addition, we introduce momenta π as further hidden variables. Similar to the Hamiltonian Monte Carlo algorithm, the momenta are related to the visible variables v through an energy function

$$H(v,w,\pi) := S(v+w) + \frac{\pi^2}{2m}\,. \tag{7.58}$$

Next, we split the action into two contributions:

$$S(v+w) = S_1(v+w) + S_2(v+w)\,. \tag{7.59}$$

This step is similar to the distinction between the real and the imaginary part of a complex action. For a real action, however, the splitting is arbitrary.

In contrast to the HMC algorithm, we demand

$$\tilde{H}(v,\pi) := S_2(v+w) + \frac{\pi^2}{2m} \tag{7.60}$$

to stay constant during the Monte Carlo evolution which is implemented by updating v and π according to the differential equations

$$\frac{\mathrm{d}v}{\mathrm{d}t} = \frac{\partial \tilde{H}(v,\pi)}{\partial \pi}\,, \quad \frac{\mathrm{d}\pi}{\mathrm{d}t} = -\frac{\partial \tilde{H}(v,\pi)}{\partial v}\,. \tag{7.61}$$

which is assumed to be possible in a numerically exact manner. The remaining contribution $S_1(x)$ is taken into account through the acceptance term

$$A(v'|v,w) = \min\left[1, \exp\left(-\left(S_1(v'+w) - S_1(v+w)\right)\right)\right]\,. \tag{7.62}$$

This approach satisfies, so far, the detailed-balance equation of a substitution sampling algorithm, cf. Eq. (7.20). Therefore, we are free to choose an update rule for the transition probability of the hidden state w as long as the constraints defined in Sec. 7.3.1 are satisfied. We define the transition probability g as a Langevin process with finite step size,

$$g(w'|w) = \varphi\left(\frac{w'-w}{\sqrt{2\epsilon}} + \sqrt{\frac{\epsilon}{2}}\,\theta w\right)\,, \tag{7.63}$$

with the Gaussian distribution φ, cf. Eq. (7.26). The transition probability translates in the limit $\epsilon \to 0$ in the Langevin evolution

$$\frac{\mathrm{d}w}{\mathrm{d}t} = -\theta w + \eta\,, \tag{7.64}$$

with Gaussian noise η.

Putting everything together, the SHMCS algorithm is defined based on the following transition probabilities

$$\begin{aligned} T(v'|v,w,\pi) &\propto \delta\left(v' - R\,\Phi_v(v,w,\pi)\right) \\ &\qquad \times \min\left[1, \exp\left(-(S_1(v'+w) - S_1(v+w))\right)\right] , \\ g(\pi'|v,w,\pi) &= \delta\left(\pi' - R\,\Phi_\pi(v,w,\pi)\right) , \\ g(w'|w) &= \varphi\left(\frac{w'-w}{\sqrt{2\epsilon}} + \sqrt{\frac{\epsilon}{2}}\,\theta w\right) . \end{aligned} \tag{7.65}$$

The functions $\Phi_v(v,w,\pi)$ and $\Phi_\pi(v,w,\pi)$ encode the end point of an evolution according to the differential equations (7.61) for a finite amount of time. The operator R negates the momenta π after the evolution. Similar to the HMC algorithm, this ensures reversibility.

The transition probabilities satisfy all constraints of a substitution sampling algorithm. According to the fourth constraint, the SHMCS algorithm formally samples from the equilibrium distribution

$$p(v,w,\pi) \propto \exp\left(-H(v,w,\pi)\right) . \tag{7.66}$$

As for complex Langevin dynamics, the actually observed steady-state distribution differs, due to the properties of the substitution sampling algorithm, from this distribution. The difference results from the required vanishing stochastic contribution in the direction of the hidden variables. In the case of the SHMCS algorithm, this requirement is violated by the Gaussian noise distribution in the transition probability g. However, these stochastic contributions are dominated in the limit of $\epsilon \to 0$ by finite correlations in the visible variables, resolving a violation of the requirement. In practice, this can be ensured by choosing θ sufficiently large.

7.6 Numerical results

In the remainder of this work, we briefly examine the applicability of our approach and the algorithms derived with it by a numerical evaluation for the polynomial model defined in Eq. (2.1) in Sec. 2.1.1,

$$S(\phi) = \frac{1}{2}\left(\sigma_{\mathrm{Re}} + i\sigma_{\mathrm{Im}}\right)\phi^2 + \frac{\lambda}{4}\phi^4 . \tag{7.67}$$

Expectation values for benchmarking are analytically accessible for the chosen set of parameters, cf. Ref. [216].

For the complex Langevin-type algorithms, we compare, in Fig. 7.5, the impact of finite step sizes on a possible extrapolation to the continuous limit and the performance for a fixed step size but a different severity of the sign problem, i.e., a more oscillating measure. The considered algorithms are defined in Tab. 7.1. The dependence of the measured average step size $\langle \phi_x \rangle$ in the real direction and on the chosen step size parameter ϵ is shown in Fig. 7.6.

The results in Fig. 7.5 show that none of the studied algorithms entails a significant difference to complex Langevin dynamics, with the exception of the Metropolis-like algorithms. The deviations can be traced back to the asymmetry between the accept and reject step for the real field variable and the independent update step of the imaginary field. The slightly worse convergence of second-order complex Langevin dynamics is likely related to an asymmetry in the adapted detailed-balance

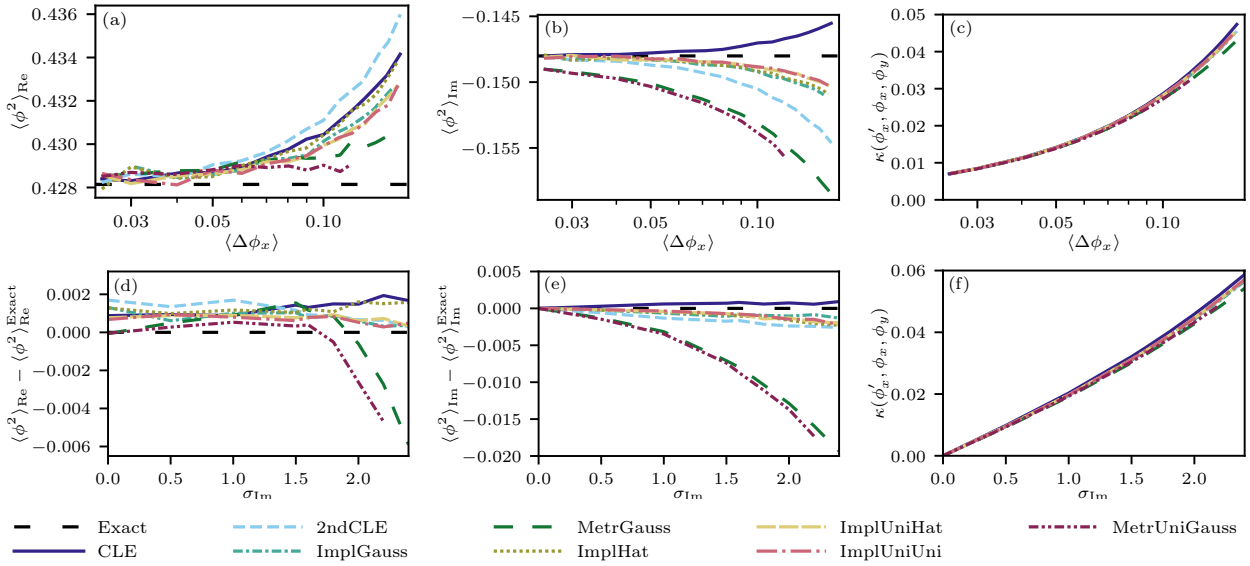

Figure 7.5: Comparison of numerical results for different complex Langevin-type sampling algorithms for the polynomial model (7.67). Details about the algorithms are given in Tab. 7.1. (a)-(c) Results for $\lambda = \sigma_{\mathrm{Re}} = \sigma_{\mathrm{Im}} = 1$ and a varying average step size $\langle\Delta\phi_x\rangle$ in the real direction of the representation space. (d)-(f) Results for $\lambda = \sigma_{\mathrm{Re}} = 1$ and a varying σ_{Im}. To get an appropriate comparison, the step sizes in the real direction were chosen to be equal for all algorithms. The plots (c) and (f) measure the violation of the adapted detailed-balance equation (7.30) based on the measure $\kappa(\phi'_x, \phi_x, \phi_y)$ defined in Eq. (8.37) in Chapter 8. In concordance with Sec. 7.3.2, the measure increases with the real step size and with the magnitude of the imaginary contribution, regulated by σ_{Im}.

equation and the Langevin symmetry, cf. Eqs. (7.29) and (7.30), introduced by the second order term of the Taylor expansion around ϕ_x, cf. Eq. (C.9).

The SHMCS algorithm is tested in a similar way. In this case, we compare the results with those from a real Langevin equation with one hidden variable. The algorithm is derived in the same manner as complex Langevin, but with a substitution $\phi = v + w$. Recall that the resulting Langevin equation with one hidden variable only works for real actions. We split the action according to

$$S(v+w) = S_1(v+w) + S_2(v+w)\,, \tag{7.68}$$

with

$$\begin{aligned} S_1(v+w) &= \frac{\sigma_{\mathrm{Re}}}{2} w^2 + \frac{\lambda}{4}\left(v^4 + 6v^2w^2 + 4vw^3\right)\,, \\ S_2(v+w) &= \frac{\sigma_{\mathrm{Re}}}{2}\left(v^2 + 2vw\right) + \frac{\lambda}{4}\left(4v^3w + w^4\right)\,. \end{aligned} \tag{7.69}$$

The numerical results in Fig. 7.7 support the theoretical framework presented in this work. In particular, the SHMCS algorithm allows larger step sizes in the visible direction due to an exact satisfaction of the detailed-balance equation (7.20). As pointed out at the end of Sec. 7.5, an infinitesimally small step size in the direction of the hidden variables is implemented by taking the limit $\epsilon \to 0$. The numerical results confirm the discussed restriction that stochastic contributions in the hidden variables need to be dominated by correlations in the visible variables. If the step size into the hidden direction is not small enough, compared to that in the visible direction, this

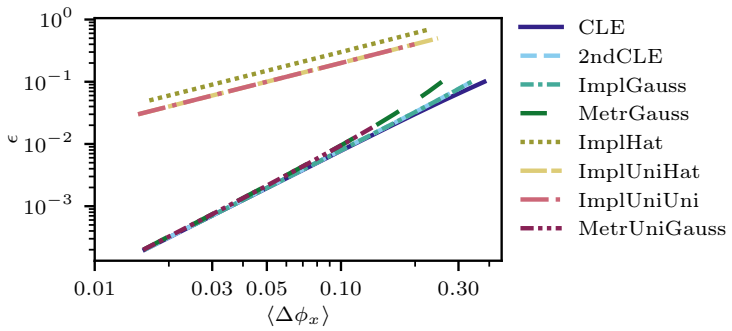

Figure 7.6: Relation between the parameter ϵ and the actually observed step size $\langle \Delta\phi_x \rangle$ in real direction for the different implemented complex Langevin-type algorithms.

domination does no longer hold. This can be observed in Fig. 7.7 for large values of ϵ and small step sizes in the real direction.

7.7 Summary and outlook

We embed complex Langevin dynamics into a generalized framework. The framework is built on the idea to substitute the integration variable in the integrals for the computation of correlations and expectation values of observables. Auxiliary parameters which are introduced by this substitution are utilized to define a Markov chain Monte Carlo algorithm that operates in a higher-dimensional state space. This space is spanned by the original representation of the state and the introduced auxiliary, hidden state variables. The sampling algorithm smoothly interpolates between different transformations of the integration variable and allows a computation of observables based on samples drawn in the Markov process. The sign problem can be circumvented in this way by a smart choice of the transition probabilities of the Markov process. Complex Langevin dynamics is derived as one possible example for such an algorithm.

The introduced substitution sampling algorithm formalizes the approach as a more general algorithm that computes observables based on this idea. We provide the necessary constraints any such algorithm must be subject to. Furthermore, the algorithms derived indicate possible directions to go within the given framework.

We anticipate that the presented derivation of complex Langevin dynamics provides the possibility for the development of novel algorithms and for the understanding of existing ones for simulating theories with a sign problem. For example, one might analyse a replacement of the substitution of the integral for the observables by a non-linear transformation, similar to the work in Ref. [80] or investigate further (existing) Markov chain Monte Carlo methods in auxiliary dimensions for a possible adaptation to complex measures. Furthermore, distributions sampled by means of a process in a real extended representation space might have overlap with a distribution sampled by complex Langevin dynamics. This property makes an application of reweighting in the extended space appear attractive. The approach is similar to reweighting in the complex plane, studied in Ref. [295]. Lastly, the provided mathematical constraints enable an integration of machine learning algorithms into the sampling procedure since the constraints allow the formulation of objective functions for training.

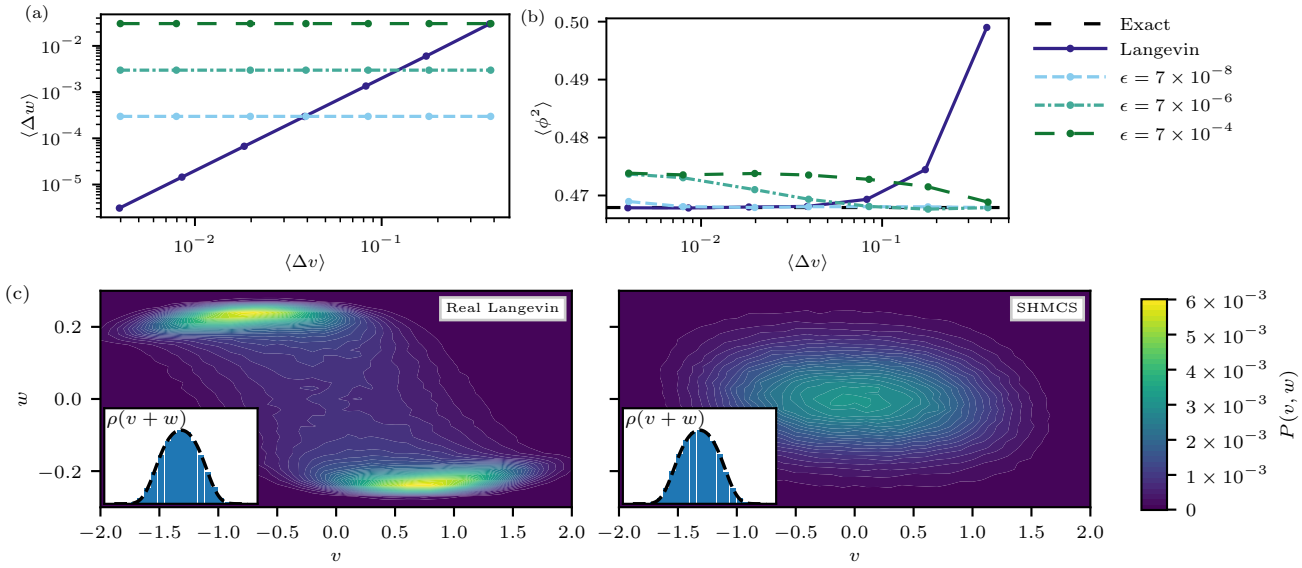

Figure 7.7: Comparison of the SHMCS algorithm, for $\theta = 100$, and the real Langevin equation in one auxiliary dimension for the polynomial model (7.67) with $\lambda = \sigma_{\mathrm{Re}} = 1$ and $\sigma_{\mathrm{Im}} = 0$. The step size of the SHMCS algorithm in the visible direction is regulated by the evolution time with respect to the Hamilton's equations (7.61) and the one in the hidden direction by the parameter ϵ, cf. Eq. (7.63). (a) Interrelationship between the step sizes in the visible and the hidden dimension. Inherent to the SHMCS algorithm, the step size in the hidden direction is independent of the real one but changes in dependence of ϵ. For real Langevin dynamics, the step sizes are related to each other. (b) Convergence of the algorithms to the analytical result of the observable $\langle \phi^2 \rangle$ as a function of the step size in the visible direction. In contrast to the real Langevin equation, exact results are obtained for the SHMCs algorithm also for large visible step sizes. This is an important observation since it shows that the provided theoretical framework in this work is correct. Furthermore, it demonstrates that, in principle, sampling from distributions with a complex contribution with larger step sizes in the visible direction is possible. The numerical results deviate for larger values of ϵ and smaller step sizes into the real direction. This property can be traced back to the constraint that the considered correlations in the visible direction need to be dominant, which is no longer the case in this limit. (c) Distribution $P(v, w)$ of the two algorithms in the higher-dimensional representation space. The histograms in the smaller plots confirm that the algorithms sample in both cases from the target distribution $\rho(\phi) = p(v + w)$, which is indicated by the dashed black line.

Name	Transition probabilities	Proposal distribution	Action difference expansion	Real update
CLE	Gaussian (Eq. (7.24))	Gaussian	1st order	Explicit
2ndCLE	Gaussian (Eqs. (7.44) and (7.45))	Gaussian	2nd order	Explicit
ImplGauss	Gaussian (Eqs. (7.25) and (7.27))	Gaussian	Exact	Implicit
MetrGauss	Gaussian (Eqs. (7.25) and (7.27))	Gaussian	Exact	Metropolis
ImplHat	Hat Function (Eqs. (7.47) and (7.49))	Hat Function	Exact	Implicit
ImplUniHat	Uniform (Eqs. (7.52) and (7.53))	Hat Function	Exact	Implicit
ImplUniUni	Uniform (Eqs. (7.52) and (7.53))	Uniform	Exact	Implicit
MetrUniGauss	Uniform (Eqs. (7.52) and (7.53))	Gaussian	Exact	Metropolis

Table 7.1: Details about the different studied algorithms in Fig. 7.5 and Fig. 7.6. The algorithms differ in their utilized transition probabilities. For the uniform transition probability, different proposal distributions are considered. The last column indicates how the update of the real part ϕ_x is implemented. For complex Langevin dynamics and the second order complex Langevin algorithm, an explicit update rule can be formulated. The implicit update is performed based on a transformation of the probability density, cf. Eqs. (7.50) and (C.7). The Metropolis update accepts or rejects a proposal state based on Eq. (7.54).

CHAPTER 8

Complex Langevin-type sampling by compensation

This chapter is based on Ref. [5].

In this chapter, we present a systematic approach to deriving an implementation of a substitution sampling algorithm, see Sec. 7.3. The algorithm represents a generalization of complex Langevin dynamics and is based on the idea to consider different proposal distributions in a respective Markov chain. We refer to the algorithm as *Langevin sampling by compensation*.

Our approach has two key ingredients: The first one is the reformulation of the transition probabilities of Langevin dynamics as functions of a set of visible and hidden variables. This leads to dynamics in a higher-dimensional state space. As pointed out in Sec. 7.2, the visible and hidden variables correspond, for complex Langevin, to the real and the imaginary parts of the field.

The second key ingredient is a compensation of certain contributions to the transition probabilities by terms which arise from the hidden variables. This feature is unique to the approach. In particular, it is useful for problems with a complex action where the imaginary part prevents an application of standard Monte Carlo algorithms. In these cases, the imaginary part can be compensated by the introduced imaginary part of the field. Initially complex transition probabilities can be adapted to get real-valued. The property is utilized in the complex Langevin-type algorithms, presented already in the previous chapter in Sec. 7.4, which are all derived based on the here presented systematic derivation.

8.1 Complex Langevin dynamics by compensation

We begin with a slightly simplified derivation of complex Langevin dynamics, while using the two mentioned key ingredients. As a reminder, this corresponds in Fig. 7.1 in Chapter 7 to the transition from a real-valued action to a complex action, depicted by the golden arrow on the left-hand side. The line of arguments of this derivation is different from the standard derivation and provides a good understanding of the key ingredients. The derivation was the initial impulse for the results of this and of the previous chapter. A generalization is discussed in the next section.

Consider the real Langevin equation, cf. Eq. (2.5) in Sec. 2.1.2, discretized in the Langevin time τ:

$$\phi' = \phi - \epsilon \frac{\delta S(\phi)}{\delta \phi} + \sqrt{2\epsilon}\eta\,, \tag{8.1}$$

where $\phi' = \phi(\tau + \epsilon)$ and $\phi = \phi(\tau)$, and thus ϵ denotes a finite time step. The transition probability from state ϕ to ϕ' is given by

$$W(\phi \to \phi') = \frac{1}{\sqrt{2\epsilon}} \varphi \left(\frac{\phi' - \phi}{\sqrt{2\epsilon}} + \sqrt{\frac{\epsilon}{2}} \frac{\delta S(\phi)}{\delta \phi} \right) , \tag{8.2}$$

where

$$\varphi(\eta) = \frac{1}{\sqrt{2\pi}} \exp\left(-\eta^2/2\right) \tag{8.3}$$

is a normalized Gaussian distribution. $\varphi(\eta)$ is the probability with which a value η of the noise is drawn and thus ϕ is updated to ϕ' according to the relation

$$\eta = \frac{\phi' - \phi}{\sqrt{2\epsilon}} + \sqrt{\frac{\epsilon}{2}} \frac{\delta S(\phi)}{\delta \phi} . \tag{8.4}$$

As part of the *sign problem*, an accept/reject step is not possible for this transition probability if the action $S(\phi)$ is complex. We will show that it is possible to resolve this sampling problem by two mathematical tricks.

In the first step, an additional variable ϕ'_y is introduced by choosing ϕ' to be complex-valued,

$$\phi' \to \phi'_x + i\phi'_y . \tag{8.5}$$

The imaginary part ϕ'_y of the field has no physical meaning. The field ϕ' now lives in two dimensions that are spanned by its real and imaginary parts. The field $\phi \to \phi_x + i\phi_y$ will also be complex after the first update step. The argument of the Gaussian transition probability (8.2) can be written, in terms of real and imaginary parts, as

$$\frac{\phi'_x + i\phi'_y - \phi_x - i\phi_y}{\sqrt{2\epsilon}} + \sqrt{\frac{\epsilon}{2}} \left(\frac{\delta S_{\mathrm{Re}}}{\delta \phi_x} + i \frac{\delta S_{\mathrm{Im}}}{\delta \phi_x} \right) , \tag{8.6}$$

where we define $S_{\mathrm{Re}} := S_{\mathrm{Re}}(\phi_x + i\phi_y)$ and $S_{\mathrm{Im}} := S_{\mathrm{Im}}(\phi_x + i\phi_y)$. Here, we write the functional derivative of $S(\phi_x + i\phi_y)$ with respect to the physical field variable ϕ_x since our initial field ϕ is identified with ϕ_x, whereas ϕ_y represents an additional variable. This in concordance with the introduction of a complex field in Sec. 7.1.2 and Sec. 7.2.3.

The second important step is to choose the free variable ϕ'_y in such a way that it compensates all imaginary contributions in the argument (8.6) in the transition probability, which arises from the imaginary part of the action. This is accomplished by setting

$$\phi'_y = \phi_y - \epsilon \frac{\delta S_{\mathrm{Im}}}{\delta \phi_x} . \tag{8.7}$$

As a result of this, despite a complex action S, the function φ in the transition probability has a real argument and thus represents a valid probability distribution for the Langevin update, cf. Eq. (8.2). Sampling from this distribution is achieved by the Langevin update rule (8.1) for ϕ_x,

$$\phi'_x = \phi_x - \epsilon \frac{\delta S_{\mathrm{Re}}}{\delta \phi_x} + \sqrt{2\epsilon}\eta . \tag{8.8}$$

The update equations (8.7) and (8.8) are equivalent to the discretized update rules of complex Langevin dynamics (7.24) since, for holomorphic actions,

$$\frac{\delta S_{\mathrm{Re}}}{\delta \phi_x} = \mathrm{Re}\left[\frac{\delta S}{\delta \phi}\bigg|_{\phi_x + i\phi_y}\right], \qquad \frac{\delta S_{\mathrm{Im}}}{\delta \phi_x} = \mathrm{Im}\left[\frac{\delta S}{\delta \phi}\bigg|_{\phi_x + i\phi_y}\right]. \tag{8.9}$$

8.2 Systematic derivation

We continue with a generalization of this derivation that permits the usage of different kinds of proposal distributions. The systematic step-by-step approach provides transition probabilities T and g for the visible and hidden variables. These are constructed to satisfy the constraints of a substitution algorithm that were pointed out at the beginning of Sec. 7.3. This is achieved by aiming at an implementation of the Langevin symmetry (7.36) and of the adapted detailed-balance equation (7.37).

The systematic approach starts with the definition of a standard Monte Carlo algorithm and continues with an application to a complex action. Accordingly, the derivation follows, in contrast to the previous derivation, the directions of the red arrows in Fig. 7.1.

Similar to complex Langevin dynamics, the necessary constraints are fulfilled, by definition, only in the limit of infinitesimally small step sizes in configuration space. A reliable estimation of observables is only feasible for an extrapolation to infinitesimally small step sizes.

For comparison, we state, in parallel to the general approach, the specific equations for the case of complex Langevin. The different steps of the derivation are sketched, for the case of complex Langevin, in Fig. 8.1. For the more general derivation, we keep the notation in terms of visible and hidden variables. For complex Langevin dynamics these correspond to the real and imaginary parts of the field. The field ϕ is represented by x.

Our derivation consists of the following steps: We start with a given proposal distribution and acceptance probability for a Markov chain Monte Carlo algorithm. Next, the representation of the state as well as these distributions are extended by auxiliary dimensions based on the substitution $x = v + w$. Following the provided theoretical framework for the substitution sampling algorithm, the dynamics is extended to take place in both the visible variables v and the hidden variables w. This introduces the constraints no. 1 to no. 4 on the algorithm to be taken into account, as defined in Sec. 7.3.1. An identification of symmetric and non-symmetric terms with respect to an exchange of v' and v will allow defining transition probabilities $g(w'|v', v, w)$ that satisfy the Langevin symmetry (7.36).

8.2.1 Setting up a Markov chain Monte Carlo algorithm

In a Markov chain Monte Carlo (MCMC) algorithm, a new state x' is proposed according to a distribution $q(x \to x')$ for a given state x. We restrict ourselves to symmetric proposal distributions

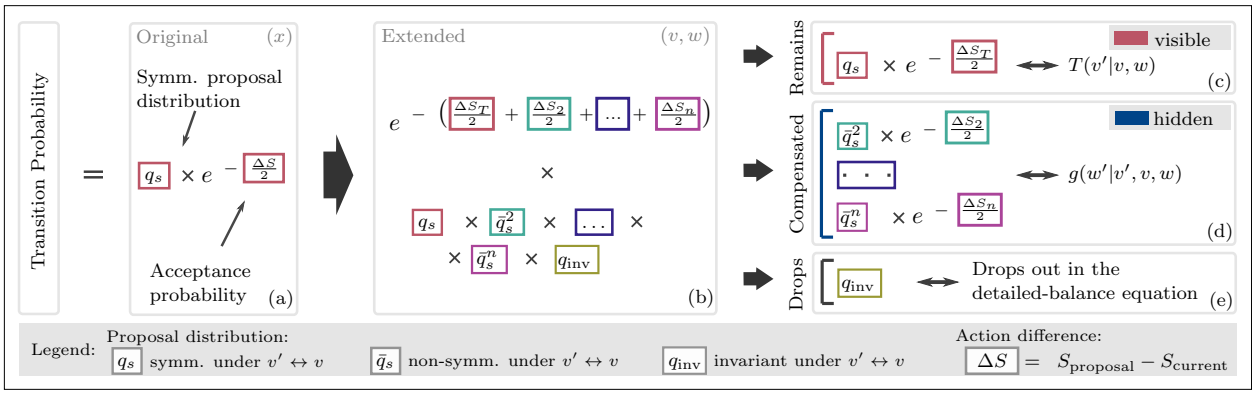

Figure 8.1: Step by step illustration of the systematic derivation of a Langevin sampling by compensation algorithm in Sec. 8.2. We consider an initial transition probability in the original representation space that can be written as product of a symmetric proposal distribution for x' and an acceptance probability that depends on the change in the action, cf. Eqs. (8.10) and (8.13). After a transition to the extended representation space defined in (v, w) (see Eq. (8.14)), symmetric, non-symmetric and invariant parts of the proposal distribution can be extracted. The scheme illustrates the case where the proposal distribution can be decomposed into a product of symmetric and non-symmetric terms, cf. Eq. (8.17). This kind of decomposition is also used for a derivation of complex Langevin dynamics. In the next step, different action contributions, defined in Eq. (8.20), are assigned to the different terms of the proposal distribution. This matching allows a definition of the transition probabilities for the visible and the hidden variables as illustrated on the right-hand side. The update of the hidden variables is based on the idea to utilise the updated hidden variables w' to compensate the associated action contributions, cf. Eqs. (8.29) and (8.30). Furthermore, the invariant term in the proposal distribution drops out in the adapted detailed-balance equation (8.15).

with

$$q(x \to x') = q(x' \to x)\,. \tag{8.10}$$

It will turn out that the adapted detailed-balance equation (7.37) can be satisfied for this algorithm only in the limit of infinitesimally small differences between the proposed state and the current state. The proposal distribution is constrained by this restriction. Hence, representations of the delta-distribution are applicable proposal distributions under these conditions. Recall that for complex Langevin dynamics, the proposal distribution is a Gaussian distribution,

$$q(\phi \to \phi') = \frac{1}{\sqrt{2\epsilon}}\varphi\left(\frac{\phi' - \phi}{\sqrt{2\epsilon}}\right)\,. \tag{8.11}$$

The transition probability $W(x \to x')$ for $x \to x'$ is commonly expressed as a product of the proposal probability $q(x \to x')$ and an acceptance probability $A(x \to x')$,

$$W(x \to x') = q(x \to x')\,A(x \to x')\,, \tag{8.12}$$

see step (a) in Fig. 8.1. If the acceptance probability is written in the exponential form

$$A(x \to x') \propto \exp\left(-\frac{\Delta S(x', x),}{2}\right)\,, \tag{8.13}$$

with $\Delta S(x', x) = S(x') - S(x)$, the resulting transition probability W satisfies the detailed-balance equation (2.15), with $\rho(x) = Z^{-1}\exp\left(-S(x)\right)$. This is the standard procedure in any Metropolis-Hastings algorithm.

8.2.2 Extending the representation space

In the following we extend the procedure to a higher-dimensional representation space. This is achieved by the substitution $x = v + w$, where the higher-dimensional space is spanned by the set (v, w) of visible and hidden variables and where v has the same dimension as x. The purpose of this is to use the state variables in the resulting auxiliary dimensions to compensate certain contributions of the action, as shown below. For the example of a complex action, we define $\phi = \phi_x + i\phi_y$ and aim to compensate the imaginary contribution of the action by the imaginary part of the field.

Next, we replace x in the steady-state distribution and in the transition probability by its higher-dimensional representation

$$\begin{aligned} x &\to v + w\,, \\ \rho(x) &\to \rho(v + w) =: p(v, w)\,, \\ q(x \to x') &\to q(v, w \to v', w')\,, \\ A(x \to x') &\to A(v, w \to v', w')\,, \end{aligned} \tag{8.14}$$

as is also indicated in step (b) in Fig. 8.1. The resulting distributions in general do not satisfy the constraints a substitution algorithm is subject to. It may, in practice, be impossible to sample from a given proposal distribution and to evaluate the acceptance probability of a proposed state. This is, for example, the case for complex Langevin dynamics, where the action is complex and thus w becomes imaginary. Accordingly, all the distributions are complex and represent no longer

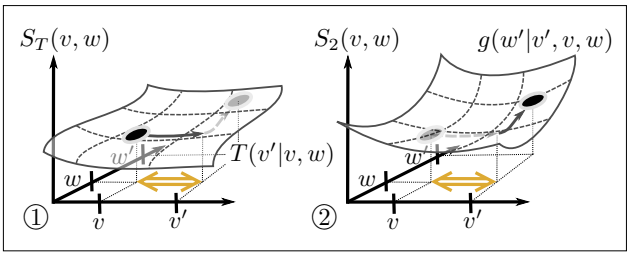

Figure 8.2: Dependence of the action on the transition probabilities for the Langevin sampling by compensation algorithm. Only the first part of the update step on the left-hand side is stochastic. The transition probability $T(v'|v,w)$ depends on the action difference $S_T(v',w) - S_T(v,w)$. In contrast, the second part of the update step is deterministic. The updated hidden variable w' is determined by $g(w'|v',v,w)$ and depends on the action difference $S_2(v',w) - S_2(v,w)$. It is important to note that in both cases w is fixed and the action difference is calculated with respect to a change in the visible variable v. This is emphasized by the golden double arrow in both illustrations.

probability distributions. However, as in the special case of complex Langevin dynamics, there is a way around these problems that allows defining transition probabilities $g(w'|v',v,w)$ and $T(v'|v,w)$.

8.2.3 The acceptance probability

We start by considering the acceptance probability in the higher-dimensional space. Based on the substitutions in Eq. (8.14), it determines the likelihood of a proposed state (v',w'). This implies a change in both, v and w. However, we aim to define a transition probability $T(v'|v,w)$ that ensures that in the long-time limit the adapted detailed-balance equation (7.37) is fulfilled, by satisfying

$$p(v,w)\,T(v'|v,w) = p(v',w)\,T(v|v',w)\,. \tag{8.15}$$

Note that the steady-state distribution p is evaluated on both sides of the equation at the same hidden state w. Hence, also the acceptance probability needs to account for changes in v only. Therefore, we define

$$A(v \to v'|w) \propto \exp\left(-\frac{S(v',w) - S(v,w)}{2}\right)\,, \tag{8.16}$$

where $S(v,w) := S(v+w)$. This choice reflects the property of the substitution sampling algorithm to incorporate a (dominant) stochastic contribution only into the direction of the visible variables, cf. Sec. 7.3.3. In the case of complex Langevin dynamics, the imaginary part ϕ_y of the field is kept constant and a change of the real part reflects the expectation value with respect to the original field ϕ.

We want to construct the transition probability T in v as a product of a proposal distribution and an acceptance probability. The acceptance probability (8.16) already satisfies the adapted detailed-balance equation (8.15) for a given transition $v \to v'$ as long as (i) the proposal distribution is symmetric under an exchange of v' and v and (ii) the transition probabilities refer to the same hidden variable w as starting point for the next update. The latter condition is depicted in Fig. 7.4 and by the golden double arrow in the first part of the update step in Fig. 8.2.

8.2.4 Symmetries

A suitable proposal distribution for T as well as a definition for the transition probability g are derived in the following by a distinction of symmetric and non-symmetric terms in the higher-dimensional distributions in Eq. (8.14). The procedure is also sketched in part (b) in Fig. 8.1.

We distinguish different terms in the proposal distribution $q(v, w \to v', w')$. Terms that are symmetric under an exchange of v' and v are denoted as q_s whereas non-symmetric terms are referred to as $\bar{q}_s$. Factors that do not depend on the visible variables are denoted as q_{inv}. The actual relation between these terms depends on the proposal distribution. For example, the terms form a product for a Gaussian proposal distribution,

$$q(v, w \to v', w') = q_s \times \bar{q}_s^2 \times \cdots \times \bar{q}_s^n \times q_{\text{inv.}} \,. \tag{8.17}$$

For complex Langevin dynamics, the following factors can be identified in the proposal distribution (8.11) after a substitution of ϕ by $\phi_x + i\phi_y$:

$$\begin{aligned} q_s &= \frac{1}{\sqrt{4\pi\epsilon}} \exp\left(-\frac{(\phi'_x - \phi_x)^2}{2\epsilon}\right) , \\ \bar{q}_s^2 &= \exp\left(-\frac{i}{2}\left[\frac{(\phi'_x - \phi_x)(\phi'_y - \phi_y)}{\epsilon}\right]\right) , \\ q_{\text{inv}} &= \exp\left(\frac{(\phi'_y - \phi_y)^2}{2\epsilon}\right) . \end{aligned} \tag{8.18}$$

In the case of the complex hat function algorithm, presented in App. C.3, the proposal distribution can be expressed as a sum

$$q(v, w \to v', w') = q_s + \bar{q}_s \,. \tag{8.19}$$

The total number of terms depends on the number of auxiliary variables.

Recall that we want to define the transition probability as a product of a proposal distribution and the acceptance probability (8.16). Keeping this in mind, the following findings are an important result of the above distinction.

On the one hand, the non-symmetric terms need to vanish in the proposal distribution for a fulfilment of the detailed-balance equation (8.15), at least in the statistical mean. On the other hand, we want to compensate certain contributions, such as the imaginary ones in the case of complex Langevin dynamics, in the action difference in the acceptance probability (8.16) that make it otherwise infeasible to sample. This is the main motivation of the entire approach.

8.2.5 Deriving $T(v'|v, w)$

We prepare the desired compensation of certain action terms by a decomposition of the acceptance probability into symmetric and non-symmetric terms and by matching these with the terms of the proposal distribution. First, we decompose the action $S(v, w)$ into n terms,

$$S(v, w) = S_T(v, w) + S_2(v, w) + \ldots + S_n(v, w) \,, \tag{8.20}$$

where S_T is used to the define the transition probability T. The terms $S_2, \ldots, S_n$ will be compensated by use of the hidden variables w'.

For a complex action the above corresponds to a separation of the real and imaginary parts. We define, for this case,

$$\begin{aligned} S_T(\phi_x, \phi_y) &= S_{\mathrm{Re}}(\phi_x + i\phi_y)\,, \\ S_2(\phi_x, \phi_y) &= iS_{\mathrm{Im}}(\phi_x + i\phi_y)\,. \end{aligned} \tag{8.21}$$

We associate the real part of the action with the update of ϕ_x and the imaginary part with that of ϕ_y.

Next, we analogously decompose the acceptance probability. The actual decomposition is dictated by the form of the proposal distribution. In the case of the product (8.17), one defines

$$A(v \to v'|w) \propto \exp\left(-\frac{S_T(v', w) - S_T(v, w)}{2}\right) \times \bar{A}_s^2 \times \cdots \times \bar{A}_s^n\,. \tag{8.22}$$

For a sum, such as Eq. (8.19), a possible decomposition is

$$A(v \to v'|w) \propto \exp\left(-\frac{S_T(v', w) - S_T(v, w)}{2}\right) \times \left[A_s^2 + \bar{A}_s^2 + \ldots + A_s^n + \bar{A}_s^n\right]\,. \tag{8.23}$$

As derived before, a change in the action is only considered in the visible direction. At this point, it is sufficient to focus on the symmetric and non-symmetric terms, A_s^i and $\bar{A}_s^i$. This allows, in the following step, a definition of T and g.

For our example of a complex action, the mathematical operation is a product and the non-symmetric term $\bar{A}_s^2$ is given by

$$\bar{A}_s^2 = \exp\left(-i\,\frac{S_{\mathrm{Im}}(\phi'_x + i\phi_y) - S_{\mathrm{Im}}(\phi_x + i\phi_y)}{2}\right)\,. \tag{8.24}$$

We continue by considering the product of the decomposed proposal and acceptance probabilities, namely:

$$q(v, w \to v', w') \times A(v \to v'|w)\,, \tag{8.25}$$

collecting all terms symmetric with respect to an exchange of v' and v, to define the transition probability T. For example, for the product form (8.17), the transition probability is defined as

$$T(v'|v, w) \propto q_s \times \exp\left(-\frac{S_T(v', w) - S_T(v, w)}{2}\right)\,. \tag{8.26}$$

The right-hand side consists of a product of the symmetric term q_s in Eq. (8.17) and of the first factor of the acceptance probability in Eq. (8.22). For the example of complex Langevin dynamics, this combination of symmetric terms is shown in step (c) in Fig. 8.1. In this case, the transition probability for the real part of the field, with the Gaussian $q_s \sim \varphi$, reads

$$T(\phi'_x|\phi_x, \phi_y) \propto \frac{1}{\sqrt{2\epsilon}}\varphi\left(\frac{\phi'_x - \phi_x}{\sqrt{2\epsilon}}\right) \exp\left(-\frac{\Delta S_{\mathrm{Re}}(\phi', \phi)}{2}\right)\,, \tag{8.27}$$

where $\Delta S_{\mathrm{Re}}(\phi', \phi) = S_{\mathrm{Re}}(\phi'_x + i\phi_y) - S_{\mathrm{Re}}(\phi_x + i\phi_y)$.

We will study, in Sec. 8.3, under which conditions the transition probability satisfies the adapted detailed-balance equation (8.15).

8.2.6 Deriving $g(w'|v', v, w)$

It remains to determine a transition probability $g(w'|v', v, w)$, which satisfies the Langevin symmetry (7.36),

$$g(w'|v', v, w) \stackrel{!}{=} g(w'|v, v', w)\,, \tag{8.28}$$

as suggested for a substitution sampling algorithm, cf. Sec. 7.3.3.

The above distinction of symmetric and non-symmetric terms allows determining the transition probability g by compensating the remaining terms in the above discussed product of the proposal distribution and the acceptance probability. More specifically, all terms of the product (8.25) that do not contribute to the transition probability (8.26) are supposed to cancel each other, for which we will use the updated hidden variables w'.

Considering first again the case of the product form (8.17) of the proposal distribution, this translates into

$$\begin{aligned} &\bar{q}_s^2 \times \cdots \times \bar{q}_s^n \times \bar{A}_s^2 \times \cdots \times \bar{A}_s^n \stackrel{!}{=} 1 \\ \Leftrightarrow \quad & w' - h(v', v, w) \stackrel{!}{=} 0\,, \end{aligned} \tag{8.29}$$

with

$$g(w'|v', v, w) = \delta\left(w' - h(v', w, w)\right)\,. \tag{8.30}$$

The matching of the remaining terms is illustrated in step (d) in Fig. 8.1. Following Sec. 7.3.3, the function $h(v', v, w)$ defines the updated value of w'. The invariant term q_{inv} has been neglected as it can be cancelled in the adapted detailed-balance equation.

This is always possible since the updated state w' can be chosen arbitrarily as long as the update rule satisfies the Langevin symmetry. As a result of the symmetric properties of the remaining terms, the resulting transition probability indeed bears this symmetry.

In the case of complex Langevin, Eq. (8.29) can be simplified to

$$\frac{(\phi'_x - \phi_x)(\phi'_y - \phi_y)}{\epsilon} + \Delta S_{\text{Im}}(\phi', \phi) \stackrel{!}{=} 0\,, \tag{8.31}$$

with $\Delta S_{\text{Im}}(\phi', \phi) = S_{\text{Im}}(\phi'_x + i\phi_y) - S_{\text{Im}}(\phi_x + i\phi_y)$. Consequently, the update rule for the hidden state is

$$\phi'_y = \phi_y - \epsilon \frac{\Delta S_{\text{Im}}(\phi', \phi)}{\phi'_x - \phi_x}\,. \tag{8.32}$$

As intended, the updated imaginary part of the field compensates imaginary contributions arising in the product (8.25) of the proposal distribution and the acceptance probability. The compensation has the same effect as in complex Langevin dynamics in the previous section, namely, resulting in a real-valued transition probability $T(\phi'_x|\phi_x, \phi_y)$ for the real part of the field.

The compensation is either exact or satisfied in a stochastic way through $h(v', v, w)$. An example for a stochastic update of the hidden variable w is given by complex Langevin with imaginary noise, see,

for example, Ref. [219]. Thereby, it is however important that the stochastic behaviour in the visible direction is dominant. This restriction is reflected by the constraints on a substitution sampling algorithm, defined in Sec. 7.3.1.

8.3 Implications

The derived transition probabilities do not yet satisfy all of the constraints a substitution sampling algorithm is subject to. In the following, we derive further restrictions which ensure this, analogous to the discussion for complex Langevin in Sec. 7.3.2.

The adapted detailed-balance equation (8.15) is violated for the transition probabilities $T(v'|v,w)$, resulting in

$$p(v,w)T(v'|v,w) = p(v',w)T(v|v',w)\exp\left(-\sum_{i=2}^{n}(S_i(v,w)-S_i(v',w))\right), \tag{8.33}$$

where $p(v,w)=\rho(v+w)$. This is the same discrepancy as for the transition probability $T(\phi'_x|\phi_x,\phi_y)$ of complex Langevin in Sec. 7.3.2, cf. Eq. (7.31). It can be traced back to the restriction to the terms S_T in the action difference, cf. Eqs. (8.20) and (8.26).

The discrepancy can be resolved by imposing

$$\exp\left(-\sum_{i=2}^{n}(S_i(v,w)-S_i(v',w))\right) \stackrel{!}{=} 1\,. \tag{8.34}$$

This can be reached with infinitesimal stepping in updating the visible variable v. The proposal distribution needs to allow implementing this limit. As pointed out previously, representations of delta-distributions are examples for appropriate proposal distributions. Since an infinitesimally small sampling step is not meaningful algorithmically, we resort to an extrapolation towards zero step size.

We conclude that the restriction to an infinitesimal step size in the visible direction entails a satisfaction of the adapted detailed-balance equation, cf. Eqs. (7.37) and (8.15). Recalling that the transition probability g of the hidden variables implements the Langevin symmetry by construction, we find that constraint no. 1 for a substitution sampling algorithm is fulfilled.

Constraint no. 2 requires that the step size in the direction of the hidden variables is infinitesimal. As the transition probabilities T and g are derived from the same proposal distribution, the step size of the hidden variables is already reduced simultaneously with the one in the visible direction.

Constraint no. 3 depends on the considered model.

It remains an analysis of constraint no. 4. It is not possible to show that this is generally fulfilled for arbitrary proposal distributions. For the case of complex Langevin dynamics it is proven in Sec. 7.3.2. We assume that the proof is also valid for other proposal distributions as long as these coincide in the limit of infinitesimally small step sizes with a delta-distribution. This assumption is supported by the numerical results in Sec. 7.6.

Keeping this in mind, the restrictions on g, in constructing substitution sampling algorithms, cf. Eq. (7.43), can be relaxed to

$$g(w'|v', v, w) = \delta\left(w' - h(v', v, w; \epsilon)\right) , \tag{8.35}$$

where the parameter ϵ and the function h ensure, as before, an infinitesimal step size in the hidden direction:

$$\lim_{\epsilon \to 0} h(v', v, w; \epsilon) = w \, . \tag{8.36}$$

More specifically, we reinserted a dependence of the transition probability on the updated visible state v'. This relaxation is, for example, utilized in the complex hat function algorithm in Sec. 7.4.2.

The derivation of the discretized update equations for complex Langevin dynamics is completed in App. C.1.

8.4 Measure for accuracy

In the previous section, it has been shown that the detailed-balance equation is violated for simulations with a finite step size in the visible states v. One can define a measure κ for the accuracy of the Langevin sampling by compensation algorithm based on Eq. (8.34),

$$\kappa(v', v, w) = \left| \sum_{i=2}^{n} S_i(v', w) - \sum_{i=2}^{n} S_i(v, w) \right| . \tag{8.37}$$

It measures the violation of the detailed-balance equation in dependence on the step size in v. A simulation satisfies the detailed-balance equation if $\kappa(v', v, w) = 0$. Our numerical results in Sec. 7.6 confirm that κ represents a reasonable measure in analysing the Langevin sampling by compensation algorithm for finite step sizes.

The measure is in accordance with an improved numerical stability of complex Langevin dynamics by introducing an adaptive step size [79, 120]. This adaptation prevents too large step sizes, leading to small measures of $\kappa(v', v, w)$.

CHAPTER 9

Self-consistent sampling of complex actions

This chapter refers to Refs. [8, 9].

The here presented *self-consistent sampling algorithm* aims to solve two main problems of complex Langevin dynamics: A convergence to unphysical fixed points and numerical instabilities, cf. Chapter 1 and Sec. 2.1. The herein presented approach concentrates again on a successful computation of observables with a complex action $S(\phi)$, cf. Eq. (1.1).

It is inspired by a sequential application of a special form of reweighting and complex Langevin dynamics. A successive application of both methods allows the exploration of various complex action problems. The idea is to use these two methods to mutually verify the correctness of computed expectation values. We will show that this can be translated into a new, model-independent criterion for correctness of complex Langevin dynamics, cf. Sec. 9.3. Furthermore, the criterion allows to derive an optimization algorithm that converges to the physically correct solutions, cf. Sec. 9.4. The self-consistent sampling process enforces correctness based on an educated training of a stabilizing potential within the Langevin sampling process, resulting in a teacher-student training algorithm. The student is represented by a stabilized complex Langevin dynamics and the teacher by an adapted form of reweighting in the complex plane.

9.1 Standard reweighting

Monte Carlo samples, drawn from a Boltzmann distribution $Z^{-1}\exp(-S(\phi))$, are used in standard reweighting to compute expectation values of a target distribution $\tilde{Z}^{-1}\exp(-\tilde{S}(\phi))$ [296]. The original distribution of samples is reweighted to the target distribution. The reweighting formula can be derived by inserting a one into the originally considered expectation value,

$$\begin{aligned}\langle\mathcal{O}(\phi)\rangle_S &= \frac{1}{Z}\int \mathcal{D}\phi\,\mathcal{O}(\phi)e^{-S(\phi)} \\ &= \frac{\tilde{Z}}{Z}\frac{1}{\tilde{Z}}\int \mathcal{D}\phi\,\mathcal{O}(\phi)e^{-(S(\phi)-\tilde{S}(\phi))}e^{-\tilde{S}(\phi)} \\ &= \frac{\tilde{Z}}{Z}\langle\mathcal{O}(\phi)e^{-(S(\phi)-\tilde{S}(\phi))}\rangle_{\tilde{S}}\,. \end{aligned} \tag{9.1}$$

The fraction $\frac{\tilde{Z}}{Z}$ remains to be determined. By using $\langle 1 \rangle_S = 1$,

$$\langle 1 \rangle_S = 1 \stackrel{!}{=} \frac{\tilde{Z}}{Z} \langle e^{-(S(\phi)-\tilde{S}(\phi))} \rangle_{\tilde{S}} \,, \tag{9.2}$$

one finds

$$\frac{\tilde{Z}}{Z} = \frac{1}{\langle e^{-(S(\phi)-\tilde{S}(\phi))} \rangle_{\tilde{S}}} \,. \tag{9.3}$$

Finally, this results in the following formula for reweighting:

$$\langle \mathcal{O}(\phi) \rangle_S = \frac{\langle \mathcal{O}(\phi) e^{-(S(\phi)-\tilde{S}(\phi))} \rangle_{\tilde{S}}}{\langle e^{-(S(\phi)-\tilde{S}(\phi))} \rangle_{\tilde{S}}} \,. \tag{9.4}$$

Reweighting only works if the original distribution and the target distribution have a sufficient overlap. The statistical error of the computed expectation value increases with a decreasing overlap. The reason for that is a missing support of samples with a high Boltzmann weight in the reweighted distribution. This problem is also referred to as *overlap problem* [297]. In QCD, it prohibits a full exploration of the QCD phase diagram. For problems with a sign problem, the method is only applicable if the complex phase in the denominator, introduced in QCD by a finite chemical potential, is sufficiently small.

9.2 Reweighting in the complex plane

As proposed in Refs. [295, 298], it is possible to combine complex Langevin dynamics and reweighting. In this approach, complex Langevin dynamics is used to generate samples $\phi_{x,i} + i\phi_{y,i}$ of a distribution $Z^{-1} \exp(-S(\phi))$ in the complex plane. The fields ϕ turn in this kind of sampling process into complex fields and expectation values can be numerically accessed by

$$\langle \mathcal{O}(\phi) \rangle_\rho = \langle \mathcal{O}(\phi_x + i\phi_y) \rangle_{P(\phi_x,\phi_y)} = \frac{1}{N} \sum_i^N \mathcal{O}(\phi_{x,i} + i\phi_{y,i}) \,, \tag{9.5}$$

where the probability distribution $P(\phi_x, \phi_y)$ is defined as the distribution over ϕ_x and ϕ_y in the complex plane, see also Sec. 2.1.

The standard reweighting formula can be also applied for reweighting sampled distributions $P(\phi_x, \phi_y)$ in the complex plane. One finds

$$\langle \mathcal{O}(\phi_x + i\phi_y) \rangle_P = \frac{\langle \mathcal{O}(\phi_x + i\phi_y) e^{-(S(\phi_x+i\phi_y)-\tilde{S}(\phi_x+i\phi_y))} \rangle_{\tilde{P}}}{\langle e^{-(S(\phi_x+i\phi_y)-\tilde{S}(\phi_x+i\phi_y))} \rangle_{\tilde{P}}} \,, \tag{9.6}$$

by making use of the first identity in Eq. (9.5) and by proceeding in the same way as for standard reweighting. This kind of reweighting exhibits the same properties as standard reweighting. In particular, the overlap of the respective distributions determines the statistical error of the computed expectation values.

9.3 Step-wise reweighting criterion for correctness

With the reweighting in the complex plane approach and complex Langevin dynamics, we have two methods to compute expectation values for a given target distribution $\tilde{Z}^{-1}\exp(-\tilde{S}(\phi))$. This offers the advantage of allowing a verification of computed expectation values using the respectively other method.

We exploit this in the so-called step-wise reweighting criterion for correctness, which has also been studied in Ref. [299]. This criterion represents a new method to verify the correctness of complex Langevin dynamics, which we present based on the example of the polynomial model with an imaginary external field,

$$S(\phi;\lambda,\sigma,h_{\mathrm{Im}}) = \lambda\phi^4 + \sigma\phi^2 + ih_{\mathrm{Im}}\phi\,, \tag{9.7}$$

with $\lambda,\sigma,h_{\mathrm{Im}} \in \mathbb{R}$. The goal is to verify if complex Langevin dynamics converges to the correct expectation values for a target action $\tilde{S} := S(\phi;\lambda,\sigma,\tilde{h}_{\mathrm{Im}})$. In a first step, we assume that there exists an action $S(\phi)$ where it is known that sampling results in the correct solutions. In our example, this is given by $S(\phi;\lambda,\sigma,h_{\mathrm{Im}}=0)$. The action has no sign problem and can be sampled via a standard Monte Carlo approach.

Next, we define a smooth line in the action parameter space $\theta := \{\lambda,\sigma,h_{\mathrm{Im}}\}$ connecting the parameters of the initial action and of the target action. For the action in Eq. (9.7), this line can be parametrized, for example, by

$$h_{\mathrm{Im}}(\alpha) = \alpha h_{\mathrm{Im}}\,, \tag{9.8}$$

with $\theta(\alpha) := \{\lambda,\sigma,h_{\mathrm{Im}}(\alpha)\}$ and $\alpha \in [0,1]$, allowing a smooth transition between the initial and the target action.

To mitigate the overlap problem, we define a grid

$$\alpha = (\alpha_1 = 0,\ldots,\alpha_k,\alpha_{k+1},\ldots,\alpha_n = 1) \leftrightarrow h_{\mathrm{Im}}(\alpha) = (0,\ldots,...,h_{\mathrm{Im}}^k,h_{\mathrm{Im}}^{k+1},\ldots,\tilde{h}_{\mathrm{Im}}) \tag{9.9}$$

that is supposed to ensure a sufficient overlap between the distributions of two successive values α_k and α_{k+1}. The approach is illustrated also in Fig. 9.2a.

The criterion for correctness is built upon simulations at the different values α_k and a subsequent verification of the computed expectation values by reweighting in the complex plane. A divergent behaviour of the expectation values indicates that at least one of the methods fails. In the case of complex Langevin dynamics, such a failure can be caused, for example, by a convergence of the dynamics to unphysical fixed points. For reweighting, an already divergent source distribution enforces failures in the reweighted expectation values.

A respective criterion can be defined, for example, as

$$M(\alpha_k,\alpha_{k+1}) = \sqrt{\sum_{j=0}^{N}\left(\langle\mathcal{O}_{\mathrm{source}}\left(\phi_x+i\phi_y;\theta\left(\alpha_{k,j}\right)\right)\rangle_{P_{\alpha_k}} - \langle\mathcal{O}_{\mathrm{target}}\left(\phi_x+i\phi_y;\theta\left(\alpha_{k,j}\right)\right)\rangle_{P_{\alpha_{k+1}}}\right)^2}\,, \tag{9.10}$$

with

$$\alpha_{k,j} = \alpha_k + j\frac{\alpha_{k+1}-\alpha_k}{N}\,, \tag{9.11}$$

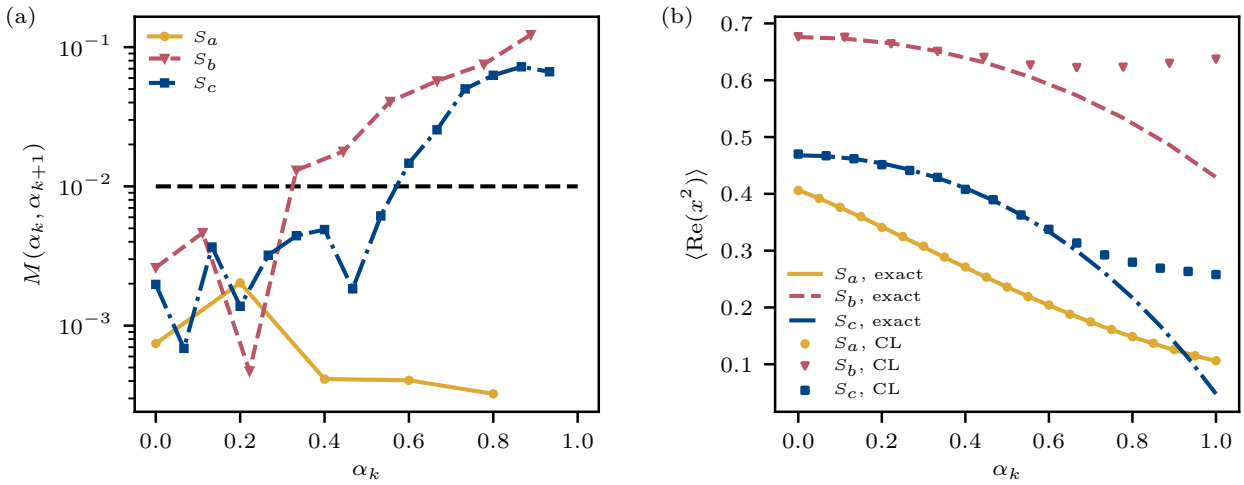

Figure 9.1: Numerical results for testing the step-wise reweighting criterion for correctness, cf. Eq. (9.10), on the example of different parametrizations of the action of the polynomial model (9.7). See Tab. 9.1, for a definition of the considered actions and their dependency on the parameter $\alpha \in [0, 1]$. (a) The step-wise reweighting criterion for correctness $M(\alpha_k, \alpha_{k+1})$ for $N = 2$ and with the real part of the second moment $\langle \phi^2 \rangle$ as considered observable. The horizontal dashed black line marks a possible threshold for an application of the criterion. Either reweighting or complex Langevin dynamics leads to wrong results for the target action parameter α_{k+1} if the criterion is above this threshold. (b) Comparison of the exact result and sampled expectation value for the real part of the second moment $\langle \phi^2 \rangle$. The observed transition to higher values of $M(\alpha_k, \alpha_{k+1})$ in part (a) correctly detect the failure of complex Langevin dynamics for the actions S_b and S_c for larger values of α. The different paths in parameter space are also depicted in Fig. 9.2a.

and N being an integer defining the number of compared reweighted expectation values between the parameters α_k and α_{k+1}. The first expectation value in the above expression

$$\langle \mathcal{O}_{\text{source}} \left(\phi_x + i\phi_y; \theta \left(\alpha_{k,j} \right) \right) \rangle_{P_{\alpha_k}} \tag{9.12}$$

denotes the expectation value with respect to the action $S(\phi; \theta(\alpha_{k,j})$ obtained by reweighting the sampled distribution of a complex Langevin process with $S(\phi; \theta(\alpha_k))$. Similarly, the expectation value

$$\langle \mathcal{O}_{\text{target}} \left(\phi_x + i\phi_y; \theta \left(\alpha_{k,j} \right) \right) \rangle_{P_{\alpha_{k+1}}} \tag{9.13}$$

refers to the reweighted observable from a simulation based on the action $S(\phi; \theta(\alpha_{k+1}))$. If the expectation values for the different sets of parameters coincide, the criterion $M(\alpha_k, \alpha_{k+1})$ vanishes and complex Langevin dynamics converges to the correct solutions. Finite values of $M(\alpha_k, \alpha_{k+1})$ indicate that either reweighting or complex Langevin dynamics leads to wrong observables. In this case, varying the step size in parameter space can help. In addition to the criterion defined Eq. 9.10, we recommend furthermore to also consider similar types of criteria, for example taking into account several simulations.

In Fig. 9.1, we apply the criterion for correctness (9.10) on different paths through the action parameters space of the polynomial model in Eq. (9.7), see Tab. 9.1 for the different chosen parametrizations. The considered paths in the space of actions are also schematically shown in Fig. 9.2a. The numerical results show that the criterion correctly uncovers regions in the parameters space where complex Langevin dynamics fails.

Name	λ	σ	h_{Im}
S_a	$1+i1$	$1+i\alpha_k \times 4$	0
S_b	1	0	$\alpha_k \times 0.9$
S_c	1	1	$\alpha_k \times 1.5$

Table 9.1: Chosen parametrizations for testing the step-wise reweighting criterion for correctness (9.10) based on the action of the polynomial model with an imaginary external field, cf. Eq. (9.7). The criterion is applied for different values of $\alpha_k \in [0,1]$. Numerical results for the different actions are shown in Fig. 9.1.

9.4 Stabilized complex Langevin dynamics

The existence of two methods for computing expectation values of complex actions can be exploited to formulate an optimization algorithm which stabilizes the dynamics of complex Langevin to ensure a correct convergence. The approach is inspired by the step-wise reweighting criterion for correctness.

We use reweighting in the complex plane to train a stabilized version of complex Langevin dynamics by minimizing deviations in the expectation values of the two methods. A successive exploration of the parameter space is feasible by applying this teacher-student training algorithm in the same step-wise manner as in the previous section. Due to the successive consistency checks and the self-supervised learning approach, we denote the algorithm as *self-consistent sampling algorithms*.

A possible approach to stabilize complex Langevin dynamics for a given action $S(\phi;\theta)$ with action parameters θ is the introduction of an auxiliary potential $\xi(\phi;\psi)$, parametrized by ψ. The stabilized process is defined by an extension of complex Langevin dynamics

$$\begin{aligned}\frac{\mathrm{d}\phi_x}{\mathrm{d}\tau} &= -\frac{\delta S_{\mathrm{Re}}(\phi;\theta)}{\delta\phi}\bigg|_{\phi_x+i\phi_y} + \eta\,,\\ \frac{\mathrm{d}\phi_y}{\mathrm{d}\tau} &= -\frac{\delta S_{\mathrm{Im}}(\phi;\theta)}{\delta\phi}\bigg|_{\phi_x+i\phi_y}\,,\end{aligned} \tag{9.14}$$

in terms of an auxiliary drift term derived from $\xi(\phi;\psi)$

$$\begin{aligned}\frac{\mathrm{d}\phi_x}{\mathrm{d}\tau} &= -\frac{\delta S_{\mathrm{Re}}(\phi;\theta)}{\delta\phi}\bigg|_{\phi_x+i\phi_y} - \frac{\delta \xi_{\mathrm{Re}}(\phi;\psi)}{\delta\phi}\bigg|_{\phi_x+i\phi_y} + \eta\,,\\ \frac{\mathrm{d}\phi_y}{\mathrm{d}\tau} &= -\frac{\delta S_{\mathrm{Im}}(\phi;\theta)}{\delta\phi}\bigg|_{\phi_x+i\phi_y} - \frac{\delta \xi_{\mathrm{Im}}(\phi;\psi)}{\delta\phi}\bigg|_{\phi_x+i\phi_y}\,.\end{aligned} \tag{9.15}$$

The auxiliary drift is expected to vanish if complex Langevin dynamics samples from the correct physical fixed points. Otherwise, the numerically computed expectation values by the dynamics (9.14) can no longer be associated with the expectation values of the given action $S(\phi;\theta)$. In particular, this implies that, cf. Eq. (9.5):

$$\langle \mathcal{O}(\phi)\rangle_\rho \neq \langle \mathcal{O}(\phi_x+i\phi_y)\rangle_{P(\phi_x,\phi_y)}, \tag{9.16}$$

i.e., the sampling process converges to wrong results. The sampled probability distribution $P(\phi_x,\phi_y)$ and therefore complex Langevin dynamics cannot be related anymore with the original distribution of interest $\rho(\phi)$, as a sampling process. The idea of self-consistent sampling is to resolve this discrepancy by introducing the auxiliary potential $\xi(\phi;\psi)$.

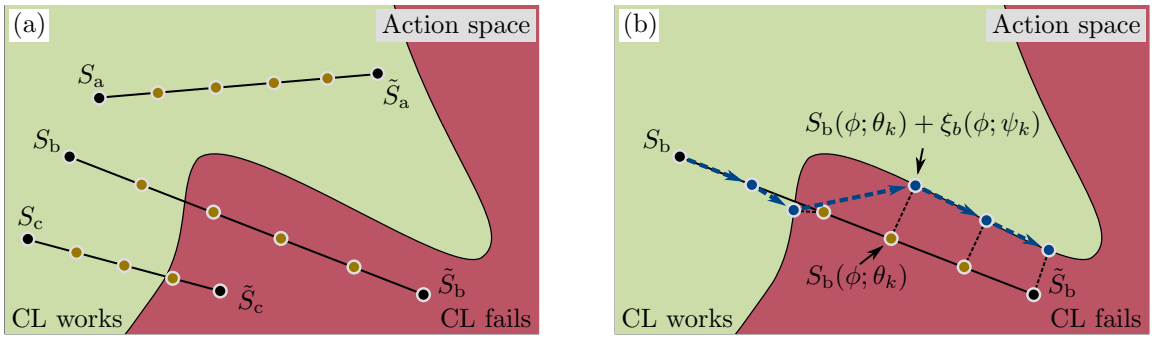

Figure 9.2: Schematic illustrations of the space of action for the polynomial model. In the green area, complex Langevin dynamics samples the correct expectation values. In the red area, the dynamics converges to unphysical fixed points, resulting in wrong expectation values. The solid straight lines represent possible paths through the space of actions by different parametrizations of the action parameters in terms of α_k, cf. Eq. (9.9) and Tab. 9.1. The step-size of changes in α_k is indicated by the blue dots on the respective lines. (a) Possible interpretation of the numerical results for the step-wise reweighting criterion for correctness in Fig. 9.1. The criterion correctly detects the failure of complex Langevin dynamics for larger values of the imaginary external field h_{Im} in the polynomial model (9.7). (b) Possible paths (in blue) for the extended actions $S(\phi;\theta) + \xi(\phi;\psi)$ of the self-consistent sampling algorithms. After convergence of the training algorithm for an auxiliary potential $\xi(\phi;\psi_k)$, the respective stabilized complex Langevin dynamics, cf. Eq. (9.15), samples in a region of the action space where the expectation values refer to the physically correct ones. The algorithms can be successfully applied as long as the overlap of the effectively sampled distribution and the target distribution $S_b(\phi;\theta_k)$ is sufficiently high with respect to a reasonable error estimation.

With respect to the inequality (9.16), there are two possible interpretations of the auxiliary potential. The first one refers to it as a stabilizer to resolve the inequality. In this case, the samples in the complex plane are supposed to be distributed according to a probability distribution $\tilde{P}(\phi_x, \phi_y)$ leading, based on Eq. (9.5), to the correct expectation values of $S(\phi;\theta)$. Accordingly, the dynamics in Eq. (9.15) can be related to the distribution $\rho(\phi) \propto \exp(-S(\phi;\theta))$,

$$\langle \mathcal{O}(\phi) \rangle_\rho = \langle \mathcal{O}(\phi_x + i\phi_y) \rangle_{\tilde{P}}. \tag{9.17}$$

Note that, in this case, the auxiliary drift term is a pure stabilizer of the dynamics resolving the failure of complex Langevin dynamics.

The second interpretation assumes that the stabilizing potential allows a correct identification of the original formulation of complex Langevin dynamics (9.15), implemented in terms of the extended action $S(\phi;\theta) + \xi(\phi;\psi)$, with the distribution $\rho_{S+\xi}(\phi) \propto \exp(-(S(\phi;\theta) + \xi(\phi;\psi)))$. In particular, this entails that it holds, in contrast to the first interpretation,

$$\langle \mathcal{O}(\phi) \rangle_{\rho_{S+\xi}} = \langle \mathcal{O}(\phi_x + i\phi_y) \rangle_{P_{S+\xi}(\phi_x,\phi_y)}. \tag{9.18}$$

Expectation values of the action $S(\phi;\theta)$ can be extracted subsequently by reweighting in the complex plane,

$$\begin{aligned} \langle \mathcal{O}(\phi) \rangle_{\text{stabilized CLE}} &:= \langle \mathcal{O}(\phi) \rangle_\rho = \langle \mathcal{O}(\phi_x + i\phi_y) \rangle_P = \\ &= \frac{\langle \mathcal{O}(\phi_x + i\phi_y) e^{\xi(\phi_x + i\phi_y;\psi)} \rangle_{P_{S+\xi}(\phi_x,\phi_y)}}{\langle e^{\xi(\phi_x + i\phi_y;\psi)} \rangle_{P_{S+\xi}(\phi_x,\phi_y)}}. \end{aligned} \tag{9.19}$$

The equality in Eq. (9.18) implies that we aim to modify the original action such that complex Langevin dynamics does not suffer from a convergence to unphysical fixed point any longer. Instead, the dynamics samples the correct physical solution to the stabilized action. This property makes the definition of a loss function based on expectation values feasible. For this reason, we focus in the following on the second interpretation for utilizing an auxiliary potential. Concerning the overlap problem of reweighting, the stabilizing potential is supposed to converge within the training to a probability distribution $P_{S+\xi}(\phi_x, \phi_x)$ which is as close as possible to the correct distribution of the original action. A possible training outcome is illustrated in Fig. 9.2. We discuss implications and limitations related to a potential remaining overlap problem in more detail in Sec. 9.5 and continue with more details on the training algorithms.

We parameterize the auxiliary potential by a neural network. The neural network is trained based on a loss function derived in the following. Similar to step-wise reweigthing, a correct sampling of a target action can be realized in a step-wise manner by a successive sampling of actions between an initial and a target action in the action parameter space. In concordance with Eqs. (9.8) and (9.9), we introduce the following notation for the corresponding path in the parameter space

$$\theta(\alpha_1 = 0), \ldots, \theta(\alpha_k), \theta(\alpha_{k+1}), \ldots, \theta(\alpha_n = 1) \equiv \theta_1, \ldots, \theta_k, \theta_{k+1}, \ldots, \tilde{\theta}\,. \tag{9.20}$$

Instead of applying this for the presented criterion of correctness of complex Langevin dynamics, we aim to use it as a tool to stabilize the dynamics in case of a convergence to wrong solutions.

We assume that complex Langevin dynamics is correct for given set of parameters θ_k and an auxiliary potential $\xi(\phi; \psi_k)$. In this case, reweighting can be applied for computing expectation values of the action defined in terms of the subsequent action parameters θ_{k+1},

$$\begin{aligned}\langle \mathcal{O}(\phi)\rangle_{\text{reweighted target}} &= \langle \mathcal{O}(\phi_x + i\phi_y)\rangle_{\text{reweighted target}} = \\ &= \frac{\langle \mathcal{O}(\phi_x + i\phi_y) e^{-(S(\phi_x+i\phi_y;\theta_{k+1}) - (S(\phi_x+i\phi_y;\theta_k) + \xi(\phi_x+i\phi_y;\psi_k)))}\rangle_{P_{S+\xi}}}{\langle e^{-(S(\phi_x+i\phi_y;\theta_{k+1}) - (S(\phi_x+i\phi_y;\theta_k) + \xi(\phi_x+i\phi_y;\psi_k)))}\rangle_{P_{S+\xi}}}\,.\end{aligned} \tag{9.21}$$

In line with the step-wise reweighting criterion for correctness, the latter expectation value is assumed to be the correct one for a sufficient overlap of the support of $S(\phi;\theta_{k+1})$ and $S(\phi;\theta_k) + \xi(\phi;\psi_k)$. We expect that this can be achieved by choosing θ_k and θ_{k+1} close enough. Note that the auxiliary potential is supposed to vanish for the initial action, defined in terms of θ_1, since the respective sampling algorithm is expected to work for a successive step-wise exploration of the parameter space.

The expectation values in Eqs. (9.19) and (9.21) allow the definition of a loss function for the training of the auxiliary potential $\xi(\phi;\psi_{k+1})$,

$$\mathcal{L}(\psi_{k+1}) = \sum_j \| \langle \mathcal{O}_j(\phi)\rangle_{\text{reweighted target}} - \langle \mathcal{O}_j(\phi)\rangle_{\text{stabilized CLE}} \|_2^2 + \text{Further regularizing terms}\,. \tag{9.22}$$

The sum runs over different considered observables $\mathcal{O}_j(\phi)$ of interest. We replace θ by θ_{k+1} and ψ by ψ_{k+1} for the computation of observables with respect to the stabilized complex Langevin dynamics defined in Eq. (9.19). Within the training, the expectation values are approximated based on a batch-wise evaluation. The auxiliary potential is trained based on a minimization of the loss function with respect to the parameters ψ_{k+1}:

$$\psi^*_{k+1} = \underset{\psi_{k+1}}{\text{argmin}}\, \mathcal{L}(\psi_{k+1})\,. \tag{9.23}$$

Additional regularizing terms might be helpful to stabilize the training. In particular, a minimization of the complex phase factor of the denominator in Eq. (9.19) by

$$\mathcal{L}_{\text{phase}}(\psi_{k+1}) = \arg\left(\left\langle e^{\xi(\phi_x+i\phi_y;\psi_{k+1})}\right\rangle_{P_{S+\xi}(\phi_x,\phi_y)}\right) \tag{9.24}$$

is useful to mitigate a potential overlap problem. Further regularizing terms can help to prevent numerical instabilities.

A drawback of the approach is that the generation of samples is expensive since the samples can only be drawn if the dynamics is in equilibrium. The simulation time can be decreased by starting the evolution for updated parameters ψ_{k+1} with the samples from the last evolution. A benefit is that the loss function does not rely on the history of the complex Langevin dynamics but instead on the statistics of multiple processes. This makes backpropagation, i.e., an optimization of the parameters ψ_{k+1}, feasible. This is the reason why our approach focuses on the second interpretation for utilizing an auxiliary potential.

Putting everything together, a single training step consists of the following steps

1. Sample a set of configurations based on stabilized Langevin dynamics (9.15) with the stabilizing potential $\xi(\phi;\psi_{k+1})$.
2. Compute reweighted observables $\langle\mathcal{O}_j(\phi)\rangle_{\text{stabilized CLE}}$.
3. Compute the loss $\mathcal{L}(\phi_{k+1})$ and update ψ_{k+1} based on backpropagation.

A successful application of the training for each of the parameters θ_k results in a correct computation of observables for a given target action $S(\phi;\theta^\star)$.

9.5 Summary and future work

We introduced the step-wise reweighting criterion for correctness as an effective and easy to compute criterion for verifying whether or not complex Langevin dynamics converges to the physically correct fixed points in the complex plane. An application to more complicated models as well as a thorough comparison to existing criteria is subject to future work.

In the second part of this chapter, we proposed to stabilize the dynamics of complex Langevin by means of an auxiliary potential which is trained with the help of the novel criterion for correctness. The training algorithm enforces that the obtained expectation values are always in compliance with the utilized action of the stabilized dynamics, cf. Fig. 9.2. In inaccessible regions in the action space, the algorithm relies on a sufficient overlap of the sampled distribution and the target distribution for reweighting to work. It remains to be seen in future work and by first numerical results to which extent the algorithm is capable of resolving these areas in the action space.

CHAPTER 10

Unsupervised neural graph embedding

This chapter refers to Ref. [7].

There has been an explosion of interest surrounding the development and application of deep learning methods on graph-structured data. Graph neural networks (GNNs) [200, 300–302] have recently gained traction as a powerful machine learning building block due to their demonstrated success in a number of domains. See Refs. [197–199, 202–204, 303, 304] for comprehensive reviews of different application domains such as graph matching and graph similarity algorithms.

Generating lower-dimensional representations in a continuous domain is a prominent use case of respective machine learning and deep learning tools acting on graphs [201, 203, 305–307]. The generation of graph embeddings and the processing of graph-structured data in a deep learning framework is different and more challenging due to the discrete and non-Euclidean structure of graphs, a possibly varying number of nodes in each graph and the absence of a fixed ordering or numbering of nodes. The latter property relates to the graph isomorphism problem (also referred to as exact graph matching problem) determining whether there exists a mapping between two graphs which preserves the edge connectivity [204, 308–310]. Ideally, all graphs belonging to the same isomorphic class lead to the same continuous representation.

In general, one has to distinguish different kinds of graph embedding [203]. In the following, we refer to network embeddings if each node of a network is embedded. The goal is to correctly preserve and embed important properties between nodes such as preserving a similar node connectivity [303, 311–315], for example. By contrast, for whole-graph embeddings the objective is to represent each graph by one vector in a lower-dimensional representation space [316–320]. In the approach presented here, we focus on the generation of whole-graph embeddings.

The representation of a set of graphs in a fixed-size lower-dimensional vector space allows one to get a notion for distances and similarity measures between graphs and to apply further downstream tasks, such as regression and clustering tasks, cf. Chapter 1. The appeal of performing deep representational learning lies on their ability to produce continuous permutationally invariant graph-level embeddings facilitating a well-suited representation for the mentioned operations on graph data. Therefore, the mapping to a different topology by means of graph neural networks represents a possible way to evade the graph matching problem and the related problem of determining similarities between graphs with algorithm operating directly on the graphs, as the graph-edit distance [204, 321–323], for example.

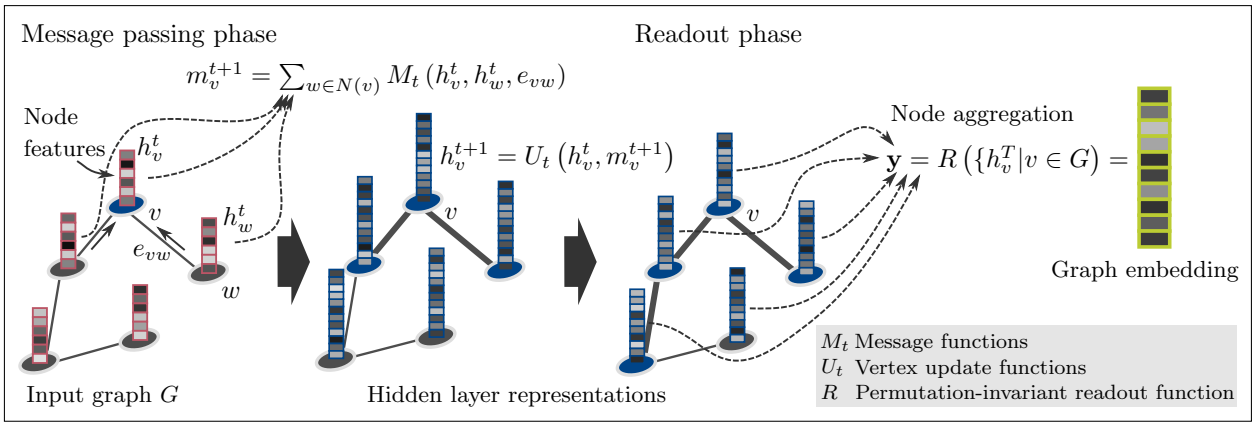

Figure 10.1: Scheme of a message passing neural network based on the example of a graph G consisting of edges e_{vw} and nodes v, w with (hidden) feature vectors h_v^t and h_w^t. The upper index t refers to the respective layer of the graph neural network. In the message passing phase, a message m_v^{t+1} is computed for each node v with respect to a message function M_t depending on the feature vectors h_v^t and h_w^t as well as the edges e_{vw}. In this example, the message is obtained by a sum over neighboring nodes $w \in N(v)$. In the second step, node feature vectors in the subsequent layer are computed in terms of a vertex update function U_t. Depending on the number of layers, information can be passed going beyond neighboring nodes, as indicated by a change in the line width of the edges and the blue color of the nodes on the example of the node with subscript v. In the readout phase, the hidden node feature vectors are aggregated by the readout function R. If the function is invariant to a permutation of nodes and vertices, the embedded feature vector $\mathbf{y}$ is invariant to graph isomorphisms. In a machine learning framework, the parameterized functions M_t, U_t and R are differentiable and can be optimized. Note that the readout function can also be relatively simple like the mean over all node feature vectors, for example [197].

An inherently built-in permutation invariance with respect to vertices and associated edges of graphs can be implemented, for example, by using message passing neural networks [197]. The embedding is generated by a message passing phase and a subsequent read-out phase, also illustrated in Fig. 10.1. In the message passing phase, information of neighboring nodes is accumulated at each node by trainable mappings and thus propagates through the network. The scope of incorporated information with respect to two nodes on the graph is determined by the node connectivity and the number of consecutive graph neural network layers. The resulting hidden representations at each node are an accumulated and condensed version of the node itself and its neighborhood. The readout phase consists of a subsequent aggregation of all nodes resulting in a fixed-size continuous latent lower-dimensional representation. A permutation invariance with respect to a different numbering of the nodes and edges of the input graph can be obtained by an appropriate pooling operation, as, for example, the mean over all hidden feature vectors of the respective nodes.

There has been a large body of work focused on integrating graph neural networks into encoder-decoder architecture, such as variational autoencoders. While purposing generic graph neural networks for encoding is fairly straightforward, devising a graph decoder capable of performing graph reconstruction has proven a major challenge due to the difficulty of accounting for the permutational invariance in the reconstruction loss [197]. In Ref. [324] the GraphVAE architecture is introduced as a solution for small graphs that uses approximate graph matching with soft discretization.

Here, we present preliminary work on an information-theoretic approach for the generation of graph embeddings. The proposed *neural adversarial embedding algorithm* works without the need of directly reconstructing embedded samples. The idea is to reduce the dimensionality of the input space while

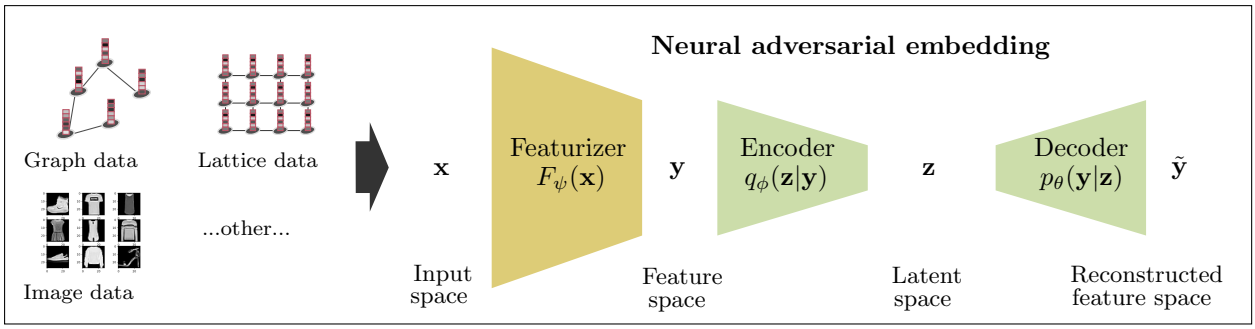

Figure 10.2: Architecture of the proposed neural adversarial embedding algorithm. The featurizer's $F_\psi(\mathbf{x})$ objective is to increase the reconstruction loss of the imitator. The imitator, implemented by the encoder $q_\phi(\mathbf{z}|\mathbf{y})$ and the decoder $q_\theta(\mathbf{y}|\mathbf{z})$, is trained based on the InfoVAE objective [195] and, therefore, aims to reduce the reconstruction loss, resulting in a zero-sum game between the featurizer and the imitator. During training, the entropy in the feature space increases, as shown in Sec. 10.2, implying a higher amount of information in the feature space. Compared to other algorithms for generating embeddings, the architecture has the advantage that a reconstruction of the input space is not necessary. This is, in particular, helpful if the embedding is supposed to be invariant to certain transformation of the input data and if this invariance is explicitly implemented in the featurizer. In this case, a direct comparison between a reconstructed sample and the input sample turns out to be difficult. Due to the desired invariance, there are multiple possible reconstructions that refer to the same embedded input sample. For example, for graph-structured data, an invariance with respect to a permutation of the nodes can be implemented by message passing neural networks by a computation of the mean over all hidden node feature vectors in the readout phase, cf. Fig. 10.1. A respective reconstruction of the input graph is hindered by the graph-matching problem. As another example, an application on lattice configurations exhibiting for most systems a translation and rotation invariance will be discussed in greater detail in Sec. 11.2.

maintaining relevant and meaningful information by entropy optimization in the latent representation space. The resulting embedding enables the definition of a distance metric and provides a condensed and structured representation of the input dataset.

The optimization is implemented by a zero-sum game of a so-called featurizer, generating the embedding, and an opponent, aiming at a reconstruction of the embedded data. The architecture is illustrated in Fig. 10.2. We refer to an optimization of the entropy instead of the mutual information since the latter is hard to estimate for graph-structured input data. Furthermore, the entropy of the embedded distribution can be directly related to the reconstruction loss of the opponent rendering a respective optimization straightforward, as will be shown in the following section. Note that the proposed method is not restricted to graph-structured data, but can be applied to other modalities of data. We discuss a potential application in the domain of physical systems in the next chapter and concentrate here on the embedding of graph-structured data, as molecules, for example.

10.1 Neural adversarial embedding

The goal is to embed an input data set $\{\mathbf{x}^{(i)}\}_{n=1}^N$ with N samples by a mapping $F_\psi(\mathbf{x})$ into a lower dimensional latent space $\{\mathbf{y}^{(i)}\}_{n=1}^N$, which we will refer to as feature space, with dimension D_{feature}:

$$\mathbf{y} := F_\psi(\mathbf{x})\,. \tag{10.1}$$

The resulting embedding is meant to contain a compressed representation of the high-dimensional input data space while important properties are preserved and a notion of distance and similarity between different input samples is feasible. From an information-theoretic point of view, the mutual information between the input and the feature space is expected to be high while the dimension is reduced.

We aim to achieve this by optimizing the entropy over the embedded data set in the feature space. From a information-theoretic point of view, this is a natural choice to obtain good representations since entropy relates to the amount of information present in the embedding. On the other hand, a too high entropy suggests a poor compression of correlations in the input data set. In the worst case, an increase of the entropy encourages an amplification of noise in the input data space while important correlations within the different samples are neglected. Therefore, a correct trade-off between a too high and a too little entropy of the feature space is crucial for the generation of meaningful embeddings. In particular, it is important to be able to control which kind of information is preserved by the mapping $F_\psi(\mathbf{x})$ of the input data set into the embedded space $\mathbf{y}$. A thorough understanding of these issues is at this point still lacking and will be analysed in greater detail in future work. In addition, we want to point out that one has to take into account a finite number of samples in the input data set with regard to the notion of entropy and a probability distribution in the feature space.

A direct optimization of the entropy in the feature space turns out be difficult since the quantity is in most cases hard to estimate due to a missing explicit expression of the underlying probability distribution. Additionally, utilizing existing methods with tractable probability distributions, such as normalizing flows [215], is limited due to the goal to embed graph-structured data and a respective implementation by means of message passing neural networks.

Here, we introduce an algorithm called *neural adversarial embedding algorithm* resolving the problem of entropy optimization by an implicit maximization based on a zero-sum game between the featurizer and an opponent, see also Fig. 10.2. While the featurizer generates lower-dimensional representations of the input data, the opponent's objective is to imitate the featurizer. The opponent encodes the embedded representation into a latent space and subsequently tries to reconstruct the original feature space representation. The quality of the reconstructions is regulated by a bottleneck in the lower-dimensional latent space. The opponent, denoted in the following as imitator, is represented in the easiest setting by a variational autoencoder (VAE). The strength of the featurizer can be adjusted by the expressiveness of the mapping $F_\psi(\mathbf{x})$ and additional regularizers in the feature space.

The featurizer tries to fool the imitator by generating more complicated representations implemented by the objective to maximize the reconstruction error of the opponent. The training set-up is a zero-sum game since the imitator is trained to minimize the reconstruction loss. Therefore, an improvement of the featurizer implies a loss for the imitator and vice versa. We will show in the next section that for the case of a variational autoencoder a respective increase of the reconstruction error can be related to a higher entropy in the feature space. Accordingly, the algorithm utilizes the interrelation between the reconstruction accuracy of the variational autoencoder and the complexity of the embedding. Competition between the two opponents within the training results in an implicit generation of entropy in the feature space.

The algorithm results in the following training set-up, cf. Fig. 10.2. The imitator consists of an encoder $q_\phi(\mathbf{z}|\mathbf{y})$ and a decoder $p_\theta(\mathbf{y}, \mathbf{z})$ with trainable parameters ϕ and θ. The VAE latent space is

denoted as $\mathbf{z}$. The featurizer is trained based on the following objective

$$\psi^* = \underset{\psi}{\operatorname{argmax}}\, \mathcal{L}_{\mathrm{F}}(\psi)\,, \tag{10.2}$$

with

$$\mathcal{L}_{\mathrm{F}}(\psi) = \mathcal{L}_{\mathrm{Rec}}(\psi) + \mathcal{L}_{\mathrm{Reg}}(\psi)\,, \tag{10.3}$$

and where the reconstruction error refers to the expectation value of the negative log-likelihood of the decoder, cf. Ref. [170]:

$$\mathcal{L}_{\mathrm{Rec}} = \mathbb{E}_{p_\psi(\mathbf{y})}\left[-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{y})}\left[\log p_\theta(\mathbf{y}|\mathbf{z})\right]\right]\,. \tag{10.4}$$

A maximization can be accomplished by changing the underlying data distribution in the feature space in dependence on the featurizer parameters ψ. An additional regularization loss $\mathcal{L}_{\mathrm{Reg}}$,

$$\mathcal{L}_{\mathrm{Reg}}(\psi) = \gamma\, \mathbb{E}_{p_\psi(\mathbf{y})}\left[\| \mathbf{y} \|_2^2\right] + \kappa\, D_{\mathrm{KL}}\left(p_\psi(\mathbf{y}) \parallel \mathcal{N}(\mathbf{0}, \mathbf{I})\right)\,. \tag{10.5}$$

ensures that the feature space does not spread to infinity, which is implemented by the first regularization term. The second term supports the featurizer, in particular at the beginning of the training, to generate a distribution in the feature space with a finite variance in each dimension.

In turn, the imitator aims to minimize the respective reconstruction loss

$$\theta^*, \phi^* = \underset{\theta,\phi}{\operatorname{argmin}}\, \mathcal{L}_{\mathrm{Im}}(\theta, \phi)\,, \tag{10.6}$$

with

$$\begin{aligned}\mathcal{L}_{\mathrm{Im}}(\theta, \phi) = \mathcal{L}_{\mathrm{Rec}}(\theta, \phi) + (1-\alpha)\, \mathbb{E}_{p_\psi(\mathbf{y})} D_{\mathrm{KL}}\left(q_\phi(\mathbf{z}|\mathbf{y}) \parallel p(\mathbf{z})\right) \\ + (\alpha + \lambda - 1)\, D_{\mathrm{KL}}\left(q_\phi(\mathbf{z}) \parallel p(\mathbf{z})\right)\,,\end{aligned} \tag{10.7}$$

referring to the loss function of the InfoVAE [195]. The loss provides better control over the trade-off between the quality of reconstructions and generated samples. In addition to the standard variational autoencoder, it takes into account the mutual information between the feature space and the latent space which is regulated in the above loss by the parameter α. The featurizer and the imitator are trained in an alternating way and the loss functions are approximated batch-wise, i.e., by small subsets of the training data. We recommend to choose a slightly higher learning rate for the imitator. Furthermore, it is helpful to perform several training steps for the imitator after each update of the featurizer to ensure that the variational autoencoder is close to convergence during training. This stabilizes the mutual training and facilitates the featurizer to correctly deduce necessary changes in the feature space for complicating reconstructions of the imitator.

After training, the imitator is most likely too weak to generate reasonable embeddings in the latent space. For this reason, it is recommended to train another more powerful embedding algorithm on the respective data set in the feature space. This algorithm is supposed to better capture the relevant information in the feature space entailing more meaningful lower-dimensional representations in the latent space.

10.2 Information-theoretic insights

In the following, we want to elaborate why the training set-up entails a successive increase of the entropy in the feature space. We start by rewriting the reconstruction loss

$$\begin{aligned}\mathcal{L}_{\text{Rec}} &= \mathbb{E}_{p_\psi(\mathbf{y})}\left[-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{y})}\left[\log p_\theta(\mathbf{y}|\mathbf{z})\right]\right] \\ &= \mathbb{E}_{q_{\psi,\phi}(\mathbf{y},\mathbf{z})}\left[-\log\frac{p_\theta(\mathbf{y},\mathbf{z})}{p_\theta(\mathbf{y})p(\mathbf{z})}\right] - \mathbb{E}_{p_\psi(\mathbf{y})}\left[\log p_\theta(\mathbf{y})\right]\,, \end{aligned} \tag{10.8}$$

where we make use of

$$p_\theta(\mathbf{y}|\mathbf{z}) = \frac{p_\theta(\mathbf{y},\mathbf{z})}{p(\mathbf{z})}\frac{p_\theta(\mathbf{y})}{p_\theta(\mathbf{y})} = \frac{p_\theta(\mathbf{y},\mathbf{z})}{p_\theta(\mathbf{y})p(\mathbf{z})}\,p_\theta(\mathbf{y})\,. \tag{10.9}$$

The last term in Eq. (10.8) can be expressed as:

$$-\,\mathbb{E}_{p_\psi(\mathbf{y})}\left[\log p_\theta(\mathbf{y})\right] = D_{\text{KL}}(p_\psi(\mathbf{y})\parallel p_\theta(\mathbf{y})) + H(p_\psi(\mathbf{y}))\,. \tag{10.10}$$

Therefore, the loss term can be rewritten,

$$\mathcal{L}_{\text{Rec}}(\psi) = -I_{q_{\psi,\phi}(\mathbf{y},\mathbf{z})\|p_\theta(\mathbf{y},\mathbf{z})}(\mathbf{y};\mathbf{z}) + D_{\text{KL}}\left(p_\psi(\mathbf{y})\parallel p_\phi(\mathbf{y})\right) + H(p_\psi(\mathbf{y}))\,. \tag{10.11}$$

We refer to the first term as mutual cross-information, cf. Ref. [180],

$$I_{q_{\psi,\phi}(\mathbf{y},\mathbf{z})\|p_\theta(\mathbf{y},\mathbf{z})}(\mathbf{y};\mathbf{z}) = \mathbb{E}_{q_{\psi,\phi}(\mathbf{y},\mathbf{z})}\left[\log\frac{p_\theta(\mathbf{y},\mathbf{z})}{p_\theta(\mathbf{y})p(\mathbf{z})}\right]\,. \tag{10.12}$$

In contrast to the work in Ref. [180], the expectation value is evaluated with respect to the modeled distribution $q_{\psi,\phi}(\mathbf{y},\mathbf{z})$ and the logarithm depends on the true distributions. The mutual cross-information coincides with mutual information if $q_{\psi,\phi}(\mathbf{y},\mathbf{z}) = p_\phi(\mathbf{y},\mathbf{z})$. Since $q_{\psi,\phi}(\mathbf{y},\mathbf{z}) = q_\phi(\mathbf{z})q_\phi(\mathbf{y}|\mathbf{z})$ and $p_\phi(\mathbf{y},\mathbf{z}) = p(z)p_\phi(\mathbf{y}|\mathbf{z})$, this can be achieved by learning the correct likelihood $q_\phi(\mathbf{y}|\mathbf{z})$ and by matching the prior in the latent space, $q_\phi(\mathbf{z}) = p(\mathbf{z})$.

The reformulation of the reconstruction error in Eq. (10.11), allows for a better understanding of the mutual training of the featurizer and the imitator. For the imitator, targeting a minimization of the reconstruction loss, the loss function has a lower bound given by

$$L_{\text{Rec}} \geq -I_{q_{\psi,\phi}(\mathbf{y},\mathbf{z})\|p_\theta(\mathbf{y},\mathbf{z})}(\mathbf{y};\mathbf{z}) + H(p_\psi(\mathbf{y})) = 0\,, \tag{10.13}$$

where we make of the property of the KL-divergence to be non-negative, i.e., $D_{\text{KL}}\left(p_\psi(\mathbf{y})\parallel p_\phi(\mathbf{y})\right) \geq 0$. This relation indicates that in case of an equality all information about the feature space is preserved by the encoder and the decoder. From an information-theoretic point of view, the mutual information term indicates that knowing $\mathbf{z}$ implies full knowledge over $\mathbf{y}$. The mutual information coincides for this case with the entropy of the feature space.

The inequality in Eq. (10.13) suggests that the lower bound can be increased by using an imperfect encoder/decoder pair not having the capacity to preserve all information and by employing the respective regularizations on the latent space. On the other hand, this statement implies that for a given fixed expressivity of the imitator, the featurizer can increase the reconstruction loss by an increase of the entropy in the feature space. According to Eq. (10.11), this is the only quantity the featurizer has exclusive access to since the entropy only depends on the distribution of the feature

space while the other two terms change based on the response of the imitator to variations of the entropy. Therefore, the mutual training can be interpreted as a source for entropy generation in the feature space.

The above observation entails that there is a dependency between the entropy of the feature space, the reconstruction accuracy and the quality of the embedding in the latent space. In the case where a low amount of information is encoded in the feature space, the reconstruction error is very small, resulting in a latent space representing the same information as the feature space. Increasing the amount information entails a higher entropy in the feature space. Despite a higher reconstruction loss, this leads to a more meaningful compressed representation in the latent space depending only on the most significant features of the feature space. This holds since significant features represent the kind of information which support a better reconstruction and can be passed more easily through the networks. At a certain point, in dependence on the properties of the imitator, determined by the parameters of the loss function (10.7), either the latent space or the reconstructed space will no longer contain meaningful information and the training diverges. A possible way to prevent this breakdown is the proposed introduction of regularizers for the training of the featurizer. Limiting the expressiveness of the featurizer represents a further approach to restrict the entropy of the feature space. Alternatively, the training can be stopped at certain point.

Defining an appropriate upper limit for the entropy of the feature space is generally not easy in the proposed unsupervised training framework since it largely depends on the distribution of the input dataset. In general, it is difficult to define measures for a 'good' embedding as this is subject to the specific application task. A possible indicator can be, for example, the performance on downstream tasks such as classification or regression which are related to the desired properties of the embedding. Testing the neural adversarial embedding algorithm in the light of this discussion in greater detail is subject to future work. This also holds for a thorough analysis of the meaningfulness of embeddings generated solely based on entropy optimization. In contrast to most of the embedding algorithms, a feedback loop to the input data is missing, rendering a reasonable compression more difficult and raising several open questions. It needs to be analysed what kind of information the mapping of the featurizer preserves and to what extent this depends on properties inherent to the input data set. Furthermore, we expect the encoder/decoder pair to have a high impact on the feature space and, because of the adversarial training, the featurizer favors embedded distributions that are not easily representable by a variational autoencoder.

10.3 Preliminary results

In the following, we report on preliminary results obtained for a graph-structured data set and a downsampled dataset of FashionMNIST [325]. The samples of the FashionMNIST dataset have been downsampled to a size of 10×10. Details on the hyperparameters for training and on the utilized network architectures can be found in App. E in Tab. E.1 and Tab. E.2.

The loss curves for the neural adversarial embedding algorithm are shown in Fig.10.3. The constantly increasing reconstruction error illustrates that the training algorithms works. Towards the end of the training, the imitator was no longer able to produce good reconstructions which is when we stopped the training. Following the information-theoretic insights of the previous section, the reconstruction error and the entropy of the feature space are correlated. In dependence on the desired compression of the input dataset, a good embedding is expected to correspond to a specific reconstruction error during training.

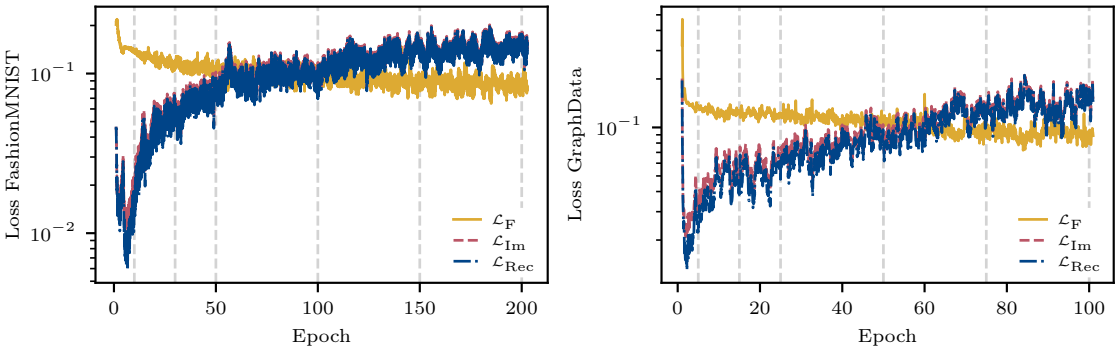

Figure 10.3: Loss curves of the neural adversarial embedding algorithm for the downsampled FashionMNIST data (left) and the mock graph dataset (right). The respective loss terms are defined in Eqs. (10.3), (10.7) and (10.4). During training, the reconstruction error constantly increases for both datasets resulting in embeddings with a growing entropy. Embeddings at different training stages, indicated by the dashed gray vertical lines, are compared in Figs. 10.4 and 10.6.

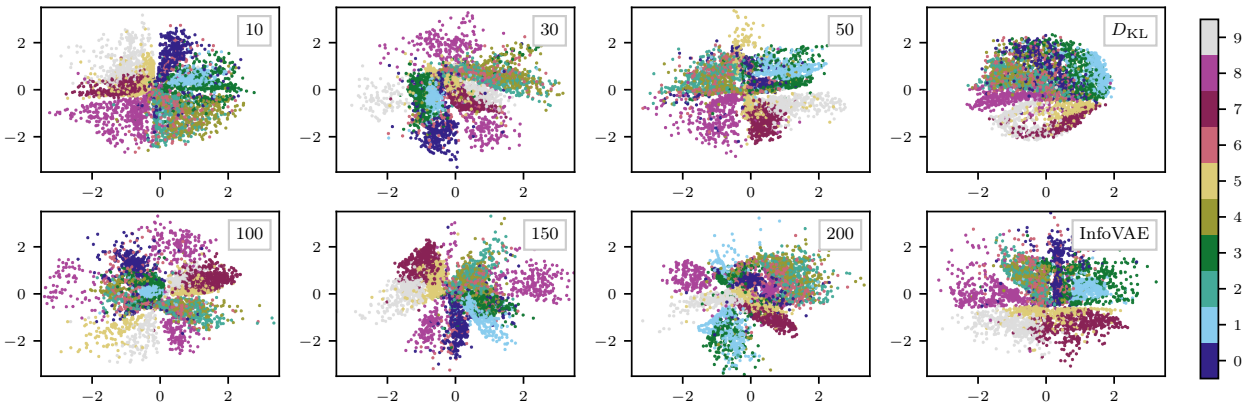

Figure 10.4: Two-dimensional latent space representations of the downsampled FashionMNIST dataset for the neural adversarial embeddings algorithm and a standard InfoVAE. The embeddings are generated by a training of several InfoVAEs based on feature space representations at different stages during training. The numbers in the small boxes refer to the respective epoch. The upper right plot refers to a training based solely on the regularization loss of the neural adversarial embeddings, cf. Eq. (10.5). The lower right plot shows the embedding of the standard InfoVAE. A comparison with the neural adversarial embedings entails that the algorithm is capable of inferring a meaningful representation of the input dataset. The increasing sparsity at larger epoch is in concordance with the expected higher entropy in the feature space.

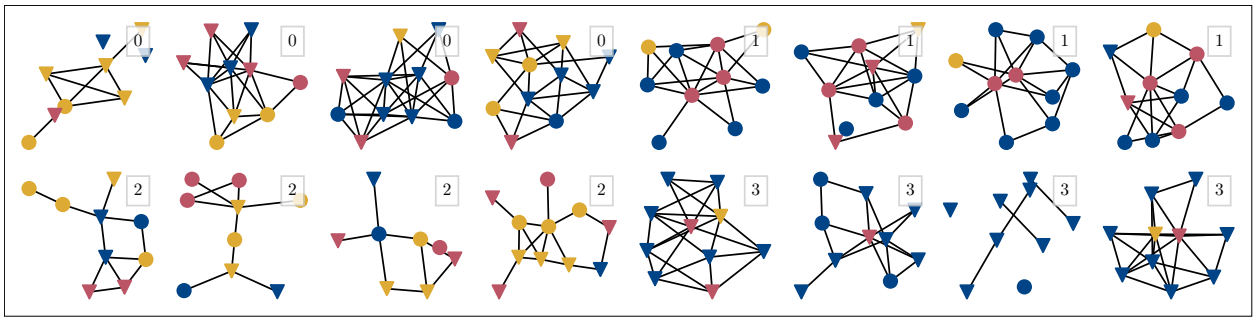

Figure 10.5: Samples of the different classes of the mock graph data set for testing the neural adversarial embedding algorithm. The data is generated with respect to varying probabilities for the shape and the color of nodes, the node connectivity and the total number of nodes per graph. A one-hot vector representation with six entries is computed for each node based on its color and its shape.

Fig. 10.4 shows embeddings of the FashionMNIST [325] dataset for the neural adversarial embedding algorithm at several epochs related to different reconstruction errors. The embeddings refer to the two-dimensional latent space of a stronger InfoVAE trained based on feature space embeddings at the respective epochs. The numbers in the small boxes of the subplots refer to the considered epochs which are also indicated by the dashed gray vertical lines in Fig. 10.3. In addition, we trained for comparison a standard InfoVAE on the dataset and the neural adversarial embedding based solely on the regularization loss. In the letter case, the imitator is not utilized at all. Instead, the feature space distribution is trained based on the D_{KL} loss to be Gaussian distributed, cf. Eq. (10.5).

The similarity of the embeddings in Fig. 10.4 implies that the feature space contains a meaningful representation of the input data set. For higher reconstruction errors of the imitator, one can observe larger deviations to a Gaussian distribution in the latent space. This can be directly related to the expected higher entropy in the feature space. For a higher entropy, the InfoVAE needs to put more effort into the reconstructions which entails a higher necessary expressive power in the latent space.

In a similar training set-up, we tested the algorithm on a mock graph-structured dataset, cf. Fig. 10.5 and Fig. 10.6. The dataset consists of graphs with a varying number of nodes whereby each node is characterized by a specific color and shape. The different classes in the graph data set are constructed based on varying class-dependent probabilities for the shape and the color of the nodes as well as for the node connectivity and the total number of nodes. Therefore, a finite overlap of the classes is expected since a single graph can, in principle, be generated by each of the classes, as can be observed in Fig. 10.6.

10.4 Summary

The chapter introduces the neural adversarial embedding algorithm as a novel algorithm for generating lower-dimensional representations. The algorithm works without the need for reconstructing input data rendering it a powerful tool for input data exhibiting invariances with respect to certain transformations. We applied the algorithm on graph-structured data. In dependence on the generation of the feature vector representation, reconstructions can be difficult due to the graph matching problem related to the absence of a fixed numeration of nodes in a graph. Additionally, embeddings of the downsampled FashionMNIST dataset are considered for a possible comparison to a standard InfoVAE.

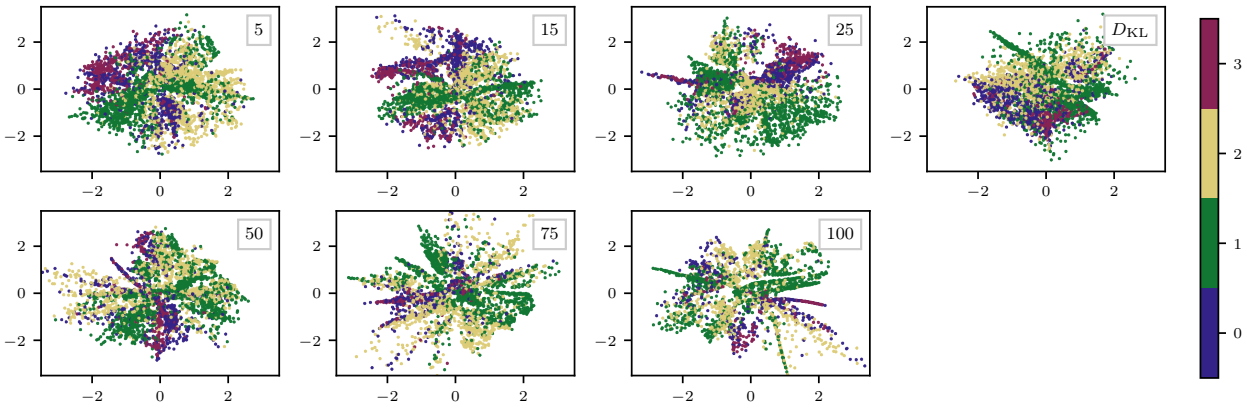

Figure 10.6: Neural adversarial graph embeddings for the mock graph data set. Similar to Fig. 10.4, the subplots show two-dimensional latent space representations obtained by InfoVAEs that were trained on feature space embeddings at different training stages. The increased sparsity matches with the observations for the FashionMNIST dataset. The partial mixing of the different classes indicates a varying overlap for a sampling of certain type of graphs. This can be traced to the probabilistic generation of the graphs for each class.

The algorithm maps the input data distribution in a feature space and is trained to optimize the entropy of the respective embedding by means of a zero-sum game. The results demonstrate that an increase of the entropy in the feature space can be sufficient for generating informative and meaningful lower-dimensional representations.

In future work, we want to apply the algorithm on real-world datasets and benchmark its applicability on molecular property prediction for the QM9 data set [197, 326]. An application on lattice configurations and related potential benefits will be discussed in Sec. 11.2. Furthermore, we want to analyse relations between the input distribution and the embedded distribution in more detail. This includes, for example, an estimation of the mutual entropy between the input space and the feature space at different stages during training with MINE [186]. Another aspect relates to evaluating properties of the embedding with regard to a distance metric and similarity measures of embedded samples. For these cases, the latent space of an InfoVAE and an Euclidean space might not be optimal. Instead, a mapping on other topologies or by other embedding algorithms is expected to be more appropriate.

In general, the featurizer looks for weaknesses of the imitator. This needs to be kept in mind for a suitable definition of regularization terms. For example, in the case of the variational autoencoder, the regularization terms keep the feature space embedding distribution in a regime which is reconstructable by the imitator. On the other hand, there might be applications where the specialization of the featurizer on weaknesses of its opponent can be utilized to support the training of a different algorithm.

Lastly, we want to point out that the architecture can be extended by replacing the reconstruction error by a discriminative network, similar to a generative adversarial neural network [191]. The task of the discriminator is to distinguish whether a sample originated from the featurizer or the imitator. The featurizer tries to support the discriminator by generating embeddings with a higher information content leading to worse reconstructions of the imitator. The imitator is trained to fool the discriminator accomplished by generating better reconstructions. Overall, the amount of information in all latent spaces increases.

CHAPTER 11

Towards novel insights in lattice field theory with explainable machine learning

This chapter is in parts based on Ref. [3] and refers to Ref. [7].

Lattice simulations of quantum field theories have proven essential for the theoretical understanding of fundamental interactions from first principles, perhaps most prominently so in quantum chromodynamics. However, an in-depth understanding of the emergent dynamics is often difficult. In cases where such an understanding remains elusive, it may be instructive to search for so far unidentified structures in the data to better characterise the dynamics.

One ansatz for the identification of relevant observables from lattice data is through representation learning, i.e. by training on a pretext task. The rationale behind this approach is that the ML algorithm learns to recognise patterns which can be leveraged to construct observables from low-level features that characterise different phases. However, solving a given task does by itself not lead to physical insights, since the inner structure of the algorithm typically remains opaque. This issue can at least partially be resolved by the use of "explainable AI" techniques, which have recently attracted considerable interest in the ML community and beyond.

Simple methods from statistics and ML often lack the capability to model complex data, whereas sophisticated algorithms typically tend to be less transparent. A commonly used example is principal component analysis (PCA). It has been successfully applied to the extraction of (albeit already known) order parameters for various systems [130, 133, 137]. However, its linear structure prohibits the identification of complex non-linear features, e.g. Wilson loops in gauge theories. Hence, we require tools capable of modeling non-linearities, such as deep neural networks [141]. They allow for a more comprehensive treatment of complex systems, which has been demonstrated e.g. for fermionic theories in Refs. [131, 136]. The approach also enables novel procedures, such as learning by confusion and similar techniques, to locate phase transitions in a semi-supervised manner [139, 159]. For lattice QCD, action parameters can be extracted from field configurations [149]. Overall, deep learning tools seem particularly well-suited to grasp relevant information about quantum field dynamics in a completely data-driven approach by learning abstract internal representations of relevant features.

However, their lack of transparency is frequently a major drawback of using such methods which prohibits access to and comprehension of these representations. A unified understanding of how and what these architectures learn, and why it seems to work so well in a wide range of applications is still pending. To better understand the processes behind neural network-driven phase detection in lattice models, multiple proposals have been made, such as pruning [114, 134, 150], utilising (kernel) support vector machines [140, 157], and saliency maps [158]. Interpretability is also investigated for other applications in theoretical physics, e.g. by employing twin neural networks [327].

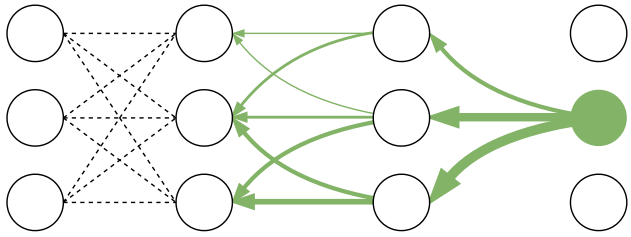

Figure 11.1: Sketch of LRP through the last two layers of a classification network that predicts one-hot vectors. Relevance is indicated by arrow width. The conservation law requires the sum of widths to remain constant during backpropagation. Diagram adapted from Ref. [335].

Also, in the broader scope of ML research, there has been growing interest in interpretability approaches, most of them focusing on post-hoc explanations for trained models. So-called attribution methods typically assign a relevance score to each of the input features that quantifies which features the classifier was particularly sensitive to, or influenced the algorithm towards/against an individual classification decision. In the domain of image recognition, such attribution maps are typically visualised as heatmaps overlaying the input image. The development of attribution algorithms is a very active field of research in the ML community. Therefore, we refer to dedicated research articles for more in-depth treatments [328, 329]. Very broadly, the most important types of such local interpretability methods can be categorised as: 1. Gradient-based, such as saliency maps [330] obtained by computing the derivative of a particular class score with respect to the input features or integrated gradients [331]. 2. Decomposition-based, such as layer-wise relevance propagation (LRP) [332] or DeepLift [333]. 3. Perturbation-based, as in Ref. [334], investigating the change in class scores when occluding parts of the input features.

In Sec. 11.1, we give a brief overview of an approach, presented in Ref. [3], for identifying observables of the scalar Yukawa model (2+1)d by combining a supervised representation learning task with interpretability methods. We continue in Sec. 11.2 with a short outlook on using the proposed neural adversarial embedding algorithm, cf. Chapter 10, as an alternative, unsupervised approach for finding expressive observables of physical systems.

11.1 Supervised representation learning

Following Ref. [3], we focus on utilizing layer-wise relevance propagation (LPR) [332] for the identification of the most important observables depending on the different phases of the considered theory. We utilize action parameter regression as a pretext task for a subsequent investigation using LPR.

Layer-wise relevance propagation is one of several popular post-hoc attribution methods that propagate the prediction of neural network back to the input domain, thereby highlighting features that influence the algorithm towards/against a particular classification decision. The method is a particular variant of decomposition-based attribution methods, which has been successfully applied to other problems in physics and chemistry, e.g. in the context of atomistic systems [336]. The general idea of LRP is to start from a relevance assignment in the output layer and subsequently propagate this relevance back to the input using certain propagation rules, see the sketch in Fig. 11.1. In this way, the method assigns a relevance score to each neuron, where positive (negative) entries strongly influence the classifier towards (against) a particular classification decision. In the work in Ref. [3], LPR correctly assigns relevances to observables in the different phases that agree with

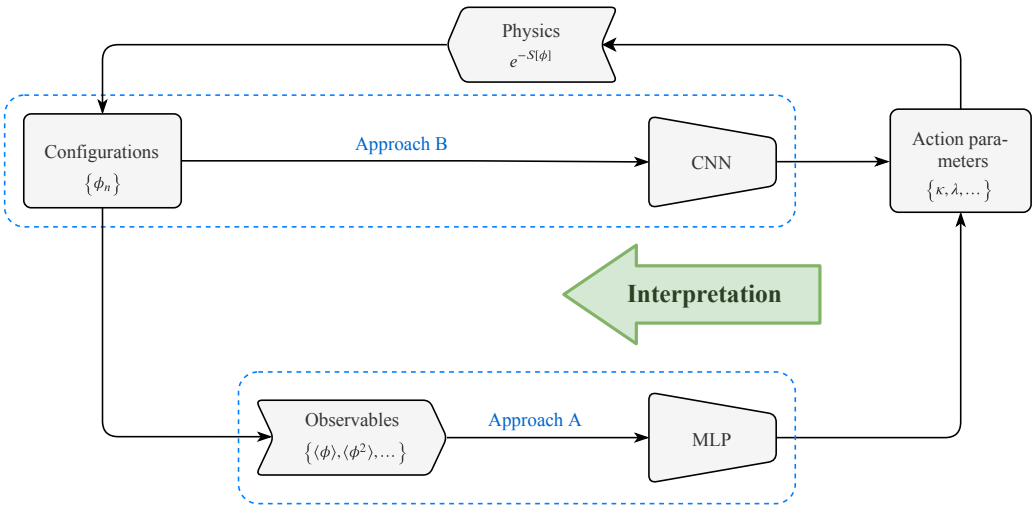

Figure 11.2: Sketch of the followed strategies in Ref. [3] to learn meaningful structures from simulation data by analysing the networks trained for action parameter inference. Field configurations used for training are either preprocessed into observables for the MLP (Approach A) or directly operated upon with a CNN (Approach B). Obtaining accurate predictions for the parameters indicates approximate cycle consistency in the above diagram, which supports the notion that the networks have successfully identified characteristic features. These can then be extracted in a subsequent interpretation step using LRP.

physical expectations and allows for the construction of observables based on the convolutional filters, as discussed in more detail in the following.

We tested this approach in the context of Yukawa theory in (2+1) dimensions. As a pretext task, we trained a multilayer perceptron (MLP) and a convolutional neural network (CNN) to infer the associated hopping parameter κ from a set of known observables (Approach A), as well as solely from the raw field configurations (Approach B), akin to Ref. [149]. Both variants are sketched in Fig. 11.2. In the first case, without providing any prior knowledge of the phase boundaries, LRP managed to reveal the underlying phase structure, cf. Fig. 11.3, and returns a phase-dependent importance hierarchy of the observables. The obtained relevance for the different observables in the respective phases is in accordance with physical expert knowledge, see also Ref. [3] for more details. In the second case, by calculating the relevances of the learned filters of the CNN, we could associate each of them with one of the physical phases and thereby extract the known order parameters, cf. Fig. 11.3. Moreover, it facilitated the construction of an observable that characterises the symmetric, parametric phase, derived by the convolutional filters in Fig. 11.4.

Following Ref. [3], this section reviewed the application of interpretability methods to deep neural network classifiers as a general-purpose framework for the identification of physical features from lattice data. The approach facilitates an interpretation of a network's predictions, permitting a quantitative understanding of the internal representations that the network learns in order to solve a pretext task—in this case, inference of action parameters. This culminates in the extraction of relevant observables from the data, leading to insights about the phase structure. The results demonstrate that due to its broad applicability, attribution methods such as LRP could prove a useful and versatile tool in our search for new physical insights.

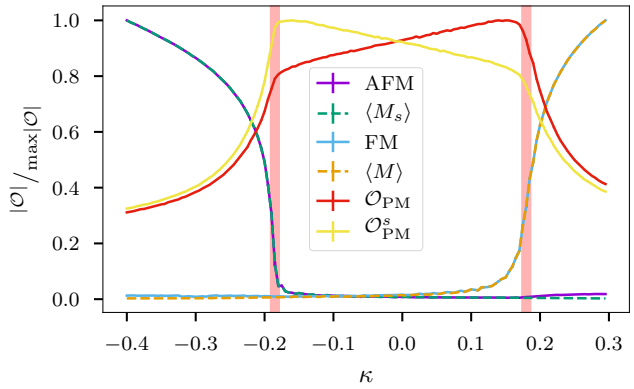

Figure 11.3: Slice of the phase diagram of the Yukawa model in terms of normalized values of the magnetization $\langle M \rangle$, the so-called staggered magnetization $\langle M_s \rangle$ and of observables reconstructed from learned convolutional filters. Phase transitions are highlighted by the shaded bars. We distinguish an antiferromagnetic (AFM), a paramagnetic (PM) and a ferromagnetic (FM) phase. The reconstructed filters AFM and FM can be associated with the observables $\langle M \rangle$ and $\langle M_s \rangle$ and describe the FM and AFM phase. $\mathcal{O}_{\mathrm{PM}}$ and $\mathcal{O}^s_{\mathrm{PM}}$ are related by the so-called staggered symmetry and exhibit an approximate mirror symmetry around $\kappa = 0$. The latter reconstructed observables are well-suited for characterizing the symmetric phase and demonstrates the usefulness of applying LPR in the context of lattice configurations. The respective convolutional filters are illustrated in Fig. 11.4. See Ref. [3] for more details.

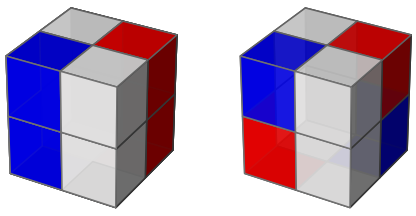

Figure 11.4: Convolutional filters corresponding to the observable $\mathcal{O}_{\mathrm{PM}}$ (left) and its corresponding staggered counterpart $\mathcal{O}^s_{\mathrm{PM}}$ (right).

11.2 Unsupervised representation learning

Encouraged by the results of the previous section, we want to discuss a possible utilization of the neural adversarial embedding algorithm, cf. Chapter 10, for the task of finding novel observables and structures in physical systems by analysing sampled lattice configurations. The algorithm is an unsupervised representation learning entailing the advantage that the learned lower-dimensional representations are not biased by a pretext task. However, as discussed in Chapter 10, it still needs to be evaluated what kind of information the mapping of the featurizer preserves. Combining the evaluation of physical systems with the expert knowledge about the respective physical properties can also help in this task.

Compared to other unsupervised learning approaches trained on a reconstruction of the input, the neural adversarial embedding algorithm has certain advantages. Invariances of the input data, which are directly related to the action of the physical system, can be explicitly implemented into the function mapping the lattice configurations into a lower-dimensional representation. These invariances are usually reflected in the observables of the given system. Lattice data often exhibit a translation and rotation invariance. In this case, observables are mostly computed by a mean over local properties of the lattice.

Unsupervised training algorithms relying on a reconstruction of the input data exhibit for this type of observables similar problems as encountered for the reconstruction of graphs, discussed in Chapter 10. Because of an often seen translational and rotational invariance of the lattice, many different lattice configurations can lead to the same observable, rendering a correct reconstruction infeasible. Depending on the observable, an even higher number of configurations can yield the same observables. For example, in case of the magnetization of the Ising model, an additional permutation invariance with respect to the lattice sites needs to be taken into account. These properties hamper a successful training of unsupervised training algorithms where a reconstruction is necessary.

Due to these properties, we want to study the generation of embeddings by means of the neural adversarial embedding algorithm in future work in greater detail and compare obtained results with other unsupervised approaches relying on a reconstruction, as studied, for example, in Ref. [133]. More specifically, we want to employ the same approach for computing lower-dimensional representations as proposed for graph-structured data in Chapter 10 by means of message passing neural networks. The computation of potential observables is implemented in the message passing phase whereas invariances of the lattice configurations are taken into account in the readout phase by computing the mean over all hidden representations of the lattice sites.

With respect to representation learning, this approach bears the advantage that the applied functions in the message passing phase can be directly interpreted as possible observables. A combination with other interpretable methods as layer-wise relevance propagation is feasible. Furthermore, we want to point that the approach also allows for studying observables going beyond a computation of local properties. The graph formed by the lattice can be replaced by any other graph that takes into account longer distances on the lattice. Applying the neural adversarial embedding algorithms on this representation allows for capturing global correlations on the lattice in respective observables. This can be, for example, beneficial in fermionic systems exhibiting global structures. A detailed analysis of this approach is postponed to future work.

CHAPTER 12

Spectral reconstruction with deep neural networks

This chapter is based on Ref. [2].

In this chapter, we build upon both the recent progress in the field of ML, particularly deep learning, as well as results and structural information gathered in the past decades from Bayesian reconstruction methods for tackling the problem of spectral reconstruction. We set out to isolate a property of neural networks that holds the potential to improve upon the standard Bayesian methods, while retaining their advantages, utilising the already gathered insight in their study.

Consider a feed-forward deep neural network that takes Euclidean propagator data as input and outputs a prediction of the associated spectral function. Although the reasoning behind this ansatz is rather different, one can draw parallels to more traditional methods. In the Bayesian approach, prior information is explicitly encoded in a prior probability functional and the optimisation objective is the precise recovery of the given propagator data from the predicted spectral function. In contrast, the neural network based reconstruction is conditioned through supervised learning with appropriate training data. This corresponds to implicitly imposing a prior distribution on the set of possible predictions, which, as in the Bayesian case, regularises the reconstruction problem. Optimisation objectives are now expressed in terms of loss functions, allowing for greater flexibility. In fact, we can explicitly provide pairs of correlator and spectral function data during the training. Hence, not only can we aim for the recovery of the input data from the predictions as in the Bayesian approach, but we are now also able to formulate a loss directly on the spectral functions themselves. This constitutes a much stronger restriction on potential solutions for individual propagators, which could provide a significant advantage over other methods. The possibility to access all information of a given sample with respect to its different representations also allows the exploration of a much broader set of loss functions, which could benefit not only the neural network based reconstruction, but also lead to a better understanding and circumvention of obstacles related to the inverse problem itself. Such an obstacle is given, for example, by the varying severity of the problem within the space of considered spectral functions. By employing adaptive losses, inhomogeneities of this type could be neutralised.

Similar approaches concerning spectral functions that consist of normalised sums of Gaussian peaks have already been discussed in Refs. [337, 338]. In this chapter, we investigate the performance of such an approach using mock data of physical resonances motivated by quantum field theory, and compare it to state-of-the-art Bayesian methods. The data are given in the form of linear combinations of unnormalised Breit-Wigner peaks, whose distinctive tail structures introduce additional difficulties (see Fig. 12.1 for an example reconstruction). Using only a rather naive implementation, the performance of our ansatz is demonstrated to be at least comparable and potentially superior,

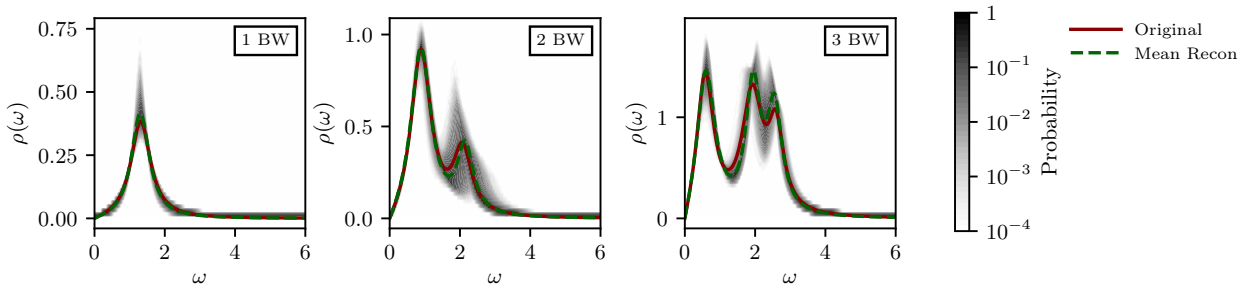

Figure 12.1: Examples of mock spectral functions reconstructed via our neural network approach for the cases of one, two and three Breit-Wigner peaks. The chosen functions mirror the desired locality of suggested reconstructions around the original function (red line). Additive, Gaussian noise of width 10^{-3} is added to the discretised analytic form of the associated propagator of the same original spectral function multiple times. The shaded area depicts for each frequency ω the distribution of resulting outcomes, while the dashed green line corresponds to the mean. The results are obtained from the FC parameter network optimised with the parameter loss. The network is trained on the largest defined parameter space which corresponds to the volume Vol O. The uncertainty for reconstructions decreases for smaller volumes as illustrated in Fig. 12.4. A detailed discussion on the properties and problems of a neural network based reconstruction is given in Sec. 12.3.1.

particularly for large noise levels. We then discuss potential improvements of the architecture, which in the future could establish neural networks to a state-of-the-art approach for accurate reconstructions with a reliable estimation of errors.

The chapter is organised as follows. The spectral reconstruction problem is defined in Sec. 12.1.1. State-of-the-art Bayesian reconstruction methods are summarised in Sec. 12.1.2. In Sec. 12.1.3 we discuss the application of neural networks and potential advantages. Sec. 12.2 contains details on the design of the networks and defines the optimisation procedure. Numerical results are presented and compared to Bayesian methods in Sec. 12.3. We summarise our findings and discuss future work in Sec. 12.4.

12.1 Spectral reconstruction and potential advantages

12.1.1 Defining the problem

Typically, correlation functions in equilibrium quantum field theories are computed in imaginary time after a Wick rotation $t \to it \equiv \tau$, which facilitates both analytical and numerical computations. In strongly correlated systems, a numerical treatment is in most cases inevitable. Such a setup leaves us with the task to reconstruct relevant information, such as the spectrum of the theory, or genuine real-time quantities such as transport coefficients, from the Euclidean data.

The information we want to access is encoded in the associated spectral function ρ. For this purpose it is most convenient to work in momentum space both for ρ and the corresponding propagator G. The relation between the Euclidean propagator and the spectral function is given by the well known Källen-Lehmann spectral representation,

$$G(p) = \int_0^\infty \frac{\mathrm{d}\omega}{\pi} \frac{\omega\,\rho(\omega)}{\omega^2 + p^2} \equiv \int_0^\infty \mathrm{d}\omega\, K(p,\omega)\rho(\omega)\,, \tag{12.1}$$

which defines the corresponding Källen-Lehmann kernel. The propagator is usually only available in the form of numerical data, with finite statistical and systematic uncertainties, on a discrete set of N_p points, which we abbreviate as $G_i = G(p_i)$. The most commonly used approach is to work directly with a discretised version of (12.1). We utilise the same abbreviation for the spectral function, i.e. $\rho_i = \rho(\omega_i)$, discretised on N_ω points. This lets us state the discrete form of (12.1) as

$$G_i = \sum_{j=1}^{N_\omega} K_{ij}\,\rho_j \ , \tag{12.2}$$

where $K_{ij} = K(p_i, \omega_j)\Delta\omega_j$. This amounts to a classic ill-conditioned inverse problem, similar in nature to those encountered in many other fields, such as medical imaging or the calibration of option pricing methods. Typical errors on the input data $G(p_i)$ are on the order of 10^{-2} to 10^{-5} when the propagator at zero momentum is of the order of unity.

To appreciate the problems arising in such a reconstruction more clearly, let us assume we have a suggestion for the spectral function ρ_{sug} and its corresponding propagator G_{sug}. The difference to the measured data is encoded in

$$\|G(p) - G_{\text{sug}}(p)\| = \left\| \int_0^\infty \frac{\mathrm{d}\omega}{\pi} \frac{\omega}{\omega^2 + p^2} \Big[\rho(\omega) - \rho_{\text{sug}}(\omega)\Big] \right\| , \tag{12.3}$$

with a suitable norm $\|.\|$. Evidently, even if this expression vanishes point-wise, i.e. $\|G(p_i) - G_{\text{sug}}(p_i)\| = 0$ for all p_i, the spectral function is not uniquely fixed. Experience has shown that with typical numerical errors on the input data, qualitatively very different spectral functions can lead to in this sense equivalent propagators. This situation can often be improved on by taking more prior knowledge into account, cf. the discussion in Ref. [339]. This includes properties such as:

1. Normalisation and positivity of spectral functions of asymptotic states. For gauge theories, this may reduce to just the normalisation to zero, expressed in terms of the Oehme-Zimmermann superconvergence [340, 341].
2. Asymptotic behaviour of the spectral function at low and high frequencies.
3. The absence of clearly unphysical features, such as drastic oscillations in the spectral function and the propagator.

Additionally, the parametrisation of the spectral function in terms of frequency bins is just one particular basis. In order to make reconstructions more feasible, other choices, and in particular physically motivated ones, may be beneficial, cf. again the discussion in Ref. [339]. In this work, we consider a basis formulated in terms of physical resonances, i.e. Breit-Wigner peaks.

12.1.2 Existing methods

The inverse problem as defined in (12.1) has an exact solution in the case of exactly known, discrete correlator data [342]. However, as soon as noisy inputs are considered, this approach turns out to be impractical [343]. Therefore, the most common strategy to treat this problem is via Bayesian inference. This approach is based on Bayes' theorem, which states that the posterior probability is essentially given by two terms, the likelihood function and prior probability:

$$P(\rho|D, I) \propto P(D|\rho, I)\, P(\rho|I)\,. \tag{12.4}$$

It explicitly includes additionally available prior information on the spectral function in order to regularise the inversion task. The likelihood $P(D|\rho)$ encodes the probability for the input data D to have arisen from the test spectral function ρ, while $P(\rho)$ quantifies how well this test function agrees with the available prior information. The two probabilities fully specify the posterior distribution in principle, however they may be known only implicitly. In order to gain access to the full distribution, one may sample from the posterior, e.g. through a Markov Chain Monte Carlo process in the parameter space of the spectral function. However, in practice one is often content with the maximum a posteriori (MAP) solution. Given a uniform prior, the Maximum Likelihood Estimate (MLE) corresponds to an estimate of the MAP.

12.1.3 Advantages of neural networks

In order to make genuine progress, we set out in this study to explore methods in which our prior knowledge of the analytic structure can be encoded in different ways. To this end, our focus lies on the use of Machine Learning in the form of artificial neural networks. These feature a high flexibility in the encoding of information by learning abstract internal representation. They possess the advantageous properties that prior information can be explicitly provided through the training data, and that the solution space can be regularised by choosing appropriate loss functions.

Minimising (12.3), while respecting the constraints discussed in Sec. 12.1.1, can be formulated as minimising the propagator loss

$$L_G(\rho_{\text{sug}}) = \|G[\rho_{\text{sug}}] - G[\rho]\| \,. \tag{12.5}$$

This corresponds to indirectly working on a norm or loss function for ρ, the spectral function loss

$$L_\rho(\rho_{\text{sug}}) = \|\rho_{\text{sug}} - \rho\| \,. \tag{12.6}$$

Of course, the optimisation problem as given by (12.6) is intractable, since it requires the knowledge of the true spectral function ρ. Minimising $L_\rho(\rho_{\text{sug}})$ for a given set of $\{\rho_{\text{sug}}\}$ also minimises L_G, since the Källén–Lehmann representation (12.1) is a linear map. In turn, however, minimising L_G does not uniquely determine the spectral function, as has already been mentioned. Accordingly, the key to optimise the spectral reconstruction is the ideal use of all known constraints on ρ, in order to better condition the problem. The latter aspect has fueled many developments in the area of spectral reconstructions in the past decades.

Given the complexity of the problem, as well as the interrelation of the constraints, this calls, in our opinion, for an application of supervised machine learning algorithms for an optimal utilisation of constraints. To demonstrate our reasoning, we generate a training set of known pairs of spectral functions and propagators and train a neural network to reconstruct ρ from G by minimising a suitable norm, utilising both L_G and L_ρ during the training. When the network has converged, it can be applied to measured propagator data G for which the corresponding ρ is unknown.

Estimators learning from labelled data provide a potentially significant advantage due to the employed supervision, because the loss function is minimised a priori for a whole range of possible input/output pairs. Accordingly, a neural network aims to learn the entire set of inverse transformations for a given set of spectral functions. After this mapping has been approximated to a sufficient degree, the network can be used to make predictions. This is in contrast to standard Bayesian methods, where the posterior distribution is explored on a case by case basis. Both approaches may also

be combined, e.g. by employing a neural network to suggest a solution ρ_{sug}, which is then further optimised using a traditional algorithm.

The given setup forces the network to regularise the ill-conditioned problem by reproducing the correct training spectrum in accord with our criteria for a successful reconstruction. It is the inclusion of the training data and the free choice of loss functions that allows the network to fully absorb all relevant prior information. This ability is an outstanding property of supervised learning methods, which could yield potentially significant improvements over existing frameworks. for such constraints are the analytic structure of the propagator, asymptotic behaviors and normalisation constraints.

The parametrisation of an infinite set or manifold of inverse transformations by the neural network also enables the discovery of new loss functions which may be more appropriate for a reliable reconstruction. This includes, for example, the exploration of correlation matrices with adapted elements, in order to define a suitable norm for the given and suggested propagators. Existing, iterative methods may also benefit from the application of such adaptive loss functions. These may include parameters, point-like representations and arbitrary other characteristics of a given training sample.

Formulated in a Bayesian language, we set out to explicitly train the neural network to predict MAP estimates for each possible input propagator, given the training data as prior information. By salting the input data with noise, the network learns to return a denoised representation of the associated spectral functions.

12.2 A neural network based reconstruction

Neural networks provide high flexibility with regard to their structure and the information they can process. They are capable of learning complex internal representations which allow them to extract the relevant features from a given input. A variety of network architectures and loss functions can be implemented in a straightforward manner using modern Machine Learning frameworks. Prior information can be explicitly provided through a systematic selection of the training data. The data itself provides, in addition to the loss function, a regularisation of possible suggestions. Accordingly, the proposed solutions have the advantage to be similar to the ones in the training data.

The section starts with notes on the design of the neural networks we employ and ends with a detailed introduction of the training procedure and the utilised loss functions.

12.2.1 Design of the neural networks

We construct two different types of deep feed-forward neural networks. The input layer is fed with the noisy propagator data $G(p)$. The output for the first type is an estimate of the parameters of the associated ρ in the chosen basis, which we denote as *parameter net (PaNet)*. For the second type, the network is trained directly on the discretised representation of the spectral function. This network will be referred to as *point net (PoNet)*. A consideration of a variable number of Breit-Wigners is feasible per construction by the point-like representation of the spectral function within the output layer. This kind of network will in the following be abbreviated by *PoNetVar*. See Fig.12.2 for a schematic illustration of our strategy. Note that in all cases a basis for the spectral function is provided either explicitly through the structure of the network or implicitly through the choice of

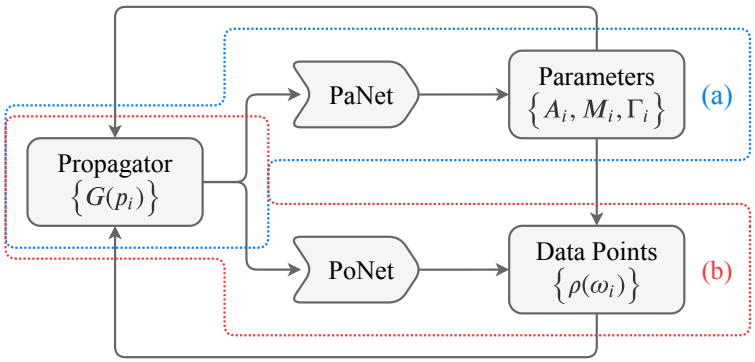

Figure 12.2: Sketch of our strategy for reconstructing (a) the parameters using the PaNet and (b) the discretised data points using the PoNet (and by extension also the PoNetVar). Details on the architectures are given in Sec. F.3.

the training data. If not stated otherwise, the numerical results presented in the following always correspond to results from the PaNet.

We compare the performance of fully-connected (FC) and convolutional (Conv) layers as well as the impact of their depth and width. In general, choosing the numbers of layers and neurons is a trade-off between the expressive power of the network, the available memory and the issue of overfitting. The latter strongly depends on the number of available training samples w.r.t. the expressivity. For fully parametrised spectral functions, new samples can be generated very efficiently for each training epoch, which implies an, in principle, infinite supply of data. Therefore, in this case, the risk of overfitting is practically non-existent. The specific dimensions and hyperparameters used for this work are provided in Sec. F.3. Numerical results can be found in Sec. 12.3.

12.2.2 Training strategy

The neural network is trained with appropriately labelled input data in a supervised manner. This approach allows to implicitly impose a prior distribution in the Bayesian sense. The challenge lies in constructing a training dataset that is exhaustive enough to contain the relevant structures that may constitute the actual spectral functions in practical applications.

From our past experience with hadronic spectral functions in lattice QCD and the functional renormalisation group, the most relevant structures are peaks of the Breit-Wigner type, as well as thresholds. The former present a challenge from the point of view of inverse problems, as they contain significant tail contributions, contrary e.g. to Gaussian peaks, which approach zero exponentially fast. Thresholds on the other hand set in at finite frequencies, often involving a non-analytic kink behavior. In this work, we only consider Breit-Wigner type structures as a first step for the application of neural networks to this family of problems.

Mock spectral functions are constructed using a superposed collection of Breit-Wigner peaks based on a parametrisation obtained directly from one-loop perturbative quantum field theory. Each individual Breit-Wigner is given by

$$\rho^{(BW)}(\omega) = \frac{4A\Gamma\omega}{(M^2 + \Gamma^2 - \omega^2)^2 + 4\Gamma^2\omega^2} \,. \tag{12.7}$$

Here, M denotes the mass of the corresponding state, Γ its width and A amounts to a positive normalisation constant.

Spectral functions for the training and test set are constructed from a combination of at most $N_{\mathrm{BW}} = 3$ different Breit-Wigner peaks. Depending on which type of network is considered, the Euclidean propagator is obtained either by inserting the discretised spectral function into (12.2), or by a computation of the propagator's analytic representation from the given parameters. The propagators are salted both for the training and test set with additive, Gaussian noise

$$G_i^{\mathrm{noisy}} = G_i + \epsilon\,. \tag{12.8}$$

This is a generic choice which allows to quantify the performance of our method at different noise levels.

The advantage of neural networks to have direct access to different representations of a spectral function implies a free choice of objective functions in the solution space. We consider three simple loss functions and combinations thereof. The (pure) propagator loss $L_G(\rho_{\mathrm{sug}})$ defined in (12.5) represents the most straightforward approach. This objective function is accessible also in already existing frameworks, such as Bayesian Reconstruction (BR) or Hamiltonian Monte Carlo (HMC) methods, in particular the GrHMC framework (referring to the retarded propagator G_r) developed in Ref. [339]. It is implemented in this work to facilitate a quantitative comparison. In contrast, the loss functions that follow are only accessible in the neural network based reconstruction framework. This unique property is owed to the possibility that a neural network can be trained in advance on a dataset of known input and output pairs. As pointed out in Sec. 12.1.3, a loss function can e.g. be defined directly on a discretised representation of the spectral function ρ. This approach is implemented through $L_\rho(\rho_{\mathrm{sug}})$, see (12.6). The optimisation of the parameters $\theta = \{A_i, M_i, \Gamma_i \,|\, 0 \leq i < N_{\mathrm{BW}}\}$ of our chosen basis is an even more direct approach. In principle, the space of all possible choices of parameters is $\mathbb{R}_+^{3\cdot N_{BW}}$, assuming they are all positive definite. Of course, only finite subvolumes of this space ever need to be considered as target spaces for reconstruction methods. Therefore, we will often refer to a finite target volume simply as the parameter space for a specific setting. Accordingly, in addition to the propagator and spectral function losses defined in Eqs. (12.5) and (12.6), the respective parameter loss in this space is given by:

$$L_\theta(\theta_{\mathrm{sug}}) = \|\theta_{\mathrm{sug}} - \theta\|\,. \tag{12.9}$$

All losses are evaluated using the 2-norm. In the case of the parameter net, we have $\rho_{\mathrm{sug}} \equiv \rho(\theta_{\mathrm{sug}})$. Apart from the three given loss functions, we also investigate a combination of the propagator and the spectral function loss,

$$L_{G,\rho}(\rho_{\mathrm{sug}}, \alpha) = L_\rho(\rho_{\mathrm{sug}}) + \alpha L_G(\rho_{\mathrm{sug}})\,, \tag{12.10}$$

where the parameter α determines the relative importance of the two losses. In our experiments, we have chosen it such as to roughly balance differences in the scales of the respective loss functions. The type of loss function that is employed as well as the selection of the training data have major impact on the resulting performance of the neural network. Given this observation, it seems likely that a further optimisation regarding the choice of the loss function can significantly enhance the prediction quality. However, for the time being, we content ourselves with the types given above and postpone the exploration of more suitable training objectives to future work.

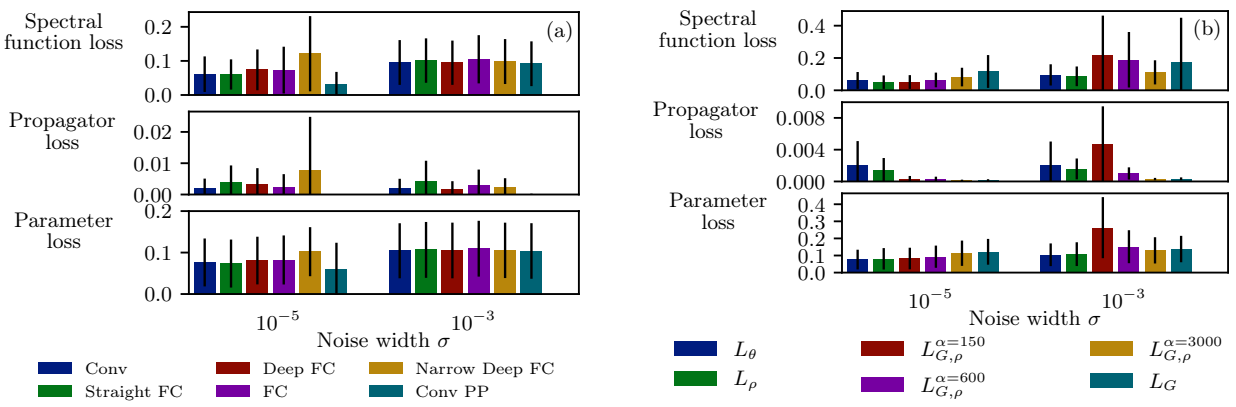

Figure 12.3: The performance of different net architectures and loss functions of a parameter net is compared for additive Gaussian noise with widths of 10^{-3} and 10^{-5} on the given input propagator: (a) Comparison of different net architecures. All networks are trained based on the parameter loss. The associated architectures can be found in Tab. F.2. (b) Comparison of different loss functions. Details on the loss functions are described at the end of Sec. 12.2.2. All results are based on networks with the architecture Conv. To (a) and (b): Shown here are the respective losses for the predicted parameters, for the discretised reconstructed spectral function and for the reconstructed propagator to the true, noise-free propagator. Both figures use the largest volume in parameter space, Vol O. The definitions of the performance measures are given at the end of Sec. F.3. The results on the left hand side imply that for larger errors, the choice of a specific network architecture has negligible impact on the quality of the reconstructions. All performance measures can be lowered for the given noise widths by applying a post-processing procedure on the suggested parameters of the network. In particular, the propagator loss can be minimised. The comparison on the right hand side shows that the choice of the loss function has major impact on the resulting performance of the network. The results underpin the importance of an appropriate loss function and support our argument of potential advantages of neural networks compared to existing approaches in Sec. 12.1.3. Contour plots in parameter space are illustrated for the respective measures in Fig. F.1 and Fig. F.2.

12.3 Numerical results

In this section we present numerical results for the neural network based reconstruction and validate the discussed potential advantages by comparing to existing methods. Details on the training procedure as well as the training and test datasets can be found in Tab. 12.1 and Sec. F.3, together with an introduction to the used performance measures. We start now with a brief summary of the main findings for our approach. Furthermore, a detailed numerical analysis and discussion of different network setups w.r.t performance differences are provided. Subsequently, additional post-processing methods for an improvement of the neural network predictions are covered. The section ends with a discussion of results from the PoNet. Readers who are interested in a comparison of the neural network based reconstruction to existing methods may proceed directly with Sec. 12.3.2.

12.3.1 Reconstruction with neural networks

Our findings concerning the optimal setup of a feed-forward network can be summarised as follows. As pointed out in Sec. 12.1.3, the network aims to learn an approximate parametrisation of a manifold of (matrix) inverses of the discretised Källén-Lehmann transformations. The inverse problem grows more severe if the propagator values are afflicted with noise. In Bayesian terms, this is caused by

a wider posterior distribution for larger noise. The network needs to have sufficient expressivity, i.e. an adequate number of hyperparameters, to be able to learn a large enough set of inverse transformations. We assume that for larger noise widths a smaller number of hyperparameters is necessary to learn satisfactory transformations, since the available information content about the actual inverse transformation decreases for a respective exact reconstruction. A varying severity of the inverse problem within the parameter space leads to an optimisation of the spectral reconstruction in regions where the problem is less severe. This effect occurs naturally, since there the network can minimise the loss more easily than in regions where the problem is more severe. Besides the severity of the inverse problem, the form of the loss function has a large impact on global optima within the landscape of the solution space. Based on these observations, an appropriate training of the network is non-trivial and demands a careful numerical analysis of the inverse problem itself, and of different setups of the optimisation procedure. A sensible definition of the loss function or a non-uniform selection of the training data are possible approaches to address the disparity in the severity of the inverse problem. A more straightforward approach is to iteratively reduce the covered parameter ranges within the learning process, based on previous suggested outcomes. This amounts to successively increasing the prediction accuracy by restricting the network to smaller and smaller subvolumes of the original solution space. However, one should be aware that this approach is only sensible if the reconstructions for different noise samples on the original propagator data are sufficiently close to each other in the solution space. A successive optimisation of the prediction accuracy in such a way can also be applied to existing methods. All approaches ultimately aim at a more homogeneous reconstruction loss within the solution space. This allows for a reliable control of systematic errors, as well as an accurate estimation of statistical errors. The desired outcome for a generic set of Breit-Wigner parameters is illustrated and discussed in Fig. 12.1.

The quality and reliability of reconstructions heavily depend on the following details of the training procedure and the inverse problem itself, with varying levels of impact given a specific situation:

- local differences in the severity of the inverse problem
- information loss in the forward pass and due to statistical noise
- loss function / prior information
- complexity / expressivity of the network architecture

In essence, we wish to emphasise that a reliable reconstruction is a multifactorial problem whose facets need to be disentangled in order to understand all contributions to systematic errors.

The impact of the net architecture and the loss function on the overall performance within the parameter space is illustrated in Fig. 12.3. Associated contour plots can be found in the appendix, see Fig. F.1 and Fig. F.2. These plots demonstrate that the minima in the loss landscape highly depend on the employed loss function. In turn, this leads to different performance measures. This observation confirms our previous discussion and the necessity of an appropriate definition of the loss function. It also reinforces our arguments regarding potential advantages of neural networks in comparison to other approaches for spectral reconstruction. The comparison of different feed-forward network architectures shows that the specific details of the network structure are rather irrelevant, provided that the expressivity is sufficient.

Differences in the performance of the networks that are trained with the same loss function become less visible for larger noise. This is illustrated by a comparison of contour plots with different noise widths, see e.g. Fig. F.1. The severity of the inverse problem grows with the noise and the information

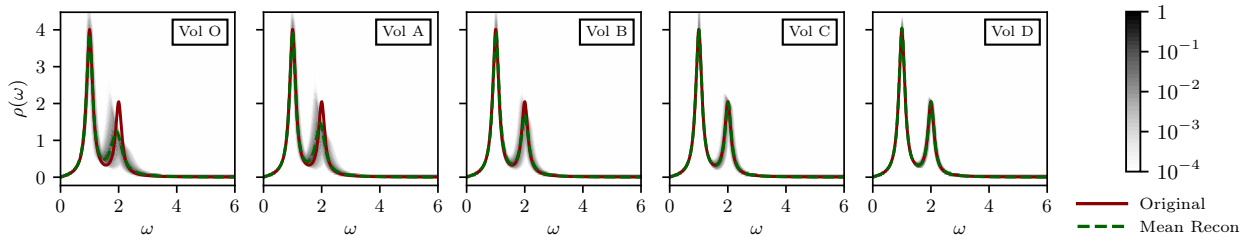

Figure 12.4: The uncertainties of reconstructions of spectral functions on the same original propagator are illustrated in the same manner as described in Fig. 12.1 for different volumes of the parameter space, again using a noise width of 10^{-3}. The plots demonstrate how the quality of the reconstruction improves if the parameter space which the network has to learn is decreased. The volumes of the corresponding parameter spaces are listed in Tab. 12.1. The results are computed from the Conv PaNet. The systematic deviation of the distribution of reconstructions for large volumes shows that the network has not captured the manifold of inverse transformations completely for the entire parameter space. This is in concordance with the results discussed in Fig. F.1 and Fig. F.3.

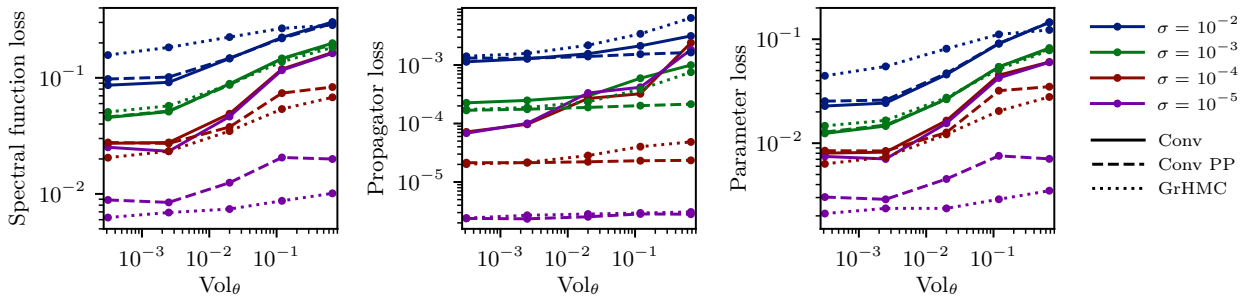

Figure 12.5: The plots in this figure quantify the impact of the parameter space volume used for the training on the performance of the parameter network. The performance measures are computed based on the test set of the smallest volume, Vol D. The parameter ranges in the training set are gradually reduced to analyse different levels of complexity of the problem. Separate networks are trained for each volume, which are listed in Tab. 12.1. The results demonstrate the potential advantage of an iterative restriction of the parameter ranges of possible solutions. The contour plots in Fig. F.3 depict changes of the performance measures within the parameter space. More strongly peaked prior distributions lead to better reconstructions. The comparison with results of the GrHMC approach illustrates the improvement of the performance of neural networks for larger errors and smaller volumes. These observations confirm the discussions of Fig. 12.4 and Fig. 12.8. Adding a post-processing step leads in particular for the propagator loss and for smaller noise widths to an improvement of the reconstruction, as has also been discussed in Fig. 12.3.

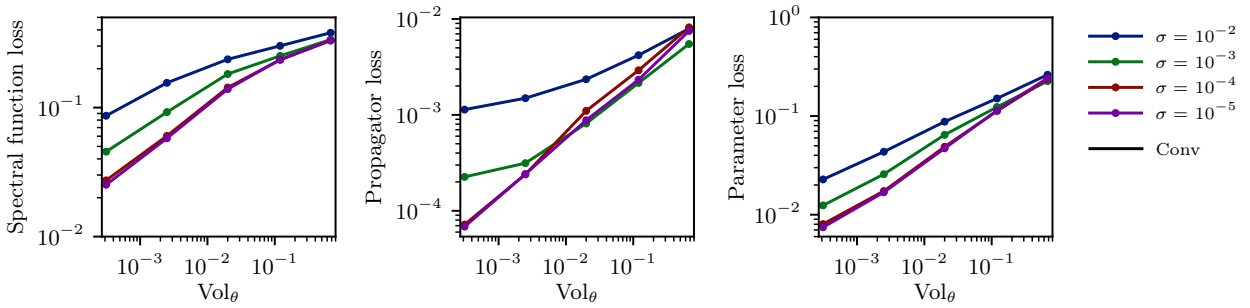

Figure 12.6: Comparison of reconstruction errors of the Conv PaNet trained only on the smallest Vol D for different noise levels, evaluated with test volumes which are also larger than D. The test datasets are equivalent to the ones used for the other tasks described in this paper. In contrast to Fig. 12.5, which shows the prediction quality as a function of the training volume with a fixed test volume Vol D, here the performance is evaluated as a function of the test volume using a fixed training volume.

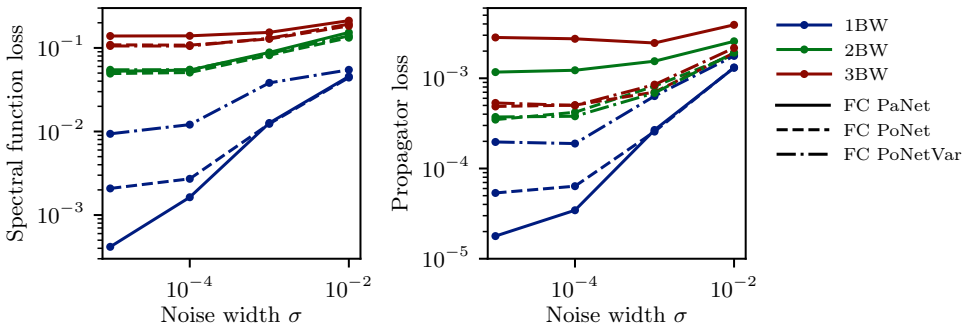

Figure 12.7: Comparison of reconstruction errors of the PaNet and the PoNet for the FC architecture. The performance measures are computed based on the test set of the largest parameter space volume Vol O for one, two and three Breit-Wigners. All networks are trained based on the parameter ranges of Vol O. Loss functions are the parameter loss L_θ for the PaNet and the spectral function loss L_ρ for both PoNets. The overall smaller losses for the point nets are due to the large number of degrees of freedom for the point-like representation of the spectral function. The partly competitive performance of the PoNetVar compared to the results of the PoNet encourage the further investigation of networks that are trained using a more exhaustive set of basis functions to describe physical structures in the spectral functions.

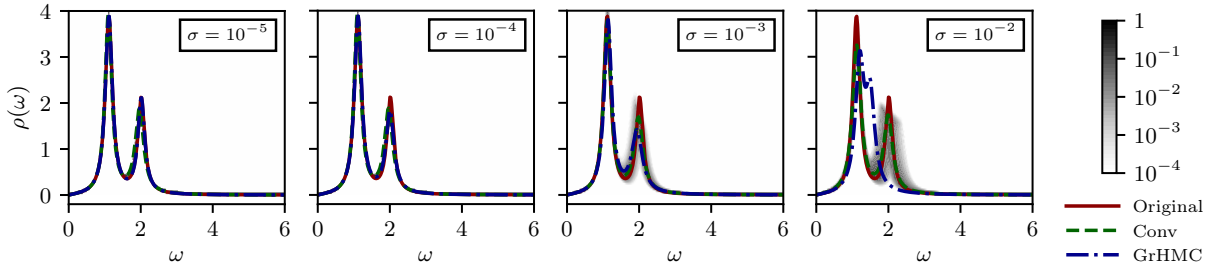

Figure 12.8: The quality of the reconstruction of two Breit-Wigner peaks is compared for different strength of additive noise on the same propagator. The labels indicate the noise width on the original propagator. It can be seen that the reconstructed spectral function of the neural network exhibits in particular for larger errors a lower deviation to the original spectral function than the GrHMC method. This mirrors the in general observable better performance of the neural network for larger errors, as can be seen in Fig. 12.5 and in Fig. 12.9. The green and the red curve correspond to reconstructions of the Conv PaNet and the GrHMC method for the same given noisy propagator. The prior is in both cases given by the parameter range of volume Vol B. The uncertainty of the reconstructions for the neural network is depicted by the grey shaded areas as described in Fig. 12.1. For small errors, this area is covered by the corresponding reconstructed spectral functions.

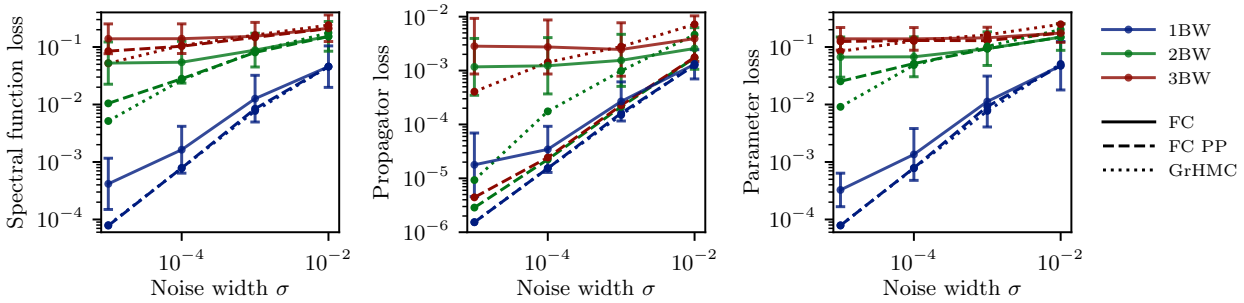

Figure 12.9: The performance of the reconstruction of spectral functions is benchmarked for the parameter network, which is trained with L_θ, by a comparison to results of the GrHMC method. The parameter network is in particular for large noise widths competitive. The worse performance for smaller noise widths is a result of an inappropriate training procedure and a too low expressive power of the neural network. The problems are caused by a varying severity of the inverse problem and by a too large parameter space that needs to be covered by the neural network, as discussed in Sec. 12.3.1. The error bars of the results for the FC network are representative for typical errors within all methods and plots of this kind.

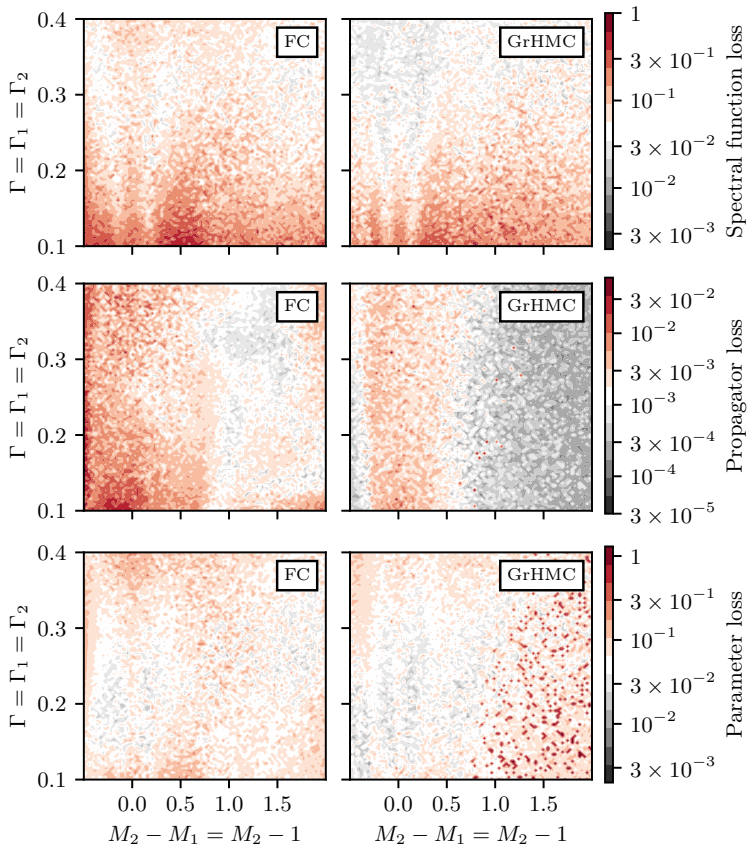

Figure 12.10: Comparison of performance measures for the reconstruction of two Breit-Wigners of the FC parameter network with the GrHMC method for input propagators with noise width 10^{-3} within the parameter space volume Vol O. The similar loss landscape emphasises the high impact of variations of the severity of the inverse problem within the parameter space on the quality of reconstructions. Contrary to expectations, the parameter network mimics, despite an optimisation based on the parameter loss L_θ, the reconstruction of the GrHMC method which relies on an optimisation of the propagator loss L_G with respect to the parameters. A reconstruction resulting in an averaged peak with the other parameter set effectively removed, as outlined in Ref. [339], results in the spiking parameter losses for the GrHMC reconstructions with large errors.

content about the actual matrix transformation decreases. These properties lead to the observation of a generally worse performance for larger noise widths, as can be inferred from Fig. 12.4, as well as Figs. 12.8 and 12.9, which are discussed later. They also imply that for specific noise widths, the neural network possesses enough hyperparameters to learn a sufficient parametrisation of the inverse transformation manifold. Furthermore, the local optima into which the network converges are mainly determined by differences in the local severity of the inverse problem. Hence, the issue remains that generic loss functions are inappropriate to address the varying local severity of the inverse problem. This issue implies the existence of systematic errors for particular regions within the parameter space, as can be seen e.g. in the left plot of Fig. 12.4.

The results shown in Figs.12.4 and 12.5 as well as Fig. F.3 in the appendix confirm our discussion regarding the expressive power of the network w.r.t. the complexity of the solution space and the decreasing information content for larger errors. The parameter space is gradually reduced, effectively increasing the expressivity of the network relative to the severity of the problem and improving the

Vol	A	M	Γ	ΔM
O	$[0.1, 1.0]$	$[0.5, 3.0]$	$[0.1, 0.4]$	$[0.0, 2.5]$
A	$[0.3, 0.7]$	$[0.5, 3.0]$	$[0.1, 0.3]$	$[0.25, 1.75]$
B	$[0.4, 0.6]$	$[0.5, 3.0]$	$[0.1, 0.2]$	$[0.5, 1.5]$
C	$[0.45, 0.55]$	$[0.5, 3.0]$	$[0.1, 0.15]$	$[0.75, 1.25]$
D	$[0.475, 0.525]$	$[0.5, 3.0]$	$[0.1125, 0.1375]$	$[0.875, 1.125]$

Table 12.1: Parameter ranges of the different volumes in parameter space used for training. Parameters are sampled uniformly based on the given bounds for the training and test sets. For the case of two and three Breit-Wigner functions, the difference in mass $\Delta M = M_2 - M_1$ is limited to restrict the minimum possible distance between two peaks. The volumes V_θ are computed based on these parameter ranges.

behavior of the loss function for a given fixed parameter space. The respective volumes are listed in Tab. 12.1. Shrinking the parameter space leads to a more homogeneous loss landscape due to the increased locality, thereby mitigating the issue of inappropriate loss functions. The necessary number of hyperparameters decreases for larger noise widths and smaller parameter ranges in the training and test dataset. The arguments above imply a better performance of the network for smaller parameter spaces. A reduction of the parameter space effectively corresponds to a sharpening of the prior information, which also has positive effects on the spread of the posterior distribution. More detailed discussions on the impact of different elements of the training procedure can be found in the captions of the respective figures.

Since increasing the expressivity of the network is limited by the computational demand required for the training, one can also apply post-processing methods to improve the suggested outcome w.r.t. the initially given, noisy propagator. These methods are motivated by the in some cases large observed root-mean-square deviation of the reconstructed suggested propagator to the input, see for example again Fig. 12.3. The application of standard optimisation methods on the suggested results of the network represents one possible approach to address this problem. Here, the network's prediction is interpreted as a guess of the MAP estimate, which is presumed to be close to the true solution. For the PaNet, we minimise the propagator loss a posteriori with respect to the following loss function:

$$\min_{\theta_{\text{sug}}} L_{\text{PP}}[\theta_{\text{sug}}] = \min_{\theta_{\text{sug}}} \left\| G_{\text{noisy}} - G\left[\rho\left(\theta_{\text{sug}}\right)\right] \right\| . \tag{12.11}$$

This ensures that suggestions for the reconstructed spectral functions are in concordance with the given input propagator. Results obtained with an additional post-processing are marked by the attachment PP in this work. The numerical results in Figs. 12.4 and 12.9 show that the finite size of the neural network can be partially compensated for small errors. The resulting low propagator losses are noteworthy, and are close to state-of-the-art spectral reconstruction approaches. One reason for this similarity is the shared underlying objective function. However, the situation is different for larger noise widths. For our choice of hyperparameters, the algorithm quickly converges into a local minimum. For large noise widths, the optimisation procedure may even lead to worse results than the initially suggested reconstruction. This is due to the already mentioned systematic deviations which are caused by the inappropriate choice of the loss function for large parameter spaces. This kind of post-processing should therefore be applied with caution, since it may cancel out the potential advantages of neural networks w.r.t. the freedom in the definition of the loss function.

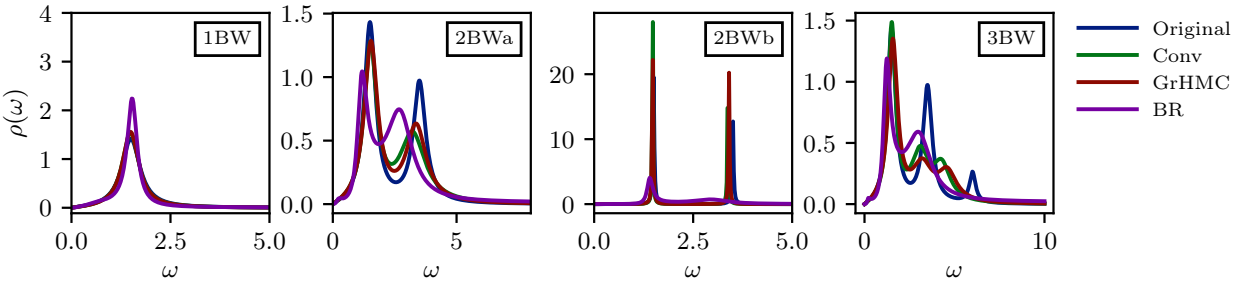

Figure 12.11: Reconstructions of one, two and three Breit-Wigners are compared for our proposed neural network approach, the GrHMC method and the BR method. The reconstructions of the first two methods are based on a single sample with noise width 10^{-3}, while the results of the BR method are obtained from multiple samples with larger errors, but an average noise width of 10^{-3} as well. In contrast to the previous plots, the neural network and the GrHMC method now use different priors for each case in order to allow for a reasonable comparison with the BR method, see Tab. F.1. We observe that all approaches qualitatively capture the features in the spectral function. Due to the comparably large error on the input data, all methods are expected to face difficulties in finding an accurate solution. The reconstructions of the neural network approach and the GrHMC method are comparable, whereas the BR method struggles in particular with thin peaks and the three Breit-Wigner case. The results demonstrate that, generally, using suitable basis functions and incorporating prior information lead to a superior reconstruction performance.

The following alternative post-processing approach preserves the potential advantages of neural networks while nevertheless minimising the propagator loss. The idea is to include the network into the optimisation process through the following objective:

$$\min_{G_{\text{input}}} L_{\text{input}}[G_{\text{input}}] = \min_{G_{\text{input}}} \left\| G_{\text{noisy}} - G\left[\rho\left(\theta_{\text{sug}}\right)\right] \right\|, \tag{12.12}$$

where G_{input} corresponds to the input propagator of the neural network and θ_{sug} to the associated outcome. This facilitates a compensation of badly distributed noise samples and allows a more accurate error estimation. The approach is only sensible if no systematic errors exist for reconstructions within the parameter space, and if the network's suggestions are already somewhat reliable. We postpone a numerical analysis of this optimisation method together with the exploration of more appropriate loss functions and improved training strategies to future work, due to a currently lacking setup to train such a network.

Figs. 12.6 and F.4 serve to quantify the generalization capability of our neural network approach with regard to data that lie outside of the training volume. Fig. 12.6 shows a comparison of error metrics obtained with the Conv PaNet trained only on the smallest Vol D when applied to larger parameter volumes, at a few different noise levels. As we expect, the performance decreases with larger test volumes, but loss values notably remain in the same orders of magnitude that are observed for larger training volumes, indicating that the generalization capability of the network has at first only a rather weak dependence on the ratio between test and training volume, which only grows more severe if the test volume becomes much larger. This is further illustrated by Fig. F.4, showing a rather flat loss landscape in the immediate vicinity of the training volume boundaries without sharp transitions, and gradual worsening of the prediction quality as one moves further away. We conclude that our approach is moderately robust against deviations of the true solution from the considered training volume and only fails at larger distances.

In Figs. 12.7 and F.5, results of the PoNet and the PaNet are compared. We observe that spectral reconstructions based on the PoNet structure suffer from similar problems as the PaNet. The point-like representation of the spectral function introduces a large number of degrees of freedom for the solution space. The training procedure implicitly regularises this problem, however, a visual analysis of individual reconstructions shows that in some cases the network struggles with common issues known from existing methods, such as partly non-Breit-Wigner like structures and wiggles. An application of the proposed post-processing methods serves as a possible approach to circumvent such problems. An inclusion of further regulator terms into the loss function, concerning e.g. the smoothness of the reconstruction, is also possible.

12.3.2 Benchmarking and discussion

Lastly, we want to emphasise differences of our proposed neural network approach to existing methods. Our arguments are supported by an in-depth numerical comparison.

Within all approaches the aim is to map out, or at least to find the maximum of, the posterior distribution $P(\rho|G)$ for a given noisy propagator G. The BR and GrHMC methods represent iterative approaches to accomplish this goal. The algorithms are designed to find the maximum for each propagator on a case-by-case basis. The GrHMC method additionally provides the possibility to implement constraints on the functional basis of the spectral function in a straightforward manner. In contrast, a neural network aims to learn the full manifold of inverse Källen-Lehman transformations for any noisy propagator (at least within the chosen parameter space). In this sense, it needs to propose for each given propagator an estimate of the maximum of $P(\rho|D)$. A complex parametrisation, as given by the network, an exhaustive training dataset and the optimisation procedure itself are essential features of this approach for tackling this tough challenge. The computational effort to find a solution in an iterative approach is therefore shifted to the training process as well as the memory demand of the network. Accordingly, the neural network based reconstruction can be performed much faster after training has been completed, which is in particular advantageous when large sets of input propagators are considered. In our experiments, the time required for running the GrHMC algorithm and training the networks was roughly similar, generally being of the order of a few hours. A quantitative comparison is difficult due to the use of completely distinct software packages and different utilisation of hardware accelerators. However, we emphasise again that one run of the GrHMC can only provide a prediction for one specific propagator, whereas the trained network can be evaluated quickly on large datasets and additionally allows fast retraining when different data are expected, without having to start from scratch.

The numerical results in Fig. 12.5 and Fig. 12.8 to Fig. 12.11 demonstrate that the formal arguments of Sec. 12.1.3 apply, particularly for comparably large noise widths as well as smaller parameter ranges. For both cases, the network successfully approximates the required inverse transformation manifold. Smaller noise widths and a larger set of possible spectral functions can be addressed by increasing the number of hyperparameters and through the exploration of more appropriate loss functions, as was already discussed previously.

12.4 Summary

In this chapter we have explored artificial neural networks as a tool to deal with the ill-conditioned inverse problem of reconstructing spectral functions from noisy Euclidean propagator data. We

systematically investigated the performance of this approach on physically motivated mock data and compared our results with existing methods. Our findings demonstrate the importance of understanding the implications of the inverse problem itself on the optimisation procedure as well as on the resulting predictions.

The crucial advantage of the presented ansatz is the superior flexibility in the choice of the objective function. As a result, it can outperform state-of-the-art methods if the network is trained appropriately and exhibits sufficient expressivity to approximate the inverse transformation manifold. The numerical results demonstrate that defining an appropriate loss function grows increasingly important for an increased variability of considered spectral functions and of the severity of the inverse problem.

In future work, we aim to further exploit the advantage of neural networks that local variations in the severity of the inverse problem can be systematically compensated. The goal is to eliminate systematic errors in the predictions in order to facilitate a reliable reconstruction with an accurate error estimation. This can be realised by finding more appropriate loss functions with the help of implicit and explicit approaches [344, 345]. A utilisation of these loss functions in existing methods is also possible if they are directly accessible. Varying the prior distribution will also be investigated, by sampling non-uniformly over the parameter space during the creation of the training data. Furthermore, we aim at a better understanding of the posterior distribution through the application of invertible neural networks [215]. This novel architecture provides a reliable estimation of errors by mapping out the entire posterior distribution by construction.

In conclusion, we believe that the suggested improvements will boost the performance of the proposed method to an as of yet unprecedented level and that neural networks will eventually replace existing state-of-the-art methods for spectral reconstruction.

CHAPTER 13

Conclusion

In this thesis, we approached several problems in quantum chromodynamics and statistical physics by bringing together state-of-the-art techniques from different disciplines. Novel insights, originating from the interdisciplinary nature of this work, unveil starting points for tackling unsolved problems of numerical, conceptional and mathematical nature in multiple domains. We established foundations for employing methods and tools ranging from deep learning, statistics and neuromorphic computing to eventually discover new physics in the long run. The derived methods and approaches are built upon leveraging well-established tools such as probability theory, stochastic processes and Markov chain Monte Carlo algorithms.

Driven by the similarity of Langevin dynamics and LIF dynamics, we explored in Chapter 4 and 5 the BrainScaleS-2 chip as an alternative more-energy efficient and scalable computing device for the numerical computation of physical systems by means of Langevin dynamics. A direct implementation of the dynamics turned out to be difficult due to hardware-related restrictions. This concerns in particular the inherent interaction on the hardware via spikes and related non-linearities in transmitted signals. The former property renders the system of neurons in the spiking-mode effectively a discrete system.

Bearing these limitations in mind, we derived in Chapter 3 the Langevin equation for discrete systems. The algorithm is shown to be equivalent to Langevin dynamics in the limit of infinitesimally small step sizes in configuration space, thus, establishing the algorithm as the complement to Langevin dynamics for a discrete system. While complex Langevin dynamics was derived in the same limit, the formulation of a respective algorithm for discrete systems with a complex action remains unsolved. A possible approach to tackle this problem is provided by the novel insights on complex Langevin dynamics in Chapter 7, stating important mathematical foundations for sampling algorithms extending their dynamics to the complex plane.

In Chapter 4, we studied a numerically exact implementation of a Boltzmann machine and the Ising model by means of the Langevin equation for discrete systems based on simplified model descriptions of LIF dynamics. A comparison with the standard way to implement these systems shows that the new kind of dynamics has the potential to integrate Boltzmann-distributed systems in a more appropriate way. An actual implementation on the BrainScaleS-2 chip is subject to future work.

Taking into account current hardware-related restrictions, we continued in Chapter 5 with a discussion of necessary properties of potential alternative computing devices. Thereby, we point out a natural restriction of neural network-inspired devices to linear or non-trivial non-linear interactions. The proposed shift of the dynamics into the weights of a neural network provides a solution allowing at

least partially an implementation of Langevin dynamics on neuromorphic hardware systems. The BrainScaleS-2 chip features such an implementation in the non-spiking mode based on a hybrid setup. In the long run, we expect this to foster the implementation of comparably large physical systems built upon a fast and energy-efficient evaluation of computationally expensive subtasks within the update rules of Langevin dynamics.

We continued in Chapter 6 with a machine learning task on the BrainScaleS-2 chip. The hardware emulates restricted Boltzmann machines which, in turn, represent entangled quantum states. The successful encoding of non-classical Bell correlations and the obtained high quantum fidelities demonstrate that the accuracy of the neuromorphic system is well-suited for representing and training Boltzmann distributed systems in a machine learning framework. Thus, the results open up new perspectives for simulating quantum many-body systems with spiking neuromorphic devices.

Chapters 7 to 9 constitute the second main building block of this work. The proposed sampling frameworks aim to solve the problem of an often observed convergence of complex Langevin dynamics to unphysical solutions. Starting from first principles, we established in Chapter 7 constraints on sampling processes facilitating a sampling of the physically correct solutions. The constraints are built on firm grounds by techniques of Markov chain Monte Carlo methods which warrant, as opposed to complex Langevin dynamics, explicit control of the underlying sampling process of complex action problems. Thereby, we avoided the standard comparison with a Fokker-Planck equation and provide a mathematically well-founded derivation in terms of the proposed framework, cf. Chapter 8. The presented framework represents a fundamental paradigm shift and opens a path for the systematic development of tailor-made sampling schemes for tackling the sign problem of complex quantum systems. While several new samples schemes were derived as a proof of concept of the provided framework, no algorithm with properties being significantly different to complex Langevin dynamics has been found yet. A respective search is subject of future work.

The appeal of such a Markov chain Monte Carlo framework for complex action problems emerges from the following benefits: The explicit construction of transition probabilities might be the key for a potential solution to the problem of wrong convergence. In contrast to complex Langevin dynamics, this allows the implementation of sampling processes with finite step sizes in the configuration space. Thus, the process does no longer rely on the drift term and the respective classical flow field. Instead, it is driven directly by the action of the considered model. This might resolve problems of possible runaway trajectories within the sampling process and supports a faster and a full exploration of the configuration space combined with a lower autocorrelation time. Lastly, we want to point out that there is the possibility for improving the theoretical framework. A respective deeper understanding of the framework can lead to less restrictive constraints simplifying the search for novel Markov chain Monte Carlo sampling algorithms for complex action problems.

Chapter 9 follows a different approach for dealing with a wrong convergence of complex Langevin dynamics. The machine learning-driven ansatz explores in a self-supervised learning framework the space of actions by a mutual application of complex Langevin dynamics and reweighting in the complex plane. In a step-by-step manner, it aims to find an auxiliary action for which complex Langevin dynamics samples the correct expectation values and where reweighting to the desired action is feasible. The self-consistent sampling algorithm makes use of the step-wise reweighting criterion for correctness. The novel criterion provides an easy-to-compute measure for verifying whether complex Langevin dynamics samples from the physically correct distribution. Utilizing the criterion in a machine learning framework is not restricted to the presented one. We expect that there are versatile applications of the presented set-up, also with respect to other algorithms tackling the sign problem, as Lefschetz thimbles, for example.

In the third main building block of the thesis, consisting of Chapter 10 and Chapter 11, we approach the sign problem from a different perspective relying on an analysis of the physics and the properties of the given model itself by machine learning tools. Extracting information about hidden structures and correlations in large data sets supports the discovery of new observables and of yet unnoticed perspectives on the data itself. Besides the gained insights into the considered physical system, respective findings can be used in a second step for developing and improving simulation algorithms allowing an exploration of hitherto unsolved problems in quantum chromo-dynamics and beyond. In future work, we aim to investigate the proposed approaches by interpretable and explainable machine learning methods in more depth, cf. Chapter 11.

The presented approach for learning lower-dimensional representations in Chapter 10 reveals many possible applications in different areas, going beyond the proposed embedding of graph-structured data and lattice configurations. This concerns in particular the generic idea of a mutual training of two opponents in a zero-sum game based on the same loss term. The resulting implicit maximization of the respective loss term is expected to have versatile applicability in semi-supervised and unsupervised training frameworks.

We continued in Chapter 12 with a further application of deep learning. We proposed a solution to the ill-conditioned problem of spectral reconstruction by means of a supervised training framework. The observed large variations in the severity of the inverse problems leave room for more sophisticated Bayesian frameworks such as invertible neural networks [215] and Gaussian processes, as proposed recently in Ref. [346], and for potential improvements by loss functions optimized for the specific problem at hand [344, 345].

All in all, this thesis covers several topics and approaches related to the simulation and interpretation of physical systems that constitute a fertile ground for future developments in various areas of research. This includes the theoretical and numerical evaluation of enhanced and more-energy efficient sampling algorithms on neuromorphic computing devices as well as promising developments centered around an exploration of the QCD phase diagram and further problems ranging from spectral reconstruction to the embedding of graph-structured data. The omnipresence of deep learning and machine learning algorithms throughout this work reinforces the still growing importance of these tools in today's research. Furthermore, the potential of interdisciplinary research to deliver significant contributions in various domains is underpinned by parallels and similarities found despite the high diversity of problems studied in this thesis.

APPENDIX A

Langevin dynamics and discrete systems

This appendix is based on Ref. [1].

A.1 Transition probability of the Langevin equation

In the following, we derive transition probabilities of the discrete Langevin equation. These are used in Sec. 2.2.2 for an interpretation of the dynamics as a standard Monte Carlo algorithm. Starting from the discrete Langevin equation

$$\phi' = \phi - \epsilon \frac{\delta S}{\delta \phi_x} + \sqrt{\epsilon}\eta\,, \tag{A.1}$$

with: $\phi := \phi(\tau)$ and $\phi' := \phi(\tau + \epsilon)$, it is straightforward to compute the transition probabilities of an infinitesimal change,

$$W(\phi \to \phi') = \frac{1}{\sqrt{2\epsilon}} \varphi\left(\frac{\phi' - \phi}{\sqrt{2\epsilon}} + \sqrt{\frac{\epsilon}{2}} \frac{\delta S}{\delta \phi} \right) . \tag{A.2}$$

By inserting the standard normal distribution $\varphi(x) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{x^2}{2}\right]$ and by computing the square in the exponent, one obtains

$$\begin{aligned} W(\phi \to \phi') &= \frac{1}{\sqrt{4\pi\epsilon}} \exp\left[-\frac{1}{2} \left(\frac{\phi' - \phi}{\sqrt{2\epsilon}} + \sqrt{\frac{\epsilon}{2}} \frac{\delta S}{\delta \phi} \right)^2 \right] \\ &= \varphi\left(\frac{\phi' - \phi}{\sqrt{2\epsilon}} \right) \exp\left[-\frac{\phi' - \phi}{2} \frac{\delta S}{\delta \phi} + \mathcal{O}(\epsilon) \right] . \end{aligned} \tag{A.3}$$

With the identifications $\delta\phi \simeq \phi' - \phi$ and $\delta S \simeq S(\phi') - S(\phi)$, this can be further simplified to

$$W(\phi \to \phi') = \frac{1}{\sqrt{2\epsilon}} \varphi\left(\frac{\phi' - \phi}{\sqrt{2\epsilon}} \right) \exp\left[-\frac{S(\phi') - S(\phi)}{2} + \mathcal{O}(\epsilon) \right] . \tag{A.4}$$

Apparently, the transition probability satisfies the detailed-balance equation (2.15) since the first factor is symmetric to an exchange of ϕ' and ϕ,

$$\frac{W(\phi \to \phi')}{W(\phi' \to \phi)} = \exp\left[-(S(\phi') - S(\phi)) \right] . \tag{A.5}$$

A.2 Relations between the cumulative normal distribution and the exponential function

A relation between the exponential function of the transition probability and the cumulative normal distribution is needed in order to define an update formalism in terms of a Heaviside function and a Gaussian noise term. It is needed to be able to accept or reject a proposal state, see Sec. 3.1. Such a relation exists and is given by

$$\lim_{\epsilon\to 0} n_{\epsilon,0}(x) = \lim_{\epsilon\to 0} \frac{\Phi\left(-\frac{1}{\sqrt{\epsilon}} + \sqrt{\epsilon}\frac{x}{\lambda_\epsilon}\right)}{\Phi(-\frac{1}{\sqrt{\epsilon}})} = \exp(x) + \mathcal{O}(\epsilon x^2)\,, \tag{A.6}$$

with a scaling factor

$$\lambda_\epsilon = \frac{\sqrt{\epsilon}\varphi\left(-\frac{1}{\sqrt{\epsilon}}\right)}{\Phi\left(-\frac{1}{\sqrt{\epsilon}}\right)}\,, \tag{A.7}$$

and where $\Phi(\cdot)$ denotes the cumulative Gaussian distribution

$$\Phi(x) = \int_{-\infty}^{x} dt \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{t^2}{2}\right]\,. \tag{A.8}$$

The relation can be extended to the m-th derivative of the cumulative distribution with $m > 0$, according to

$$\lim_{\epsilon\to 0} n_{\epsilon,m}(x) = \lim_{\epsilon\to 0} \frac{\frac{\partial^m}{\partial t^m}\Phi\left(-\frac{1}{\sqrt{\epsilon}} + \sqrt{\epsilon}t\right)\Big|_{t=x/\sigma_{m,\epsilon}}}{\frac{\partial^m}{\partial t^m}\Phi(-\frac{1}{\sqrt{\epsilon}} + \sqrt{\epsilon}t)\big|_{t=0}} = \exp(x) + \mathcal{O}(\epsilon x^2)\,, \tag{A.9}$$

where the scaling factor $\sigma_{m,\epsilon}$ is defined as

$$\sigma_{m,\epsilon} = -\frac{\sqrt{\epsilon}\mathrm{He}_m\left(-\frac{1}{\sqrt{\epsilon}}\right)}{\mathrm{He}_{m-1}\left(-\frac{1}{\sqrt{\epsilon}}\right)}\,, \tag{A.10}$$

and where $\mathrm{He}_m(x)$ denote the m-th probabilists' Hermite polynomials. Fig. A.1 illustrates the dependence of the two relations (A.6) and (A.9) on the parameter ϵ and on x.

For $m = 1$, the scaling factor $\sigma_{m,\epsilon}$ evaluates to one and the relation (A.9) simplifies to

$$\lim_{\epsilon\to 0} n_{\epsilon,1}(x) = \lim_{\epsilon\to 0} \frac{\varphi\left(-\frac{1}{\sqrt{\epsilon}} + \sqrt{\epsilon}x\right)}{\varphi\left(-\frac{1}{\sqrt{\epsilon}}\right)} = \exp(x) + \mathcal{O}(\epsilon x^2)\,, \tag{A.11}$$

which can also be derived by a similar computation as in App. A.1.

In the following, we provide a detailed derivation of the relations (A.6) and (A.9). For reasons of readability, $\sqrt{\epsilon}$ is abbreviated by ε and the shorthand notation $\frac{\partial^n}{\partial x^n} = \partial^n$ is used in the following.

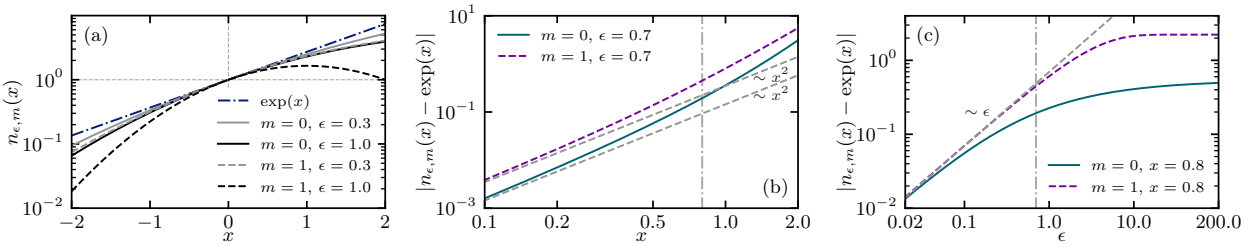

Figure A.1: Illustrations regarding the limit $\lim_{\epsilon\to 0} n_{\epsilon,m}(x)$ of relations (A.6) and (A.9) for $m = 0$ and $m = 1$: (a) Comparison to the exponential function. (b) Dependency on x for a fixed $\epsilon = 0.7$. (c) Dependency on ϵ for a fixed value of $x = 0.8$. (a)-(c): The vertical lines in (b) and (c) indicate the respective fixed value of ϵ and x. In general, the limit of the cumulative Gaussian distribution ($m = 0$) has a lower deviation for equal values of ϵ then the limit of the Gaussian distribution ($m = 1$) for $\epsilon \to 0$.

We start with a Taylor series around $x = 0$ of the m-th derivative of the cumulative Gaussian contribution,

$$\partial^m \Phi\left(-\frac{1}{\varepsilon} + \varepsilon x\right) = \sum_{n=0}^{\infty} \frac{1}{n!} \partial^{n+m} \Phi\left(-\frac{1}{\varepsilon} + \varepsilon x\right)\Big|_{x=0} x^n \,. \tag{A.12}$$

The following important identity between the cumulative Gaussian distribution $\Phi(x)$ and the probabilists' Hermite polynomials $\mathrm{He}_n(x)$ is useful for an evaluation of the Taylor expansion:

$$\partial^{n+m} \Phi\left(-\frac{1}{\varepsilon} + \varepsilon x\right) = (-\varepsilon)^{n+m-1} \mathrm{He}_{n+m-1}\left(-\frac{1}{\varepsilon} + \varepsilon x\right) \partial\Phi\left(-\frac{1}{\varepsilon} + \varepsilon x\right) , \tag{A.13}$$

for $n > 0$. Since this relation holds only for $n > 0$, the cases for $m = 0$ and $m > 0$ have to be treated separately, leading eventually to the relations (A.6) and (A.9).

Evaluation for $m = 0$

By inserting relation (A.13) into the Taylor expansion (A.12) and setting $m = 0$, one yields

$$\begin{aligned}
&\Phi\left(-\frac{1}{\varepsilon} + \varepsilon x\right) \\
&\quad = \Phi\left(-\frac{1}{\varepsilon}\right) + \sum_{n=1}^{\infty} \frac{1}{n!} (-\varepsilon)^{n-1} \mathrm{He}_{n-1}\left(-\frac{1}{\varepsilon} + \varepsilon x\right)\Big|_{x=0} \partial\Phi\left(-\frac{1}{\varepsilon} + \varepsilon x\right)\Big|_{x=0} x^n \\
&\quad = \Phi\left(-\frac{1}{\varepsilon}\right) + \sum_{n=1}^{\infty} \frac{1}{n!} (-1)^{n-1} \varepsilon^n \mathrm{He}_{n-1}\left(-\frac{1}{\varepsilon}\right) \varphi\left(-\frac{1}{\varepsilon}\right) x^n \,.
\end{aligned} \tag{A.14}$$

A comparison with the Taylor expansion of $\exp(x) = 1 + x + \mathcal{O}(x^2)$ shows that the first two terms in the above expression can be fixed by a division of the entire equation by $\Phi\left(-\frac{1}{\varepsilon}\right)$ and an additional rescaling of x by

$$\lambda(\varepsilon) = \frac{\varepsilon\, \varphi\left(-\frac{1}{\varepsilon}\right)}{\Phi\left(-\frac{1}{\varepsilon}\right)} \,. \tag{A.15}$$

This gives

$$\frac{\Phi\left(-\frac{1}{\varepsilon}+\varepsilon\frac{x}{\lambda(\epsilon)}\right)}{\Phi\left(-\frac{1}{\varepsilon}\right)}=1+x+\sum_{n=2}^{\infty}\frac{1}{n!}\frac{(-1)^{n-1}\varepsilon^{n-1}\mathrm{He}_{n-1}\left(-\frac{1}{\varepsilon}\right)}{\lambda(\varepsilon)^{n-1}}x^n\,. \tag{A.16}$$

It remains to be shown that the fractional factor converges to 1 for $\epsilon\to 0$ and for arbitrary values of n. This is done in two steps. First, we argue that $\lim_{\varepsilon\to 0}\lambda(\varepsilon)=1+\mathcal{O}(\varepsilon^2)$ and, second, a limit is derived for the fractional factor.

The limit of $\lim_{\varepsilon\to 0}\lambda(\varepsilon)$ can be derived by showing the identity $\lim_{\varepsilon\to 0}\Phi\left(\frac{1}{\varepsilon}\right)=\lim_{\varepsilon\to 0}\varphi\left(\frac{1}{\varepsilon}\right)$. By the substitution $u:=\frac{1}{x}$ and a subsequent partial integration, one finds that a second order term in $x=\frac{1}{u}$ vanishes and arrives directly at the identity, which entails that

$$\lim_{\varepsilon\to 0}\lambda(\varepsilon)=1+\mathcal{O}(\varepsilon^2)\,. \tag{A.17}$$

Since the highest order term of the n-th probabilists' Hermite polynomial equals x^n, it can be directly concluded that

$$\lim_{\varepsilon\to 0}\varepsilon^n\mathrm{He}_n\left(-\frac{1}{\varepsilon}\right)=(-1)^n+\mathcal{O}(\varepsilon^n)\,. \tag{A.18}$$

Using the Taylor expansion

$$\frac{1}{(1+x)^{n-1}}=1-(n-1)x+\mathcal{O}(x^2)\,, \tag{A.19}$$

and inserting the two limits (A.17) and (A.18), one arrives at the following limit for the fractional factor:

$$\lim_{\varepsilon\to 0}\frac{(-1)^{n-1}\varepsilon^{n-1}\mathrm{He}_{n-1}\left(-\frac{1}{\varepsilon}\right)}{\lambda(\varepsilon)^{n-1}}=1+\mathcal{O}(\varepsilon^2)\,. \tag{A.20}$$

The final limit between the cumulative normal distribution and the exponential function can be stated with the corresponding order of accuracy,

$$n_{\varepsilon^2,0}(x)=\frac{\Phi\left(-\frac{1}{\varepsilon}+\varepsilon\frac{x}{\lambda(\varepsilon)}\right)}{\Phi\left(-\frac{1}{\varepsilon}\right)}=\exp(x)+\mathcal{O}(\varepsilon^2x^2)\,. \tag{A.21}$$

The relation can also be proven by applying L'Hôpital's rule to relation (A.11) as shown in [347].

Evaluation for $m>0$

Proceeding similarly as for the case of $m=0$, the Taylor expansion can be written as

$$\begin{aligned}&\partial^m\Phi\left(-\frac{1}{\varepsilon}+\varepsilon x\right)\\&\quad=\partial^m\Phi\left(-\frac{1}{\varepsilon}+\varepsilon x\right)\Big|_{x=0}+\sum_{n=1}^{\infty}\frac{1}{n!}(-1)^{n+m-1}\varepsilon^{n+m}\mathrm{He}_{n+m-1}\left(-\frac{1}{\varepsilon}\right)\varphi\left(-\frac{1}{\varepsilon}\right)x^n\,.\end{aligned} \tag{A.22}$$

Fixing the first two order terms of the exponential function leads to

$$\frac{\partial^m \Phi\left(-\frac{1}{\varepsilon}+\varepsilon x\right)\big|_{x=x/\sigma_m(\epsilon)}}{\partial^m \Phi\left(-\frac{1}{\varepsilon}+\varepsilon x\right)\big|_{x=0}} = 1 + x + \sum_{n=2}^{\infty} \frac{1}{n!} \frac{(-1)^{n+m-1}\varepsilon^{n+m}\mathrm{He}_{n+m-1}\left(-\frac{1}{\varepsilon}\right)\varphi\left(-\frac{1}{\varepsilon}\right)}{(-1)^{m-1}\varepsilon^m \mathrm{He}_{m-1}\left(-\frac{1}{\varepsilon}\right)\varphi\left(-\frac{1}{\varepsilon}\right)\sigma_m(\varepsilon)^n} x^n$$

$$= 1 + x + \sum_{n=2}^{\infty} \frac{1}{n!} \frac{(-1)^n \varepsilon^n \mathrm{He}_{n+m-1}\left(-\frac{1}{\varepsilon}\right)}{\mathrm{He}_{m-1}\left(-\frac{1}{\varepsilon}\right)\sigma_m(\varepsilon)^n} x^n \,, \tag{A.23}$$

where we have evaluated the expression $\partial^m \Phi\left(-\frac{1}{\varepsilon}+\varepsilon x\right)\Big|_{x=0}$ on the right-hand side and where the scaling factor $\sigma_m(\varepsilon)$ is given by

$$\sigma_m(\varepsilon) = -\frac{\varepsilon\, \mathrm{He}_m\left(-\frac{1}{\varepsilon}\right)}{\mathrm{He}_{m-1}\left(-\frac{1}{\varepsilon}\right)} \,. \tag{A.24}$$

Again, the asymptotic behavior of the fractional factor has to be computed for $\epsilon \to 0$. From the highest order term of the probabilists' Hermite polynomial, it can be deduced that

$$\lim_{\varepsilon\to 0} \frac{\varepsilon^n\, \mathrm{He}_{n+m-1}\left(-\frac{1}{\varepsilon}\right)}{\mathrm{He}_{m-1}\left(-\frac{1}{\varepsilon}\right)} = (-1)^n + \mathcal{O}(\epsilon^n) \,. \tag{A.25}$$

For $n = 1$, this corresponds to (-1) times the scaling factor $\sigma(\varepsilon)$, therefore,

$$\lim_{\varepsilon\to 0} \sigma(\varepsilon) = 1 + \mathcal{O}(\varepsilon^n) \,. \tag{A.26}$$

It can be derived with the same arguments as for the case of $m = 0$, that the fractional factor converges to 1 and that the limit with its order of accuracy is given by

$$n_{\varepsilon^2,m} = \frac{\partial^m \Phi\left(-\frac{1}{\varepsilon}+\varepsilon x\right)\big|_{x=x/\sigma_m(\epsilon)}}{\partial^m \Phi\left(-\frac{1}{\varepsilon}+\varepsilon x\right)\big|_{x=0}} = \exp(x) + \mathcal{O}(\varepsilon^2 x^2) \,. \tag{A.27}$$

A.3 Statistical properties of the sign-dependent Ornstein-Uhlenbeck process

The transition probability of the sign-dependent Ornstein-Uhlenbeck process, defined in Eq. (4.22) in Chapter 4, can be derived in the same manner as for Langevin dynamics in App. A.1. We start by considering the process with discrete time steps,

$$u_i' = u_i + \varepsilon \left[\frac{\mathrm{sign}(u_i)}{\sqrt{\epsilon}} + \frac{\sqrt{\epsilon}}{2} \left(\sum_{\mathrm{syn}j} W_{ij} z_j + b_i \right) - u_i \right] + \sqrt{2\varepsilon}\tilde{\eta} \,, \tag{A.28}$$

with $\theta = 1$ and $\sigma = \sqrt{2}$. The t-dependency is hidden for simplicity by introducing $u_i := u_{i,\mathrm{eff}}(t)$ and $u_i' := u_{i,\mathrm{eff}}(t+\varepsilon)$ as well as $z_j := z_j(t)$. We define the term in the squared brackets as

$$\frac{\partial K(u_i)}{\partial u_i} := \frac{\mathrm{sign}(u_i)}{\sqrt{\epsilon}} + \frac{\sqrt{\epsilon}}{2} \left(\sum_{\mathrm{syn}j} W_{ij} z_j + b_i \right) - u_i \,. \tag{A.29}$$

This definition allows a one-to-one comparison between the discrete update equation (A.28) and the discretized Langevine equation (A.1). The transition probability for the sign-dependent Ornstein-Uhlenbeck process can be expressed based on this comparison by

$$W(u_i \to u_i') = \frac{1}{\sqrt{2\varepsilon}}\varphi\left(\frac{u_i' - u_i}{\sqrt{2\varepsilon}}\right)\exp\left[\frac{K(u_i') - K(u_i)}{2} + \mathcal{O}(\varepsilon)\right], \tag{A.30}$$

with

$$K(u_i) = \frac{|u_i|}{\sqrt{\epsilon}} + \frac{\sqrt{\epsilon}}{2}\left(\sum_{\text{syn}j} W_{ij}z_j + b_i\right)u_i - \frac{u_i^2}{2}. \tag{A.31}$$

The equilibrium distribution can be inferred from the detailed-balance equation and is given by

$$P_{\text{eq}}(u_{i,\text{eff}}) \propto \exp\left[K(u_i)\right]. \tag{A.32}$$

The distribution facilitates a derivation of the probability distributions of $P_{\text{OU}^2}(z_i = 1)$ and $P_{\text{OU}^2}(z_i = -1)$, where we use that $z_i = \Theta\left[u_{i,\text{eff}}\right]$. One obtains for the distributions

$$P_{\text{OU}^2}(z_i = 1) = \int_0^\infty P_{\text{eq}}(u_{i,\text{eff}})\mathrm{d}u_{i,\text{eff}} \propto \exp\left[\frac{(\epsilon m_i - 2)^2}{8\epsilon}\right]\Phi\left[\frac{1}{\sqrt{\epsilon}} - \frac{\sqrt{\epsilon}}{2}m_i\right] \tag{A.33}$$

and

$$P_{\text{OU}^2}(z_i = -1) = \int_{-\infty}^0 P_{\text{eq}}(u_{i,\text{eff}})\mathrm{d}u_{i,\text{eff}} \propto \exp\left[\frac{(\epsilon m_i + 2)^2}{8\epsilon}\right]\Phi\left[\frac{1}{\sqrt{\epsilon}} + \frac{\sqrt{\epsilon}}{2}m_i\right], \tag{A.34}$$

where we defined the total input of a neuron i as

$$m_i := -\sum_{\text{syn}j} W_{ij}z_j - b_i. \tag{A.35}$$

Since we consider only two states for z_i, $P_{\text{OU}^2}(z_i = 1)$ can be normalized based on the two previous definitions, leading to

$$P_{\text{OU}^2}(z_i = 1) = \frac{1}{1 + \exp\left[\alpha_\epsilon(m_i) \times m_i\right]}. \tag{A.36}$$

The correction factor $\alpha(m_i)$ is given by

$$\alpha(m_i) := 1 + \frac{1}{m_i}\log\left[\frac{\Phi\left(\frac{1}{\sqrt{\epsilon}} + \frac{\sqrt{\epsilon}}{2}m_i\right)}{\Phi\left(\frac{1}{\sqrt{\epsilon}} - \frac{\sqrt{\epsilon}}{2}m_i\right)}\right]. \tag{A.37}$$

It converges in the limit of $\epsilon \to 0$ to one,

$$\lim_{\epsilon\to 0}\alpha_\epsilon(m_i) = 1. \tag{A.38}$$

A.4 Derivation of the dynamics of the Langevin machine

We show that the Langevin equation for discrete systems (3.1), defined in Sec. 3.1, can be rewritten for the simulation of a Boltzmann machine, resulting in the simplified update rule (4.19), introduced in Chapter 4.

We start by considering the energy of the Boltzmann machine

$$E(\vec{z}) = -\sum_{i<j} W_{ij} z_i z_j - \sum_i b_i z_i \,, \tag{A.39}$$

with a total input $m_i := -\sum_j W_{ij} z_j - b_i$ for a neuron i. The possible resulting energy differences for a change of the state of the neuron are given by

$$\begin{aligned} \Delta E(z_i' = 1, z_i = 0) &= m_i \,, \\ \Delta E(z_i' = 0, z_i = 1) &= -m_i \,. \end{aligned} \tag{A.40}$$

The proposed state for a two-state system always corresponds to the other state. This results in the following two update rules for a transition from $z_i = 0 \to 1$ and $z_i = 1 \to 0$:

$$\begin{aligned} z_i' &= \Theta\left[-1 - \frac{\epsilon}{2\lambda_\epsilon} m_i + \sqrt{\epsilon}\tilde{\eta}_i{}^T\right] , \quad \text{for} \quad z_i = 0 \to 1 \,, \\ z_i' &= \Theta\left[1 - \frac{\epsilon}{2\lambda_\epsilon} m_i + \sqrt{\epsilon}\tilde{\eta}_i{}^T\right] , \quad \text{for} \quad z_i = 1 \to 0 \,. \end{aligned} \tag{A.41}$$

Taking the current state into account, the relations can be merged into a common update rule,

$$z_i' = \Theta\left[(2z_i - 1) - \frac{\epsilon}{2\lambda_\epsilon} m_i + \sqrt{\epsilon}\tilde{\eta}_i{}^T\right] . \tag{A.42}$$

After a division by $\sqrt{\epsilon}$ and a rearranging of the summands, one arrives at the following update rule for the Langevin machine,

$$z_i' = \Theta\left[\frac{2}{\sqrt{\epsilon}} z_i + \sum_j \frac{\sqrt{\epsilon}}{2\lambda_\epsilon} W_{ij} z_j + \frac{\sqrt{\epsilon}}{2\lambda_\epsilon} b_i - \frac{1}{\sqrt{\epsilon}} + \tilde{\eta}_i{}^T\right] . \tag{A.43}$$

By the identifications

$$W_{ii}' = \frac{2}{\sqrt{\epsilon}} \,, \quad W_{ij}' = \frac{\sqrt{\epsilon}}{2\lambda_\epsilon} W_{ij} \,, \quad b_i' = \left(\frac{\sqrt{\epsilon}}{2\lambda_\epsilon} b_i - \frac{1}{\sqrt{\epsilon}}\right) , \tag{A.44}$$

the update rule can be written as, cf. Eq. (4.19),

$$z_i' = \Theta\left[W_{ii}' z_i + \sum_j W_{ij}' z_j + b_i' + \tilde{\eta}^T\right] . \tag{A.45}$$

APPENDIX B

Detailed-balance equation in multiple variables for different algorithms

This appendix is based on Ref. [5].

B.1 Hamiltonian Monte Carlo

The Hamiltonian Monte Carlo (HMC) algorithm assigns a momentum π to each state x of a considered system with probability distribution $\rho(x) \propto \exp(-S(x))$ [293]. Therefore, it can be considered as a Monte Carlo algorithm in auxiliary dimensions, see Sec. 7.2. The state x and the momentum π can be identified with the variables v and w. This appendix briefly demonstrates how the algorithm samples, in equilibrium, from the desired distribution $\rho(x)$. A thorough introduction to the HMC algorithm can be found, for example, in Refs. [348, 349].

A common implementation of the HMC algorithm consists of the following steps:

1. Sample a momentum π from a Gaussian distribution according to $q(\pi) = \varphi(\pi)$, with

$$\varphi(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) . \tag{B.1}$$

2. Perform an integration of Hamilton's equations

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \frac{\partial H}{\partial \pi} , \quad \frac{\mathrm{d}\pi}{\mathrm{d}t} = -\frac{\partial H}{\partial x} \tag{B.2}$$

   for a finite amount of time and negate the proposed momentum. We refer to the proposed state and momentum as

$$x' = R\,\Phi_x(x,\pi) , \quad \pi' = R\,\Phi_p(x,\pi) , \tag{B.3}$$

   where Φ represents the outcome of the integration and R negates the resulting proposed momentum.

3. Accept or reject the proposed state with probability

$$\min\left[1, \exp\left(-(H(x',\pi') - H(x,\pi))\right)\right] . \tag{B.4}$$

The Hamiltonian is defined by

$$H(x,\pi) = S(x) + \frac{\pi^2}{2m}\,, \tag{B.5}$$

where $S(x)$ represents the action or energy function one wants to sample from.

The algorithm implements the transition probability:

$$\begin{aligned} W(x,\pi \to x',\pi') \propto \varphi(\pi)\,\delta\left(x' - R\,\Phi_x(x,\pi)\right)\,\delta\left(\pi' - R\,\Phi_\pi(x,\pi)\right) \\ \times\, \min\left[1, \exp\left(-(H(x',\pi') - H(x,\pi))\right)\right]\,. \end{aligned} \tag{B.6}$$

The resulting equilibrium distribution in the higher-dimensional state space is given by

$$p(x,\pi) \propto \exp\left(-H(x,\pi)\right)\,. \tag{B.7}$$

The Metropolis accept/reject step takes into account numerical errors in the integration scheme for x and π. Otherwise, the proposal state can be always accepted since it holds $H(x',\pi') = H(x,\pi)$, as a result of Hamilton's equations.

The transition probability is designed to satisfy a detailed-balance equation in the higher-dimensional state space, cf. Eq. (7.12) in Chapter 7:

$$p(x,\pi)W(x,\pi \to x',\pi') = p(x',\pi')W(x',\pi' \to x,\pi)\,. \tag{B.8}$$

Detailed balance is ensured since the evolution of x and π is time-reversible and volume-preserving.

Due to the statistical independence of x and π, the target distribution $\rho(x)$ can be obtained by a marginalization of the joint distribution $p(x,\pi)$:

$$\rho(x) = \int \mathrm{d}\pi\, p(x,\pi)\,. \tag{B.9}$$

Hence, the sampled momenta π can be ignored in a numerical computation of observables $\mathcal{O}(x)$.

B.2 Restricted Boltzmann machine

Restricted Boltzmann machines (RBMs) are stochastic and generative neural networks [243, 294, 350] typically used to parametrize probability distributions over a set of input samples. New samples can be drawn in equilibrium by Gibbs sampling. In contrast to the Hamiltonian Monte Carlo algorithm, a detailed-balance equation is only fulfilled in subsampling steps, but not for the entire update. The detailed-balance equation (7.12) is not satisfied. Instead, the transition probabilities satisfy the more general constraint (7.11) to sample correctly from the desired equilibrium distribution, as we will show in the following. Before that, we provide a short reminder of the algorithm.

The restricted Boltzmann machine consists of visible and hidden neurons, denoted as v and w. They form a visible and a hidden layer. Each neuron can be either active or inactive. This is implemented by a binary state space, $v_i \in \{0,1\}$ and $w_j \in \{0,1\}$.

An energy function can be defined in dependence of a given configuration $(v,\, w)$,

$$E(v,w) = -\sum_i b_i v_i - \sum_j c_j w_j - \sum_{i,j} W_{ij} v_i w_j \,. \tag{B.10}$$

The neural network parameters are given by the set $\{\vec{b}, \vec{c}, \mathbf{W}\}$. In contrast to the Boltzmann machine, the weight matrix is restricted to connections between single neurons of the visible and of the hidden layer. The resulting probability distributions for the RBM is defined as

$$p(v,w) = \frac{1}{Z} \exp\left(-E(v,w)\right) \,, \tag{B.11}$$

with Z being the partition sum,

$$Z = \sum_{v,w} \exp\left(-E(v,w)\right) \,. \tag{B.12}$$

In general, one aims to learn a probability distribution that is defined over the visible neurons v. It is given by the marginal distribution

$$\rho(v) = \frac{1}{Z} \sum_w \exp\left(-E(v,w)\right) \,, \tag{B.13}$$

where the sum runs over all possible configurations of w. The hidden neurons correspond to latent variables that increase the expressibility of the represented distribution.

One possible approach to train the restricted Boltzmann machine is by contrastive divergence. The network parameters are adapted by a step-wise training procedure to best approximate the distribution of a training set over samples v. For more details, see Refs. [243, 266], for example.

The trained restricted Boltzmann machine can be used as a generative model to draw samples from a parametrized distribution of the training set. This is realised by an update of the visible and hidden neurons based on the conditional distributions

$$p(v'|w) = \frac{\exp\left(-E(v',w)\right)}{\sum_v \exp\left(-E(v,w)\right)} \tag{B.14}$$

and

$$p(w'|v) = \frac{\exp\left(-E(v,w')\right)}{\sum_w \exp\left(-E(v,w)\right)} \,. \tag{B.15}$$

The sums in the denominator run again over all possible configurations of v or w, respectively.

The transition probabilities define a Markov process for the restricted Boltzmann machine. With the above definitions we are now able to analyse how the RBM samples in equilibrium from the desired distribution $p(v,w)$, namely, by satisfying the constraint (7.11)

$$p(v',w') \stackrel{!}{=} \sum_{v,w} p(v,w)\, W(v,w \to v',w') \,. \tag{B.16}$$

The time-dependence has been dropped since we assume the distribution to be in equilibrium.

A full update step is implemented by a consecutive sampling from the conditional distributions in Eqs. (B.14) and (B.15), resulting in the transition probability

$$W(v,w \to v',w') = p(v'|w')\, p(w'|v) \,. \tag{B.17}$$

After inserting this into Eq. (B.16), one obtains for the right-hand side

$$\sum_{v,w} p(v,w)\, p(v'|w')\, p(w'|v) = \sum_{v,w} p(v,w')\, p(v'|w')\, p(w|v)\,, \tag{B.18}$$

where we used in the second line that the transition probability $p(w'|v)$ satisfies, for a fixed v, the detailed-balance equation

$$p(w'|v)\, p(v,w) = p(w',w,v) = p(w|v)\, p(v,w')\,. \tag{B.19}$$

We can now perform the sum over w and are left with

$$p(v',w') \stackrel{!}{=} \sum_{v} p(v,w')\, p(v'|w')\,. \tag{B.20}$$

After replacing w' by w and factoring out the transition probability, it is easy to see that the constraint matches with the transition probability of the visible variable in Eq. (B.14).

APPENDIX C

Complex Langevin-type sampling by compensation algorithms

This appendix is based on Ref. [5].

C.1 Complex Langevin dynamics

For completeness, the discretized update equations of complex Langevin dynamics, see Eq. (7.24) in Chapter 7, are derived explicitly from the transition probabilities for complex actions, defined in Sec. 8.2:

$$T(\phi'_x|\phi_x,\phi_y) \propto \frac{1}{\sqrt{2\epsilon}}\varphi\left(\frac{\phi'_x-\phi_x}{\sqrt{2\epsilon}}\right)\exp\left(-\frac{\Delta S_{\mathrm{Re}}(\phi',\phi)}{2}\right), \tag{C.1}$$

and

$$\phi'_y = \phi_y - \epsilon\frac{\Delta S_{\mathrm{Im}}(\phi',\phi)}{\phi'_x-\phi_x}. \tag{C.2}$$

The action difference $\Delta S(\phi',\phi)$ can be expanded around ϕ_x since the step sizes in the real direction are constraint to be infinitesimally small:

$$\begin{aligned}\Delta S(\phi',\phi) &= S(\phi_x+\delta\phi_x+i\phi_y) - S(\phi_x+i\phi_y)\\ &\simeq \delta\phi_x\frac{\delta S(\phi_x+i\phi_y)}{\delta\phi_x} = \delta\phi_x\left[\frac{\delta S(\phi)}{\delta\phi}\bigg|_{\phi_x+i\phi_y}\right],\end{aligned} \tag{C.3}$$

where

$$\begin{aligned}S(\phi_x+i\phi_y) &= S_{\mathrm{Re}}(\phi_x+i\phi_y) + iS_{\mathrm{Im}}(\phi_x+i\phi_y)\\ &\equiv S_{\mathrm{Re}} + iS_{\mathrm{Im}}.\end{aligned} \tag{C.4}$$

The expansion simplifies the update rule (C.2) of the imaginary part ϕ_y, resulting in

$$\phi'_y = \phi_y - \epsilon\frac{\delta S_{\mathrm{Im}}}{\delta\phi_x}, \tag{C.5}$$

the discrete update dynamics of the imaginary field ϕ_y in complex Langevin dynamics.

The transition probability (C.1) turns, with this expansion, into

$$T(\phi'_x|\phi_x,\phi_y) = \frac{1}{\sqrt{2\epsilon}}\varphi\left(\frac{\phi'_x-\phi_x}{\sqrt{2\epsilon}} + \sqrt{\frac{\epsilon}{2}}\frac{\delta S_{\mathrm{Re}}}{\delta\phi_x}\right). \tag{C.6}$$

The absorption of the action term into the Gaussian distribution can be shown by expanding the argument of the exponential function in Eq. (C.1) with a first order term in ϵ. The first Gaussian distribution and the exponential term are contracted by completing the square in the exponent. The computation is equivalent to the one in App. A.1.

An explicit update rule for ϕ_x can be derived by a transformation of the transition probability, by demanding

$$\int_{-\infty}^{\phi'_x} \mathrm{d}\tilde{\phi}_x\, T(\tilde{\phi}_x|\phi_x,\phi_y) \stackrel{!}{=} \int_{-\infty}^{\eta} \mathrm{d}\tilde{\eta}\,\varphi(\tilde{\eta})\,. \tag{C.7}$$

Evaluating both integrals and solving for ϕ'_x results in the discrete update rule:

$$\phi'_x = \phi_x - \epsilon\frac{\delta S_{\mathrm{Re}}}{\delta\phi_x} + \sqrt{2\epsilon}\eta\,. \tag{C.8}$$

By using the relations in Eq. (8.9) in Chaper 8, the derived update rules coincide with the ones of complex Langevin dynamics, Eq. (7.24).

C.2 Second order complex Langevin

It is possible to formulate a discrete second order complex Langevin equation. The derivation follows the same line of argumentation as in the previous section, and the name refers to the second order terms in the expansion of the action difference for infinitesimally small step sizes. We keep this term in the expansion in Eq. (C.3),

$$\begin{aligned}\Delta S(\phi',\phi) &= S(\phi_x+\delta\phi_x+i\phi_y) - S(\phi_x+i\phi_y)\\ &\simeq \delta\phi_x\frac{\delta S(\phi_x+i\phi_y)}{\delta\phi_x} + \frac{\delta\phi_x^2}{2}\frac{\delta^2 S(\phi_x+i\phi_y)}{\delta\phi_x^2}\,.\end{aligned} \tag{C.9}$$

The second order expansion of the action difference in the imaginary part can be inserted into Eq. (C.2), the update rule of the imaginary field ϕ_y. This results in

$$\phi'_y = \phi_y - \epsilon\frac{\delta S_{\mathrm{Im}}}{\delta\phi_x} - \frac{\epsilon}{2}\left(\phi'_x-\phi_x\right)\frac{\delta^2 S_{\mathrm{Im}}}{\delta\phi_x^2}\,, \tag{C.10}$$

where we used $\phi'_x-\phi_x = \delta\phi_x$. The update rule again compensates the imaginary contributions in the transition probability.

An update rule for the real part of the field can be derived in the same manner as for the Langevin equation. We complete the exponent of the product of the first Gaussian distribution and of the exponential function in Eq. (C.1) by

$$\frac{\epsilon}{2}\left[\frac{\delta S_{\mathrm{Re}}}{\delta\phi_x} + \frac{\phi'_x-\phi_x}{2}\frac{\delta^2 S_{\mathrm{Re}}}{\delta\phi_x^2}\right]. \tag{C.11}$$

The argument of the Gaussian distribution in the transition probability (C.6) now reads

$$\frac{\phi'_x - \phi_x}{\sqrt{2\epsilon}} + \sqrt{\frac{\epsilon}{2}}\left[\frac{\delta S_{\mathrm{Re}}}{\delta \phi_x} + \frac{\phi'_x - \phi_x}{2}\frac{\delta^2 S_{\mathrm{Re}}}{\delta \phi_x^2}\right] = \frac{\phi'_x - \phi_x}{\sqrt{2\epsilon}}\left[1 + \frac{\epsilon}{2}\frac{\delta^2 S_{\mathrm{Re}}}{\delta \phi_x^2}\right] + \sqrt{\frac{\epsilon}{2}}\frac{\delta S_{\mathrm{Re}}}{\delta \phi_x}\,. \tag{C.12}$$

Because of the additional second order term, the normalization factor of the transition probability needs to be adjusted by the factor

$$1 + \frac{\epsilon}{2}\frac{\delta^2 S_{\mathrm{Re}}}{\delta \phi_x^2}\,. \tag{C.13}$$

An explicit update rule can be derived again by performing a transformation of the probability density, cf. Eq. (C.7) for more details. We finally arrive at

$$\phi'_x = \phi_x - \left(\epsilon\frac{\delta S_{\mathrm{Re}}}{\delta \phi_x} + \sqrt{2\epsilon}\eta\right) \Big/ \left(1 + \frac{\epsilon}{2}\frac{\delta^2 S_{\mathrm{Re}}}{\delta \phi_x^2}\right)\,. \tag{C.14}$$

The update rule of the imaginary part now depends on the outcome of the real part. Accordingly, one has to update at first ϕ_x and then ϕ_y.

C.3 Complex hat function algorithm

We derive a Langevin sampling by compensation algorithm for a different representation of the delta-distribution, namely, the triangular hat function. We consider again a complex action $S(\phi)$, as given, for example, in Eq. (2.1) for the polynomial model. The derivation is in line with the systematic derivation in Sec. 8.2.

The hat function is given by

$$\eta_\epsilon(x) = \frac{1}{\epsilon}\begin{cases} 1 - \frac{x}{\epsilon} & \text{for } 0 \le x < \epsilon\,, \\ 1 + \frac{x}{\epsilon} & \text{for } -\epsilon < x < 0\,, \\ 0 & \text{otherwise.} \end{cases} \tag{C.15}$$

We rewrite this, for simplicity, as

$$\eta_\epsilon(x) = \frac{1}{\epsilon}\left[1 - s\frac{x}{\epsilon}\right] \quad \text{for} \quad -\epsilon < x < \epsilon\,, \tag{C.16}$$

where $s := \mathrm{sign}\,(x)$. Similar to the Gaussian distribution, the hat function converges, in the limit of $\epsilon \to 0$, to the delta-distribution.

The corresponding transition probability is

$$W(\phi \to \phi') = \frac{1}{N}\left[1 - s\frac{\phi' - \phi}{\epsilon}\right] \times \exp\left(-\frac{\Delta S}{2}\right)\,, \tag{C.17}$$

with N being a normalization factor.

With the same substitution as for complex Langevin dynamics,

$$\phi \to \phi_x + i\phi_y\,, \tag{C.18}$$

we identify the imaginary part ϕ_y as hidden dimension.

After replacing ϕ by (C.18) in the transition probability (C.17), we identify

$$q_s = \frac{1}{\epsilon}\left[1 - \operatorname{sign}_{\phi'_x-\phi_x}\frac{\phi'_x-\phi_x}{\epsilon}\right],$$
$$\bar{q}_s = \frac{i}{\epsilon}\left[\operatorname{sign}_{\phi'_x-\phi_x}\frac{\phi'_y-\phi_y}{\epsilon}\right], \tag{C.19}$$

which are summed according to Eq. (8.19) and continue by decomposing the acceptance probability,

$$\begin{aligned} A(\phi',\phi) &\propto \exp\left(-\frac{\Delta S_{\mathrm{Re}}(\phi',\phi)}{2}\right)\times\left[\cos\left(-\frac{\Delta S_{\mathrm{Im}}(\phi',\phi)}{2}\right)+i\sin\left(-\frac{\Delta S_{\mathrm{Im}}(\phi',\phi)}{2}\right)\right] \\ &= \exp\left(-\frac{\Delta S_{\mathrm{Re}}(\phi',\phi)}{2}\right)\times\left[A_s+\bar{A}_s\right], \end{aligned} \tag{C.20}$$

with the action given by $S(\phi_x,\phi_y) = S_{\mathrm{Re}}(\phi_x,\phi_y) + iS_{\mathrm{Re}}(\phi_x,\phi_y)$. This allows distinguishing contributions that have no impact on detailed balance and contributions that need to vanish. The term A_s is symmetric with respect to an exchange of ϕ'_x and ϕ_x whereas $\bar{A}_s$ is antisymmetric. Expanding the product of the proposal distribution and the acceptance probability gives a proposal distribution which is symmetric under an exchange of ϕ'_x and ϕ_x, and which will drop out in the detailed-balance equation. Based on this, the transition probability for the real field ϕ_x is defined up to a normalisation factor as

$$T(\phi'_x|\phi_x,\phi_y) \propto \exp\left(-\frac{\Delta S_{\mathrm{Re}}(\phi',\phi)}{2}\right)\times\left[q_sA_s+\bar{q}_s\bar{A}_s\right]. \tag{C.21}$$

The remaining terms must vanish,

$$h(\phi'_y|\phi'_x,\phi_x,\phi_y) = \exp\left(-\frac{\Delta S_{\mathrm{Re}}(\phi',\phi)}{2}\right)\times\left[q_s\bar{A}_s+\bar{q}_sA_s\right] \stackrel{!}{=} 0, \tag{C.22}$$

for the not yet assigned parameter ϕ'_y. This defines the update rule for ϕ_y:

$$\phi'_y = \phi_y + \left[\frac{\epsilon}{s} - (\phi'_x-\phi_x)\right]\tan\left(-\frac{\Delta S_{\mathrm{Im}}(\phi',\phi)}{2}\right), \tag{C.23}$$

and thus $g(\phi'_y|\phi'_x,\phi_x,\phi_y)$. Because of the distinction of symmetric and antisymmetric parts, the transition probability g posses Langevin symmetry

$$g(\phi'_y|\phi'_x,\phi_x,\phi_y) = g(\phi'_y|\phi_x,\phi'_x,\phi_y). \tag{C.24}$$

The update rule of the imaginary part can be used to further simplify the transition probability. We can solve the update rule (C.23) for $\bar{q}_s$ and insert the result into the transition probability to obtain

$$T(\phi'_x|\phi_x,\phi_y) \propto \exp\left(-\frac{\Delta S_{\mathrm{Re}}(\phi',\phi)}{2}\right)\times q_s\left[A_s-\frac{\bar{A}_s^2}{A_s}\right]. \tag{C.25}$$

The resulting transition probability T leads to the same violation of the adapted detailed-balance equation (7.30) as for complex Langevin dynamics. The algorithm samples from the correct distribution only for $\epsilon \to 0$.

C.4 Uniform complex Langevin

Utilizing the results of the previous appendix, we define a sampling algorithm that uses a centred uniform distribution to propose states ϕ'_x.

This leads to the following ansatz for the transition probability $W(\phi \to \phi')$:

$$W(\phi \to \phi') \int_{-l}^{l} \frac{\mathrm{d}r}{2l}\, \delta\left(\phi' - (\phi + r)\right) \times \exp\left(-\frac{\Delta S(\phi', \phi)}{2}\right) , \tag{C.26}$$

where, in practice, r is sampled from a uniform distribution in the interval $[-l, l]$. Replacing the delta-distribution by the triangular hat function (C.15), gives

$$W(\phi \to \phi') \propto \int_{-l}^{l} \frac{\mathrm{d}r}{2l} \frac{1}{\epsilon} \left[1 - \tilde{s}\frac{\phi' - (\phi + r)}{\epsilon}\right] \times \exp\left(-\frac{\Delta S(\phi', \phi)}{2}\right) , \tag{C.27}$$

with

$$\tilde{s} = \mathrm{sign}\left(\phi' - (\phi + r)\right) . \tag{C.28}$$

Performing the same steps as for the complex hat function algorithm, this yields the update rule of the imaginary part,

$$\phi'_y = \phi_y + \left[\frac{\epsilon}{\tilde{s}} - \left(\phi'_x - (\phi_x + r)\right)\right] \tan\left(-\frac{\Delta S_{\mathrm{Im}}(\phi', \phi)}{2}\right) . \tag{C.29}$$

To restore the original uniform distribution, we take the limit $\epsilon \to 0$. The update rule simplifies to

$$\phi'_y = \phi_y + \left(\phi'_x - (\phi_x + r)\right) \tan\left(-\frac{\Delta S_{\mathrm{Im}}(\phi', \phi)}{2}\right) . \tag{C.30}$$

Following the same approach for the transition probability $T(\phi'_x|\phi_x, \phi_y)$, one arrives at

$$\begin{aligned} T(\phi'_x|\phi_x, \phi_y) \propto \int_{-l}^{l} \frac{\mathrm{d}r}{2l}\, \delta\left(\phi'_x - (\phi_x + r)\right) \\ \times \exp\left(-\frac{\Delta S_{\mathrm{Re}}(\phi', \phi)}{2}\right) \cos^{-1}\left(-\frac{\Delta S_{\mathrm{Im}}(\phi', \phi)}{2}\right) . \end{aligned} \tag{C.31}$$

According to the proposal distribution, the second term of the update rule (C.30) of the imaginary part vanishes, resulting in $\phi'_y = \phi_y$. To prevent this, we draw, in each update, two proposal states, ϕ'_x and $\tilde{\phi}'_x$, and adapt the update rule of the imaginary part:

$$\phi'_y = \phi_y + \left(\tilde{\phi}'_x - (\phi_x + r)\right) \tan\left(-\frac{\Delta S_{\mathrm{Im}}(\phi', \phi)}{2}\right) . \tag{C.32}$$

Similar to the complex hat function algorithm, the derived algorithm satisfies the constraints of the substitution algorithm only in the limit of infinitesimally small step sizes into the ϕ'_x direction. Based on the proposal distribution this can be implemented by considering the limit $l \to 0$.

C.5 Absorbing the imaginary contribution

We start by considering

$$\begin{aligned} &T(\phi_x|\phi'_x,\phi_y)\exp\left(-i\Delta S_{\mathrm{Im}}(\phi,\phi')\right)\\ &\quad\propto\varphi\left(\frac{\phi_x-\phi'_x}{\sqrt{2\epsilon}}\right)\exp\left(-\frac{\Delta S_{\mathrm{Re}}(\phi,\phi')}{2}\right)\exp\left(-i\Delta S_{\mathrm{Im}}(\phi,\phi')\right)\,, \end{aligned} \tag{C.33}$$

with the goal to express the transition probability in terms of a Gaussian distribution.

Expanding ΔS_{Re} and ΔS_{Im} around ϕ'_x, one obtains

$$\begin{aligned} &T(\phi_x|\phi'_x,\phi_y)\exp\left(-i\Delta S_{\mathrm{Im}}(\phi,\phi')\right)\\ &\quad\propto\exp\left(-\frac{1}{2}\left(\frac{\phi_x-\phi'_x}{\sqrt{2\epsilon}}\right)^2-\frac{\phi_x-\phi'_x}{2}\left(\frac{\delta S_{\mathrm{Re}}(\phi'_x+i\phi_y)}{\partial\phi'_x}+2i\frac{\delta S_{\mathrm{Im}}(\phi'_x+i\phi_y)}{\partial\phi'_x}\right)\right)\,. \end{aligned} \tag{C.34}$$

Next, we complete the square in the exponent,

$$\begin{aligned} &T(\phi_x|\phi'_x,\phi_y)\exp\left(-i\Delta S_{\mathrm{Im}}(\phi,\phi')\right)\\ &\quad\propto\varphi\left(\frac{\phi_x-\phi'_x}{\sqrt{2\epsilon}}+\sqrt{\frac{\epsilon}{2}}\left(\frac{\delta S_{\mathrm{Re}}(\phi'_x+i\phi_y)}{\partial\phi'_x}+2i\frac{\delta S_{\mathrm{Im}}(\phi'_x+i\phi_y)}{\partial\phi'_x}\right)\right)\,. \end{aligned} \tag{C.35}$$

We insert this expression into constraint (7.34) in Chapter 7. As a result, an integration over ϕ_x is possible since the dependence on ϕ_x in the action was eliminated by the expansion around ϕ'_x.

APPENDIX D

Entangled quantum states and learning on the spiking neuromorphic chip

This appendix is based on Ref. [4].

D.1 Representation of the Bell state

The Bell state, $|\Psi^+\rangle = 1/\sqrt{2}\,(|\uparrow\uparrow\rangle + |\downarrow\downarrow\rangle)$, is described by the density matrix

$$\rho_{\mathrm{B}} = \frac{1}{2}\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

in the standard basis. To encode this state in a spiking neural network, we map it to a POVM probability distribution.

While several choices of POVM representations are possible, we here focus on the tetrahedral representation, where each measurement projects a single qubit onto one corner of a tetrahedron in the Bloch sphere [87]. The POVM elements M_{a_i} for each qubit i can hence be expressed in the form $M_{a_i} = \left(\mathbb{1} + \vec{s}_{a_i}\boldsymbol{\sigma}\right)/4$, with Pauli operators $\boldsymbol{\sigma} = (\sigma^x, \sigma^y, \sigma^z)$ and $s_{a_i=0} = (0, 0, 1)$, $s_{a_i=1} = 1/3\left(2\sqrt{2}, 0, -1\right)$, $s_{a_i=2} = 1/3\left(-\sqrt{2}, \sqrt{6}, -1\right)$, $s_{a_i=3} = 1/3\left(-\sqrt{2}, -\sqrt{6}, -1\right)$. The POVM elements thus take the form

$$\begin{aligned} M_{a_i=0} &= \frac{1}{2}\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad M_{a_i=1} = \frac{1}{6}\begin{bmatrix} 1 & \sqrt{2} \\ \sqrt{2} & 2 \end{bmatrix}, \\ M_{a_i=2} &= \frac{1}{12}\begin{bmatrix} 2 & -\sqrt{2} - \mathrm{i}\sqrt{6} \\ -\sqrt{2} + \mathrm{i}\sqrt{6} & 4 \end{bmatrix}, \\ M_{a_i=3} &= \frac{1}{12}\begin{bmatrix} 2 & -\sqrt{2} + \mathrm{i}\sqrt{6} \\ -\sqrt{2} - \mathrm{i}\sqrt{6} & 4 \end{bmatrix}. \end{aligned} \tag{D.1}$$

With this, the POVM probability distribution of the Bell state, $P_{\mathrm{B}}(a_1, a_2) = \mathrm{Tr}\left[\rho_{\mathrm{B}} M_{a_1} \otimes M_{a_2}\right]$, evaluates to

$$P_{\mathrm{B}} = \frac{1}{8}\begin{bmatrix} 1 & 1/3 & 1/3 & 1/3 \\ 1/3 & 1 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 & 1 \\ 1/3 & 1/3 & 1 & 1/3 \end{bmatrix}, \tag{D.2}$$

where columns correspond to the index a_1 and rows to the index a_2.

To reconstruct the density matrix from this probability distribution, the inverse of the full-system overlap matrix T is needed, which can be constructed as the product of the single-qubit overlap matrices, $T = T_1 \otimes T_2$. Each single-qubit overlap matrix consists of the elements $T_{a_i,a_i'} = \mathrm{Tr}\left[M_{a_i} M_{a_i'}\right]$. For the tetrahedral POVM the inverse T_i^{-1} of the single-qubit overlap matrix takes the form

$$T_i^{-1} = \begin{bmatrix} 5 & -1 & -1 & -1 \\ -1 & 5 & -1 & -1 \\ -1 & -1 & 5 & -1 \\ -1 & -1 & -1 & 5 \end{bmatrix}. \tag{D.3}$$

The density matrix can then be reconstructed linearly as $\rho = \sum_{\{a_1,a_2\}} P(a_1, a_2)\, \mathcal{Q}_{a_1,a_2}$, with operators $\mathcal{Q}_{a_1,a_2} = \sum_{\{a_1',a_2'\}} (T^{-1}_{a_1,a_1'} \otimes T^{-1}_{a_2,a_2'})(M_{a_1'} \otimes M_{a_2'})$.

Furthermore, expectation values of general operators $\mathcal{O}$ can be rewritten in terms of the probability distribution $P(a_1, a_2)$,

$$\begin{aligned} \langle \mathcal{O} \rangle &= \mathrm{Tr}\left[\rho \mathcal{O}\right] \\ &= \sum_{\{a_1,a_2\}} Q^{\mathcal{O}}_{a_1,a_2} P(a_1, a_2), \end{aligned} \tag{D.4}$$

with $Q^{\mathcal{O}}_{a_1,a_2} = \sum_{\{a_1',a_2'\}} \mathrm{Tr}\left[\mathcal{O} M_{a_1'} \otimes M_{a_2'}\right] T^{-1}_{a_1,a_1'} \otimes T^{-1}_{a_2,a_2'}$. This enables an efficient evaluation of expectation values by sampling configurations from $P(a_1, a_2)$ in the POVM representation, where the density matrix does not need to be calculated explicitly. The POVM representations of important classes of quantum states can be approximated well and in a scalable way by generative modelling approaches [87]. The computational bottleneck of these methods is the generation of samples from the model distribution, see App. D.3, and can potentially be alleviated using neuromorphic devices.

The Bell state is encoded in a sampling spiking network as follows. The visible neurons $\vec{v}$ are identified with the qubits $\vec{a}$ in the POVM representation. The network parameters are trained such that the distribution $P_{\mathrm{B}}(a_1, a_2)$ is represented by the network. To achieve this, we need to translate the variables a_1, a_2, which can take four possible values each, into binary neurons $\vec{v}$, where each neuron can take the values 0 or 1. The mapping to four binary visible neurons $v_1, \ldots, v_4$ is accomplished by defining

$$a_1 = 2v_1 + v_2, \quad a_2 = 2v_3 + v_4. \tag{D.5}$$

From this we can derive the distribution $p^*_{\mathrm{B}}(\vec{v})$ over the states of the visible neurons and have all ingredients to encode the Bell state in our spiking network.

Analogously, the probability distribution for the two-qubit Werner state with noise contribution r can be derived from its density matrix [284, 351],

$$\rho_{\mathrm{W}} = \frac{1}{4}\begin{bmatrix} 1+r & 0 & 0 & 2r \\ 0 & 1-r & 0 & 0 \\ 0 & 0 & 1-r & 0 \\ 2r & 0 & 0 & 1+r \end{bmatrix}. \tag{D.6}$$

The same is true for Greenberger-Horne-Zeilinger (GHZ) states of more than two qubits, described by the density matrices [285],

$$\rho_{\mathrm{GHZ}} = \frac{1}{2}\begin{bmatrix} 1 & 0 & \dots & 0 & 1 \\ 0 & & & & 0 \\ \vdots & & 0 & & \vdots \\ 0 & & & & 0 \\ 1 & 0 & \dots & 0 & 1 \end{bmatrix}. \tag{D.7}$$

We can then approximate the corresponding probability distributions by a spiking sampling network.

D.2 Training algorithm

Our goal is to approximate a target distribution $p^*(\vec{v})$ by the model distribution $p(\vec{v};\mathcal{W})$ encoded by the spiking neuromorphic hardware. The distance between the two distributions is quantified by the Kullback-Leibler divergence,

$$\begin{aligned} D_{\mathrm{KL}}(p^*\|p) &= \sum_{\{\vec{v}\}} p^*(\vec{v}) \ln\left[\frac{p^*(\vec{v})}{p(\vec{v};\mathcal{W})}\right] \\ &= \sum_{\{\vec{v}\}} p^*(\vec{v}) \Big(\ln[p^*(\vec{v})] - \ln[\tilde{p}(\vec{v};\mathcal{W})] + \ln[Z(\mathcal{W})]\Big). \end{aligned}$$

Here we assumed that $p(\vec{v};\mathcal{W})$ is well described by the marginal of a Boltzmann distribution and introduced the unnormalized probability distribution $\tilde{p}(\vec{v};\mathcal{W}) = \sum_{\{\vec{h}\}} \exp\left[-E\left(\vec{v},\vec{h};\mathcal{W}\right)\right]$ as the exponential of the negative network energy, as well as the partition sum $Z(\mathcal{W}) = \sum_{\{\vec{v},\vec{h}\}} \exp\left[-E\left(\vec{v},\vec{h};\mathcal{W}\right)\right]$, which allows us to replace $p(\vec{v};\mathcal{W}) = \tilde{p}(\vec{v};\mathcal{W})/Z(\mathcal{W})$ in the second line of Eq. (D.8).

The gradient of the Kullback-Leibler divergence with respect to a general connecting weight $W_{i,j}$ is given by

$$\begin{aligned}
\frac{\partial D_{\mathrm{KL}}}{\partial W_{i,j}} &= \sum_{\{\vec{v}\}} p^*\left(\vec{v}\right)\left[-\frac{1}{\tilde{p}\left(\vec{v};\mathcal{W}\right)}\frac{\partial \tilde{p}\left(\vec{v};\mathcal{W}\right)}{\partial W_{i,j}} + \frac{1}{Z\left(\mathcal{W}\right)}\frac{\partial Z\left(\mathcal{W}\right)}{\partial W_{i,j}}\right] \\
&= \sum_{\{\vec{v}\}} p^*\left(\vec{v}\right)\left[-\frac{1}{\tilde{p}\left(\vec{v};\mathcal{W}\right)}\left(\sum_{\{\vec{h}\}} v_i h_j e^{-E\left(\vec{v},\vec{h};\mathcal{W}\right)}\right) + \frac{1}{Z\left(\mathcal{W}\right)}\left(\sum_{\{\vec{v}',\vec{h}\}} v_i' h_j e^{-E\left(\vec{v}',\vec{h};\mathcal{W}\right)}\right)\right] \\
&= -\sum_{\{\vec{v},\vec{h}\}} \frac{p^*\left(\vec{v}\right)}{p\left(\vec{v};\mathcal{W}\right)} v_i h_j \frac{e^{-E\left(\vec{v},\vec{h};\mathcal{W}\right)}}{Z\left(\mathcal{W}\right)} + \sum_{\{\vec{v}',\vec{h}\}} v_i' h_j \frac{e^{-E\left(\vec{v}',\vec{h};\mathcal{W}\right)}}{Z\left(\mathcal{W}\right)} \\
&= \sum_{\{\vec{v},\vec{h}\}} \left[1 - \frac{p^*\left(\vec{v}\right)}{p\left(\vec{v};\mathcal{W}\right)}\right] v_i h_j \frac{\exp\left[-E\left(\vec{v},\vec{h};\mathcal{W}\right)\right]}{Z\left(\mathcal{W}\right)} \\
&= \left\langle \left[1 - \frac{p^*\left(\vec{v}\right)}{p\left(\vec{v};\mathcal{W}\right)}\right] v_i h_j \right\rangle_{p\left(\vec{v},\vec{h};\mathcal{W}\right)} .
\end{aligned} \tag{D.8}$$

Thus, the weight updates are calculated by drawing a sample set of network states, evaluating the probability $p\left(\vec{v};\mathcal{W}\right)$ underlying the configurations in the set, and calculating the expectation value of the product of the two connected neurons, weighted with $1 - p^*\left(\vec{v}\right)/p\left(\vec{v};\mathcal{W}\right)$. When using the spiking network on the BrainScaleS-2 system, we draw these sample states by observing the network at regular points in time spaced by 2 µs (for a refractory time of about 10 µs, see App. D.4).

The weight update in training epoch t then reads

$$W_{i,j}^{t} = W_{i,j}^{t-1} - \eta \left\langle \left[1 - \frac{p^*\left(\vec{v}\right)}{p\left(\vec{v};\mathcal{W}\right)}\right] v_i h_j \right\rangle_{p\left(\vec{v},\vec{h};\mathcal{W}\right)} , \tag{D.9}$$

with learning rate η. Analogously, updates for the biases can be derived,

$$\begin{aligned}
b_j^t &= b_j^{t-1} - \eta \left\langle \left[1 - \frac{p^*\left(\vec{v}\right)}{p\left(\vec{v};\mathcal{W}\right)}\right] h_j \right\rangle_{p\left(\vec{v},\vec{h};\mathcal{W}\right)} , \\
d_i^t &= d_i^{t-1} - \eta \left\langle \left[1 - \frac{p^*\left(\vec{v}\right)}{p\left(\vec{v};\mathcal{W}\right)}\right] v_i \right\rangle_{p\left(\vec{v},\vec{h};\mathcal{W}\right)} .
\end{aligned} \tag{D.10}$$

If connections between the visible neurons exist in the network structure, the updates for those connecting weights are analogous to Eq. (D.9), where the weighted expectation value of the product of the corresponding visible neurons is evaluated. This learning scheme is a modified version of wake-sleep learning [235].

Since this training algorithm is based on a gradient-descent ansatz, we can apply further modifications which lead to better convergence, such as a momentum approach to avoid getting stuck at a local minimum. In our simulations, we apply the Adam optimizer scheme. This scheme combines a momentum approach with an adaptive learning rate which is chosen for each network parameter

individually. The update for a general network parameter $\mathcal{W}_k$ is given, in the Adam optimizer, by

$$\begin{aligned} m_k^t &= \beta_1 m_k^{t-1} + (1-\beta_1)\frac{\partial D_{\mathrm{KL}}\left(p^*\|p\right)}{\partial \mathcal{W}_k}, \\ v_k^t &= \beta_2 v_k^{t-1} + (1-\beta_2)\left[\frac{\partial D_{\mathrm{KL}}\left(p^*\|p\right)}{\partial \mathcal{W}_k}\right]^2, \\ \hat{m}_k^t &= \frac{m_k^t}{1-\beta_1^t}, \quad \hat{v}_k^t = \frac{v_k^t}{1-\beta_2^t}, \\ \mathcal{W}_k^t &= \mathcal{W}_k^{t-1} - \eta\frac{\hat{m}_k^t}{\sqrt{\hat{v}_k^t}+\varepsilon}, \end{aligned} \tag{D.11}$$

where m_k acts as a momentum and v_k sets the adaptive learning rate. Here we follow the common choice and set the hyper-parameters to $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 10^{-8}$, [352]. We additionally multiply the adaptive learning rate with an exponentially decaying factor $\eta(t)$ from an initial value of $\eta_{\mathrm{init}} = 1$ to a minimum value of $\eta_{\mathrm{min}} = 0.001$,

$$\eta(t) = \max\left(\eta_{\mathrm{init}}\exp\left[-0.001t\right], \eta_{\mathrm{min}}\right), \tag{D.12}$$

where t counts the training epochs. Note that this learning rate is a hyper-parameter that needs to be chosen accordingly and requires a special form for the discrete-valued weights and biases on the neuromorphic hardware. With the exponentially decaying factor we ensure that the learning rate is large enough to cause changes in the weights at short time scales, but is small enough to enable convergence at later times.

In general, Hebbian training algorithms are based on minimizing the correlation mismatch between data and model distributions. The traditional way for estimating this mismatch is contrastive divergence [34, 235], where the target and model distributions are approximated by a single layer-wise network update (CD-1). This method can only be used to obtain an update of the network parameters for layer-wise connected networks and essentially represents a performance optimization with respect to sampling from the complete distributions. For physical neuromorphic systems, the notion of a "single network update" becomes meaningless and the performance characteristics make the actual sampling run cheaply as compared with the start-up cost. We take advantage of this difference by using the full model distribution for calculating updates. We thus reconstruct the target visible distribution from the model distribution by reweighting as described above. The advantage of this strategy is that arbitrary network architectures including partially restricted and deep networks can be used.

D.3 Potential applications in quantum many-body physics

Our suggested neuromorphic sampling scheme finds many possible applications in the field of quantum many-body physics. This potential is connected to recent developements in the field of neural network quantum states (NQS), which we briefly outline in the following. It has been shown that important and physically relevant classes of quantum many-body states can be approximated well using neural-network inspired variational wave functions [38, 87, 164, 242, 353, 354]. A large variety of different network architectures, including networks with real and complex parameters, and encoding pure and general mixed quantum states have been explored in this context.

NQS have been shown to approximate important classes of quantum states faithfully using a number of variational parameters that scales polynomially in the system size [38, 242, 353, 354]. For training and evaluation of observables, Monte-Carlo sampling techniques are used [38, 87]. Applications include the determination of ground and excited states of many-body problems (relevant to condensed matter physics and quantum chemistry [164, 242, 353, 354]) and simulation of quantum dynamics [353], as well as efficient methods for quantum state tomography [87, 242, 354].

In Chapter 6 we apply similar variational wave functions and replace the creation of Monte Carlo samples in software by a fast and energy efficient sampling procedure on the neuromorphic devices. By demonstrating the learnability of paradigmatic entangled quantum states on the spiking hardware we provide an important first step towards employing these devices for applications in quantum many-body problems. Using the neuromorphic device we expect a similar behavior to software NQS concerning the scalability to large system sizes while gaining a speed-up in the creation of Monte Carlo samples during training and evaluation of observables.

D.4 Implementation details of BrainScaleS-2

A network of leaky integrate-and-fire (LIF) neurons can implement a sampling spiking network (SSN) if the neurons are under stochastic noise influence, their membrane time constant is sufficiently small and the synaptic and refractory time constants roughly match [77], see also Sec. 2.3. A system-specific calibration is required to configure the analog core of BrainScaleS-2, shown in Fig. D.1a according to these requirements. For ease of implementation we use a simple routing scheme in which the on-chip network looks like 128 unique sources which can be arbitrarily connected. This allows the association of each of the 128 synapse drivers with one spike source while using the double line to implement signed synapses (cf. Fig. D.1b).

The stochastic input spikes are generated via two of the eight on-chip linear shift registers (LSFRs). We assign the spike source IDs 0-63 to the network neurons and split the spike trains from the LSFRs among the IDs 64-127. For networks smaller than 64 neurons, the upper part of (0-63) remains unused. Again simplifying the implementation we use the first half of the noise IDs (64-95) as excitatory and the second half (96-127) as inhibitory sources (cf. Fig. D.1c lower part). This scheme allows in principle all-to-all connectivity within the network. Choosing to use a layered network structure results in a block structure of the upper part of the synapse array (cf. Fig. D.1c).

Each sampling neuron is connected to 5 randomly chosen excitatory and 5 randomly chosen inhibitory noise sources. This introduces correlations between neurons even without synaptic connections, but in general does not hinder training [75, 265]. Synaptic connections on BrainScaleS-2 are 6-bit-valued circuits. The dynamical impact of a single network spike (used to mediate the stochastic response of the receiving sampling unit) onto another neuron is given by its own strength relative to the total strength of the input provided by the background sources. The latter defines the transfer function and thereby the excitability of the neurons (cf. Fig. D.1g). Choosing the noise parameters (weight and number of sources) is done such as to attain the competing goals of allowing the network neurons to drive each other significantly while allowing for small weight changes within the 6-bit resolution limit. The particular choice is, in general, problem dependent.

Having chosen the noise parameters, the sampling interface of BrainScaleS-2 becomes a black box that requires a weight (6-bit) matrix and a (10-bit) bias vector and returns a set of spike trains. Neurons are assigned a state of $z = 1$ at time t if they emitted a spike within their effective refractory period $\tau_{\text{ref}}^{\text{eff}}$ prior to t (cf. Fig. 6.1c in Sec. 6.1). We determine $\tau_{\text{ref}}^{\text{eff}}$ by setting the leak potential of the

neurons to its maximum value and measuring the resulting inter-spike intervals (cf. Fig. D.1e). The effective refractory time consists of the clamped part which is digitally driven and therefore does not vary between different neurons and the drift part back to the spiking threshold in the end. Due to the circuit variability (e.g. different membrane time constants) of the analog circuits we see some modest variation in $\tau_{\text{ref}}^{\text{eff}}$ (cf. Fig. D.1f). Using the measured $\tau_{\text{ref}}^{\text{eff}}$ we assign a state every 2 µs and use the set of these states for the evaluation and the update calculation (cf. App. D.2).

Fig. D.1h demonstrates the correctness of an approximated distribution for a simulated sampling spiking network (using [355]) as a function of the number of samples for different state assign times dt (cf. Fig. 6.1 in Sec. 6.1). For more than two samples per refractory period τ_{ref} the number of samples required to achieve a given performance level increases due to the correlated states as expected from the Nyquist-Shannon theorem. Both the noise parameters and the sample frequency were chosen such that they enable sufficiently accurate sampling, but without performing an exhaustive optimization.

As discussed above, a chip-specific calibration is required but can be reused for each training. For each experiment the chip needs to be initialized (blue period in Fig. D.1d) once. This ensures that the correct calibration is loaded and the routing is configured correctly before the training iterations (orange period in Fig. D.1d) can start. After the initialization only the synapse array (weights) and the leak potentials of the neurons (biases) are reconfigured once per epoch (green period in Fig. D.1d). Each training epoch consists of 26 sampling runs (red section in Fig. D.1d) and a single calculation of the parameter update (purple in Fig. D.1d). In each hardware run we build a program for the FPGA to execute (dark red in Fig. D.1d), transfer it to the FPGA with some initial buffering (yellow in Fig. D.1d) in order to compensate for network latencies, perform the actual execution on chip (light blue in Fig. D.1d) and transfer the spikes back to the host computer (grey in Fig. D.1d).

In total, an epoch takes about 1.9 s of which roughly half is spent in the sampling and the other half is used to calculate the parameter updates. While some time was spent to improve performance, both parts can still be optimized. For example the gradient calculation is implemented in Python and most of the sampling time is spent buffering and reading back the results. The actual hardware runtime is only 30 % of the time marked as *HW-run* in Fig. D.1d. Using a more complex routing setup an increase to at least 256-spike sources is possible and since BrainScaleS-2 is a physical system the runtime of the hardware part is not affected by the network size.

D.5 Computation time benchmark for sampling from neural networks

In this appendix, we provide a speed comparison between the BrainScaleS-2 neuromorphic chip and a C++-implemented software solution to the sampling from binary Boltzmann machines. The software implements standard Gibbs sampling, i.e. it sequentially calculates the "membrane potential" $u_i = b_i + \sum_i W_{ki} z_i$ for each neuron and assigns a new state $z_i = 1$ with probability $\sigma(u_i) = \frac{1}{1+\exp -u_i}$ and $z_i = 0$ otherwise. This implementation, while fairly optimized in single-thread performance, does not take into account the potential parallelism of a layered structure. Since the simulator is optimized for large-scale systems it drops all terms with $W_{ki} = 0$, at the price of an additional indirection. The sum now runs over a list of indices which is harder to optimize than a simple sequential iteration. We executed this on the bwForCluster NEMO cluster [356] which uses Intel Xeon E5-2630v4 (Broadwell) CPUs.

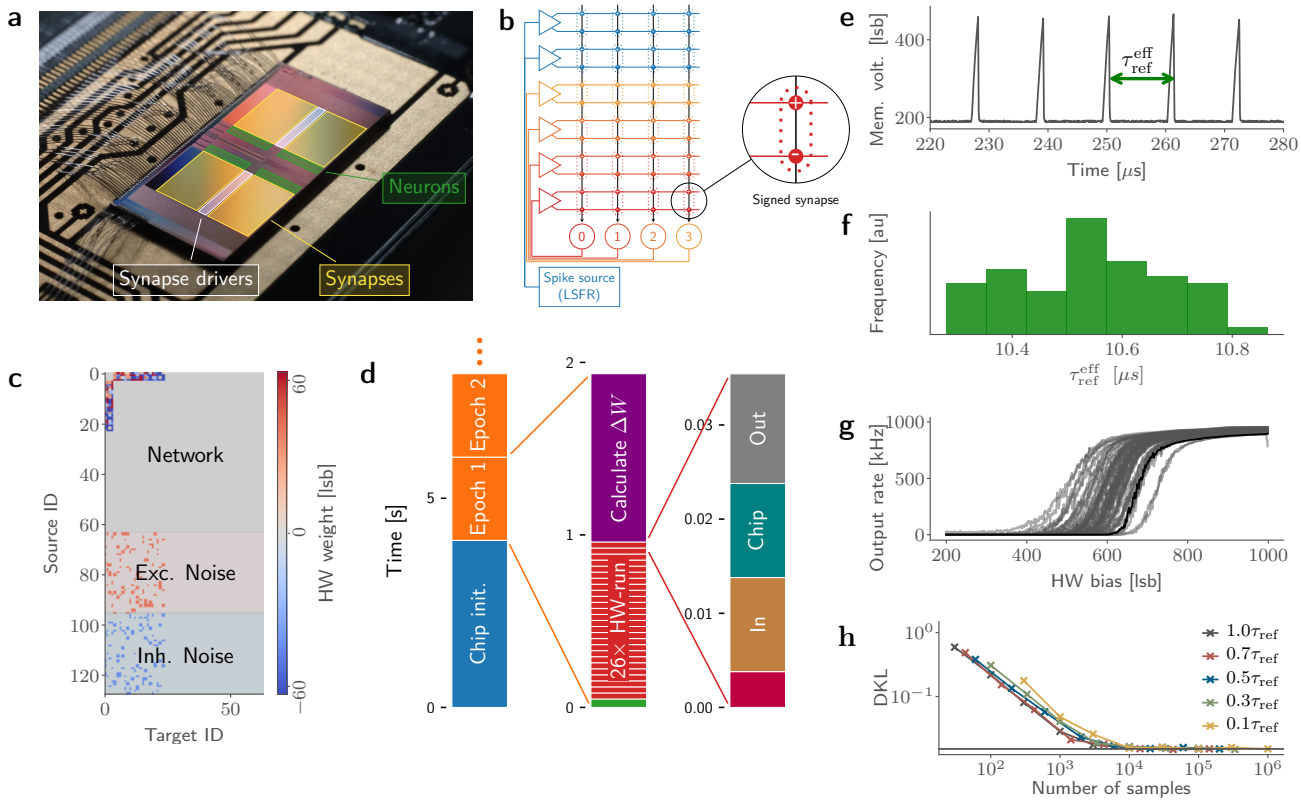

Figure D.1: **Details of the BrainScaleS-2 neuromorphic chip.** **a**, Photograph of the BrainScaleS-2 chip with circuits of 4×128 AdEx-LIF neurons (green), 2×2×128 synapse drivers (white) and 4 synapse arrays with 256×128 synapses (yellow). **b**, Routing schematic used to implement the sampling spiking network. Each synapse driver projects to two synapse rows in order to allow signed synapses. **c**, Utilized logical connectivity matrix projecting onto the 64 neurons used. Network (neuron-to-neuron) connections are truncated at index 24 (4 visible and 20 hidden) and intra-layer connections are not used. Each neuron receives noise input from 5 excitatory (64-95) and 5 inhibitory (96-127) sources, generated by one on-chip LSFR each. Each connection selects the appropriate synapse row depending on its sign (cf. **b**) **d**, Time usage across a training experiment. The initial configuration (blue) of the chip is comparable to a single epoch (orange). Each epoch consists of a parameter update (green), 26 sampling runs (red) and the update calculation (purple). Each hardware run consist of the construction of the playback program (ruby), the initial buffering on the FPGA (brown), the actual chip runtime (turquoise) and the readout to the host (grey). **e**, Membrane trace of an exemplary neuron at the high-bias end. $\tau_{\text{ref}}^{\text{eff}}$ is the inter spike interval. **f**, Histogram of measured $\tau_{\text{ref}}^{\text{eff}}$. Variations are due to the analog nature of the system. **g**, Activation functions as a function of the leak potential under noise input of the 64 neurons used. $\tau_{\text{ref}}^{\text{eff}}$ is estimated by the output frequency at the high-bias end. **h**, Sampling performance as a number of samples, rather than execution time for different sampling time deltas dt. More than two samples per refractory time $\tau_{\text{ref}}^{\text{eff}} \approx 10\,\mu\text{s}$ increase the Kullback-Leibler divergence as the samples are not independent.

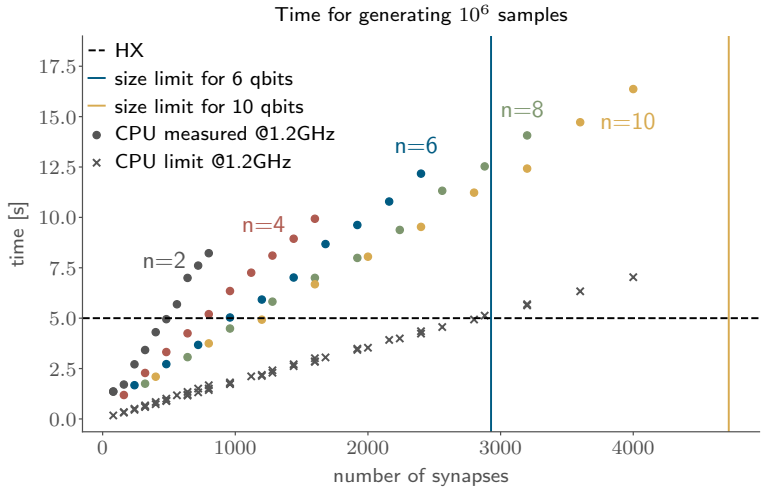

Figure D.2: Measured (dots) and estimated (crosses) sampling times for the generation of a million samples, for different quantum system size ($N = 2,\ 4,\ 6,\ 8,\ 10$ spins, colours) and hidden layer sizes ($M = 20,\ 40,\ 60,\ 80,\ 100,\ 120,\ 140,\ 160,\ 180,\ 200$) on a Intel Xeon E5-2630v4 compared to the constant runtime of the BrainScaleS-2 system (horizontal line). The software time estimation assumes one FLOP per clock cycle and one FLOP required per synaptic interaction, bias and state assignment (see text). The number of neurons on BrainScaleS-2 is limited to $(2N) + M < 256$ which limits the implementable system size (vertical lines, for 6 spins and 244 hidden neurons and 10 spins and 236 hidden neurons).

Generating a new state requires the update of all neurons, and each update of a single neuron requires the calculation of u_i plus a comparison with a random number for the probabilistic update. For the architecture used in Chapter 6, i.e. layered networks with $2N$ visible and M hidden neurons and assuming a perfect implementation without additional cost for memory accesses, generating a new update takes $2(2N)M$ evaluations and additions of the term $W_{ki}z_i$, besides $(2N) + M$ additions of b_i, and $(2N) + M$ comparisons to a random number. Assuming further that each of these steps takes one clock cycle, we can estimate the expected time required.

In order to reduce the impact of the initialization of the software sampler (loading of the network configuration and initialization) we measure the time to generate 10^6 samples. We note that the number of operations per update is dominated by the number of connections (synapses) $2(2N)M$. As such, the time required scales linearly in the number of hidden units only for a fixed number of visible units, which is given by the size of the physical system (cf. Fig. D.2).

On the other hand, the BrainScaleS-2 implementation, due to its inherently parallel architecture, requires a sample generation time that is independent of the size of the sampled network. With $\tau_{\text{ref}}/2 = 5\,\mu\text{s}$ per sample (cf. Fig. D.1h), this leads to a constant time of 5 s. This constant scaling is only true if the network fits onto the system (up to 256 sampling neurons). Since the number of visible neurons is given by the size of the physical system that is represented (N spins), larger physical systems give a greater speedup. Already for the case of 8 spins (16 visible units and 180 hidden units) the fixed runtime of the BrainScaleS-2 system is exceeded by our estimation from the idealized software estimate (cf. Fig. D.2). Larger system sizes will skew this comparison further to favor of BrainScaleS-2, which can even implement more densely connected network topologies without incurring a performance penalty. We also note that the BrainScaleS-2 chip requires less than 500 mW [69, 246], while the Intel Xeon E5-2630v4 has a thermal design power (TDP) of 85 W for 10 cores. As such BrainScaleS-2 is using comparable energy even for the smallest systems we

implemented in the prototype system used in the main manuscript. While the system size at which the BrainScaleS-2 chip outperforms CPU implementations may shift to larger values when comparing to the fastest currently available CPUs, the fundamental difference in scaling behavior, i.e. constant v.s. linear, persists.

APPENDIX E

Unsupervised neural graph embedding

Name	CenterModule	Number of parameters
MNIST Featurizer	Conv2D(1, 16, 3, 1, 1) ⇒ ReLU ⇒ Conv2D(16, 32, 3, 2, 1) ⇒ Tanh ⇒ FC(800, 128) ⇒ ReLU ⇒ FC(128, 16)	109×10^3
Graph Data Featurizer	GCNConv(6, 32) ⇒ ReLU ⇒ GCNConv(32, 6) ⇒ Global mean pooling ⇒ ReLU ⇒ FC(16, 64) ⇒ ReLU ⇒ FC(64, 16)	2.9×10^3
Imitator	FC(16, 12) ⇒ ReLU ⇒ μ: FC(12, 2); $\log \sigma^2$: FC(12, 2) ⇒ FC(2, 12) ⇒ ReLU ⇒ FC(12, 16)	500
PostTrain InfoVAE	FC(16, 64) ⇒ ReLU ⇒FC(64, 128) ⇒ μ: FC(128, 2); $\log \sigma^2$: FC(128, 2) ⇒ FC(2, 128) ⇒ ReLU ⇒ FC(128, 64) ⇒ ReLU ⇒ FC(64, 16)	19.6×10^3
MNIST VAE	Conv2D(1, 16, 3, 1, 1) ⇒ ReLU ⇒ Conv2D(16, 32, 3, 2, 1) ⇒ Tanh ⇒ FC(800, 64) ⇒ ReLU ⇒ μ: FC(64, 2); $\log \sigma^2$: FC(64, 2) ⇒ FC(2, 64) ⇒ ReLU ⇒ FC(64, 800) ⇒ ReLU ⇒ ConvTranspose2D(32, 16, 3, 2, 1, 1) ⇒ ReLU ⇒ ConvTranspose2D(16, 1, 3, 1, 1)	113×10^3

Table E.1: Details on the implemented network architectures for the generation of embeddings. The arguments of FC denote the number of input and output neurons of a fully-connected feed-forward network. The terminology for the remaining layers is adopted from PyTorch [357] and PyTorch Geometric [358].

Simulation	MNIST NAE	MNIST NAE PostTrain	MNIST InfoVAE	Graph NAE	Graph NAE PostTrain
γ	0.0625	-	-	0.0625	-
κ	0.1	-	-	0.1	-
α	0.001	0.001	0.001	0.001	0.001
λ	0.9999	0.9999	0.9999	0.9999	0.9999
Ratio $N_{\mathrm{Im}}/N_{\mathrm{F}}$	29/1	-	-	29/1	-
Learning rate $\mathcal{L}_{\mathrm{F}}$	2×10^{-4}	-	-	5×10^{-3}	-
Learning rate $\mathcal{L}_{\mathrm{Im}}$ / $\mathcal{L}_{\mathrm{InfoVAE}}$	5×10^{-3}	5×10^{-4}	5×10^{-4}	5×10^{-3}	5×10^{-4}

Table E.2: Hyperparameters for training the different networks for analysing neural adversarial graph embeddings.

APPENDIX F

Spectral reconstruction

This appendix is based on Ref. [2].

F.1 BR method

Different Bayesian methods propose different prior probabilities, i.e. they encode different types of prior information. The well-known Maximum Entropy Method e.g. features the Shannon-Jaynes entropy

$$S_{\mathrm{SJ}} = \int d\omega \big(\rho(\omega) - m(\omega) - \rho(\omega)\log\big[\frac{\rho(\omega)}{m(\omega)}\big]\big), \tag{F.1}$$

while the more recent BR method uses a variant of the gamma distribution

$$S_{\mathrm{BR}} = \int d\omega \big(1 - \frac{\rho(\omega)}{m(\omega)} + \log\big[\frac{\rho(\omega)}{m(\omega)}\big]\big). \tag{F.2}$$

Both methods e.g. encode the fact that physical spectral functions are necessarily positive definite but are otherwise based on different assumptions.

As Bayesian methods they have in common that the prior information has to be encoded in the functional form of the regulator and the supplied default model $m(\omega)$. Note that discretising ρ by choosing a particular functional basis also introduces a selection of possible outcomes. The dependence of the most probable spectral function, given input data and prior information, on the choice of S, $m(\omega)$ and the discretised basis comprises the systematic uncertainty of the method.

One major limitation to Bayesian approaches is the need to formulate our prior knowledge in the form of an optimisation functional. The reason is that while many of the correlation functions relevant in theoretical physics have very well defined analytic properties it has not been possible to formulate these as a closed regulator functional S. Take as an example the retarded propagator (for a more comprehensive discussion see [339]). Its analytic structure in the imaginary frequency plane splits into two parts, an analytic half-plane, where the Euclidean input data is located, and a meromorphic half-plane which contains all structures contributing to the real-time dynamics. Encoding this information in an appropriate regulator functional has not yet been achieved.

Instead the MEM and the BR method rather use concepts unspecific to the analytic structure, such as smoothness, to derive their regulators. Among others this e.g. manifests itself in the presence of artificial ringing, which is related to unphysical poles contributing to the real-time propagator, which however should be suppressed by a regulator functional aware of the physical analytic properties.

F.2 GrHMC method

The main idea of the setup is already stated in the main text in Sec. 12.1 and was first introduced in [339]. Nevertheless, for completeness we outline the entire reconstruction process here. The approach is based on formulating the basis expansion in terms of the retarded propagator. The resulting set of basis coefficients are then determined via Bayesian inference. This leaves us with two objects to specify in the reconstruction process, the choice of a basis/ansatz for the retarded propagator and suitable priors for the inference.

Once a basis has been chosen it is straightforward to write down the corresponding regression model. As in the reconstruction with neural nets we use a fixed number of Breit-Wigner structures, c.f. (12.7), corresponding to simple poles in the analytically continued retarded propagator. The logarithm of all parameters is used in the model in order to enforce positivity of all parameters. The uniqueness of the parameters is ensured by using an ordered representation of the logarithmic mass parameters.

The other crucial point is the choice of priors, which are of great importance to tame the ill-conditioning practically and should therefore be chosen as restrictive as possible. For comparability to the neural net reconstruction, the priors are matched to the training volume in parameter space. However, it is more convenient to work with a continuous distribution. Hence the priors of the logarithmic parameters are chosen as normal distributions where we have fixed the parameters by the condition that the mean of the distribution is the mean of the training volume and the probability at the boundaries of the trainings volume is equal. Details on the training volume in parameter space can be found in Tab. 12.1 and Sec. F.3.

All calculations for the GrHMC method are carried out using the python interface [359] of Stan [360].

F.3 Mock data, training set and training procedure

We consider three different levels of difficulty for the reconstruction of spectral functions to analyse and compare the performance of the approaches in this work. These levels differ by the number of Breit-Wigners that need to be extracted based on the given information of the propagator. We distinguish between training and test sets with one, two and three Breit-Wigners. A variable number of Breit-Wigners within a test set entails the task to determine the correct number of present structures. This can be done a priori or a posteriori based either on the propagator or on the quality of the reconstruction. While it is straightforward to implement this for the PoNet, it is not completely clear how one should proceed for the PaNet. We postpone this problem to future work.

The training set is constructed by sampling parameters uniformly within a given range for each parameter. The ranges for the parameters of a Breit-Wigner function of (12.7) are as follows: $M \in [0.5, 3.0]$, $\Gamma \in [0.1, 0.4]$ and $A \in [0.1, 1.0]$. In addition, we investigate the impact of the size of the parameter space on the performance of the network for the case of two Breit-Wigner functions.

BR Comparison	A	M	Γ
1BW	$[0.1, 1.0]$	$[0.5, 3.0]$	$[0.1, 0.4]$
2BWa	$[0.2, 1.8]$	$[0.8, 3.8]$	$[0.2, 1.0]$
2BWb	$[0.3, 1.2]$	$[0.8, 3.8]$	$[0.002, 0.02]$
3BW	$[0.2, 1.8]$	$[1.0, 6.0]$	$[0.2, 1.0]$

Table F.1: Parameter ranges for the training of the neural networks for the comparison in Fig. 12.11.

This is done by decreasing the ranges of the parameters Γ and A gradually. We proceed differently for the two masses to guarantee a certain finite distance between the two Breit-Wigner peaks. Instead of decreasing the mass range, the minimum and maximum distance of the peaks is restricted. Details on the different parameter spaces were stated in Tab. 12.1. The propagator function is parametrised by $N_p = 100$ data points that are evaluated on a uniform grid within the interval $\omega \in [0, 10]$. For a training of the point net, the spectral function is discretised by $N_\omega = 500$ data points on the same interval. Details about the training procedure can be found at the end of the section. The parameter ranges deviate for the comparison of the neural network approach with existing methods. The corresponding ranges are listed in Tab. F.1. To avoid any confusion, Tab. F.3 provides a comprehensive list of all figures with the associated model details and parameter ranges.

The different approaches are compared by a test set for each number of Breit-Wigners consisting of 1000 random samples within the parameter space. Another test set is constructed for two Breit-Wigners with a fixed scaling $A_1 = A_2 = 0.5$, a fixed mass $M_1 = 1$ and equally chosen widths $\Gamma := \Gamma_1 = \Gamma_2$. The mass M_2 and the width Γ are varied according to a regular grid in parameter space. This test set allows the analysis of contour plots of different loss measures. It provides more insights into the minima of the loss functions of the trained networks and into the severity of the inverse problem. The contour plots are averaged over 10 samples for the noise width of 10^{-3} (except for Fig. 12.10).

We investigate three different performance measures and different setups of the neural network for a comparison to existing methods. The root-mean-square-deviation of the predicted parameters in parameter space, of the reconstructed spectral function and of the reconstructed propagator are considered. For the latter case, the error is computed based on the original propagator without noise. The spectral function loss and the propagator loss are computed based on the discretised representations on the uniform grid. Representative error bars for all methods are depicted in Fig. 12.9.

The training procedure for the neural networks in this work is as follows. A separate neural network is trained for each training set, i.e., for each error and for each range of parameters. The learning rates are between 10^{-5} and 10^{-7}. The batch size is between 128 and 500 and the number of generated training samples per epoch is around 6×10^5. Depending on the kind of network, the nets are trained for 80 to 160 epochs. The used loss functions are described at the end of Sec. 12.2.2. The implemented net architectures are provided in Tab. F.2. Details about the training of the different networks and about the respective utilized test set for the evaluation can be found in Tab. F.3 for each figure.

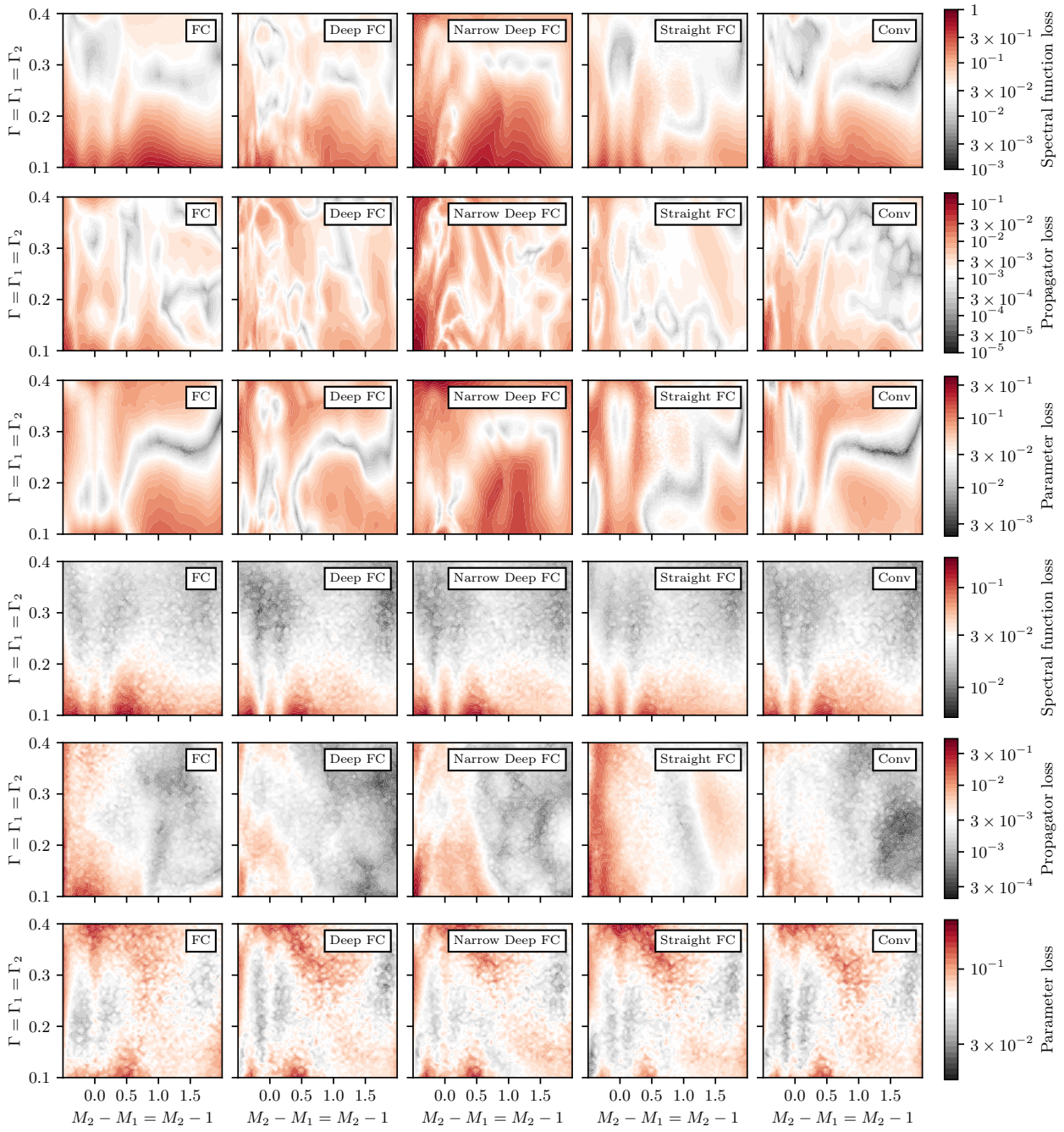

Figure F.1: **Comparison of network architectures -** Contour plots of loss measures are shown for different net architectures. The upper three rows correspond to reconstructions of propagators with a noise width of 10^{-5}, the lower ones with 10^{-3}. The plots illustrate the loss measures in a hyperplane within the parameter space whose properties are described in Sec. F.3. The networks are trained with the parameter loss on the training set of volume Vol O. The contour plots show that the local minima are slightly different for small noise widths, whereas the global structures remain similar for all network architectures. These differences are caused by a slightly differing utilization of the limited number of hyperparameters. The differences between the network architectures become less visible for larger errors due to the growing severity of the inverse problem and a decreasing knowledge about the correct inverse transformations. Interestingly, the loss landscape of the convolutional neural network, which intrinsically operates on local structures, and of the fully connected networks are almost equal. The non-locality of the inverse integral transformation represents a possible reason for why the specific choice of the network structure is largely irrelevant. We conclude that the actual architecture is rather negligible in comparison to other attributes of the learning process, such as the selection of training data and the choice of the loss function.

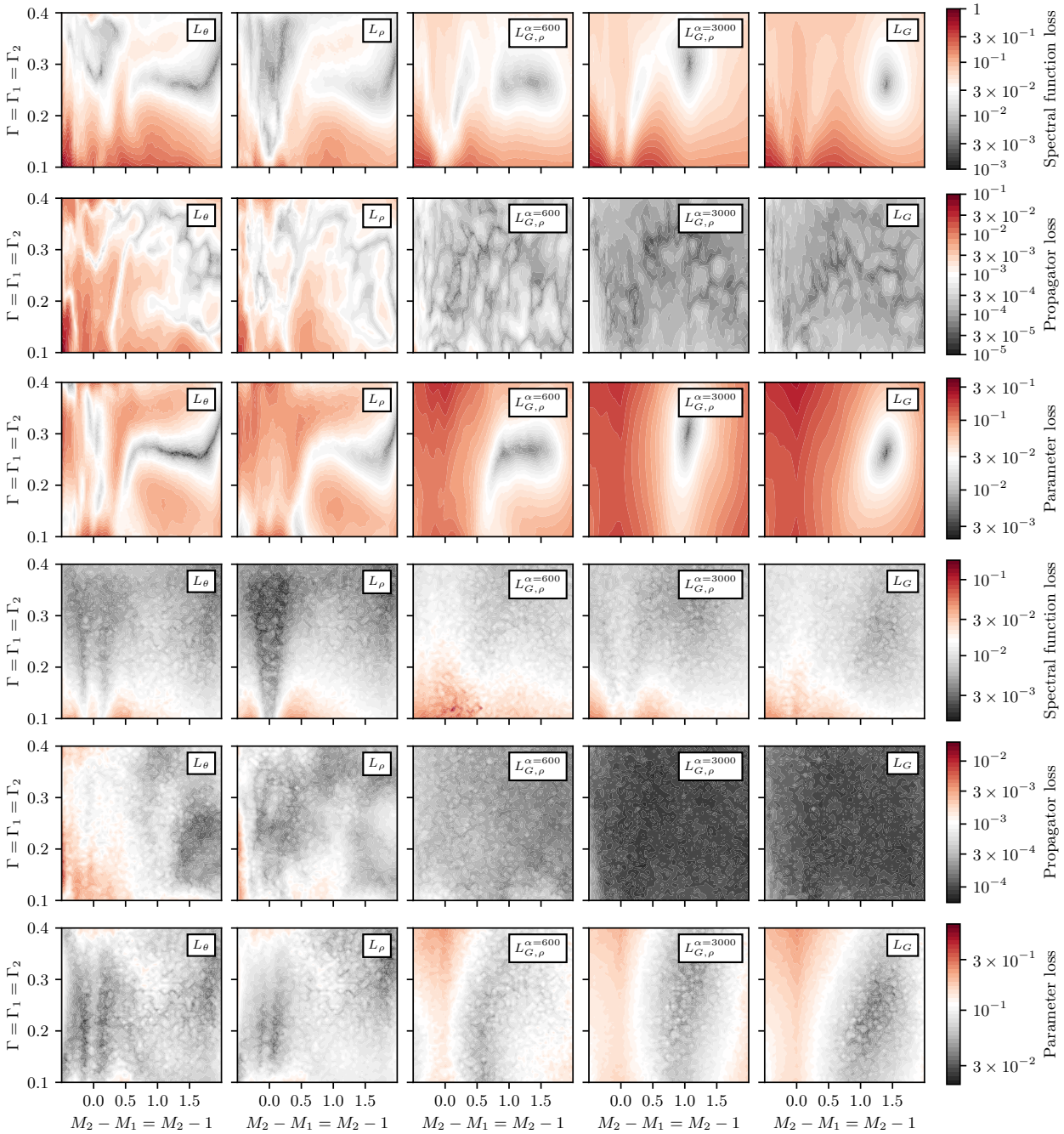

Figure F.2: **Comparison of loss functions -** Contour plots of loss measures are illustrated in the same manner as in Fig. F.1, but with a comparison o different loss functions. The considered loss functions are introduced in Sec. 12.2.2. The results are based on the Conv PaNet that is trained on volume Vol O. The optima in the loss function differ and, consequently, lead to different mean squared errors for the different measures. It is interesting that the network with the pure propagator loss function leads to a rather homogeneous propagator loss distribution. In contrast, the networks with the pure parameter and the pure spectral function loss do not result in homogeneous distributions for their corresponding loss function. The large set of nearly equal propagators for different parameters explains this observation. It confirms also once more the necessity of approaches that can be trained using loss functions with access to more information than just the reconstructed propagator.

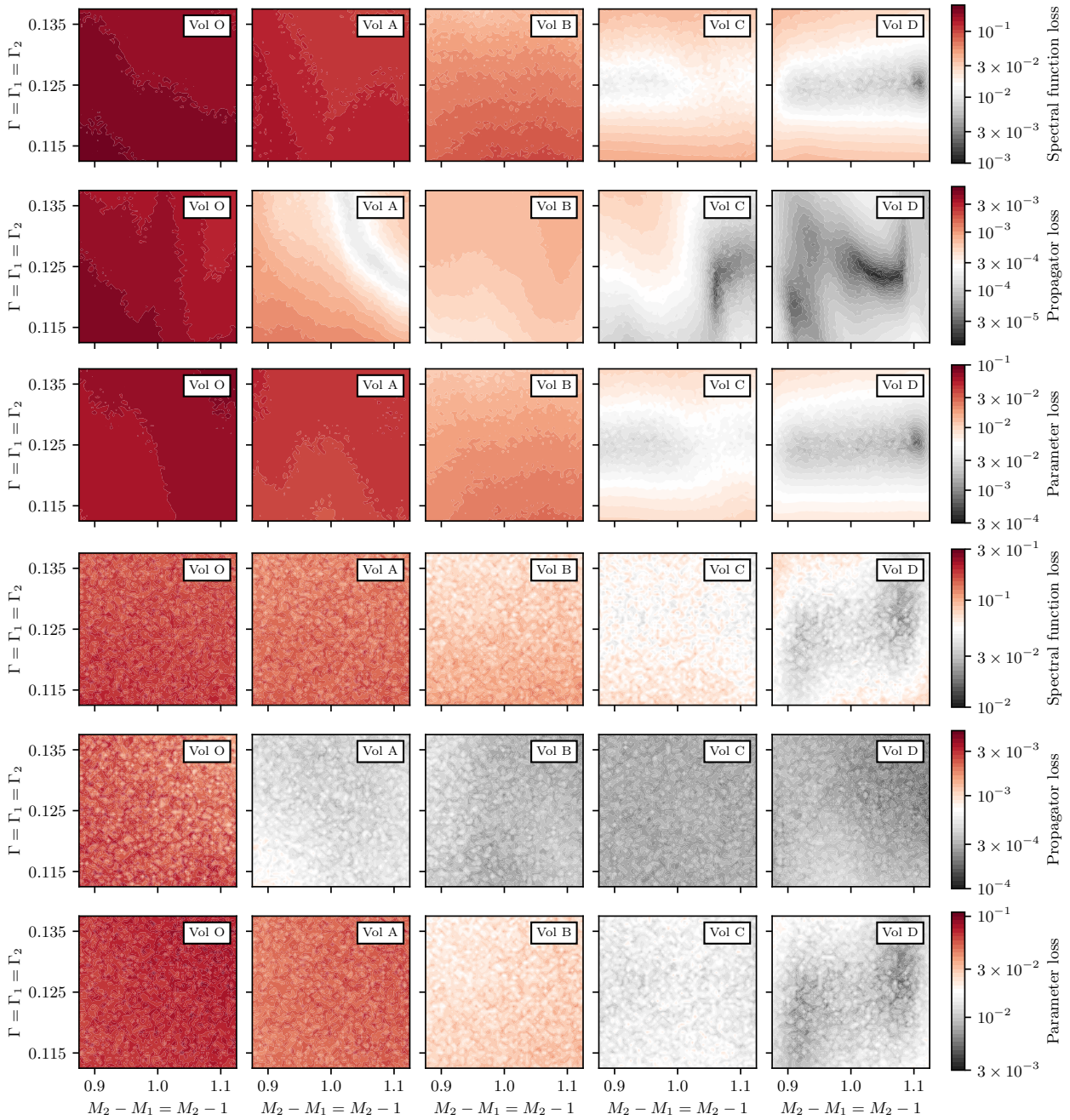

Figure F.3: **Analysis of prior information (parameter space of the training data) and of local differences in the severity of the inverse problem -** The evolution of the landscape of different loss measures is shown for Conv PaNets that are trained on different parameter spaces. All contour plots are based on the same section of the parameter space, namely the range that is spanned by volume D. The upper three and lower rows correspond again to reconstructions of propagators with noise widths 10^{-5} and 10^{-3}. The gradual reduction of the parameter space allows the analysis of different levels of complexity of the problem. A general improvement of performance can be observed besides a shift of the global optima. The more homogeneous loss landscape demonstrates that the problem of a different severity of the inverse problem is still present, but damped.

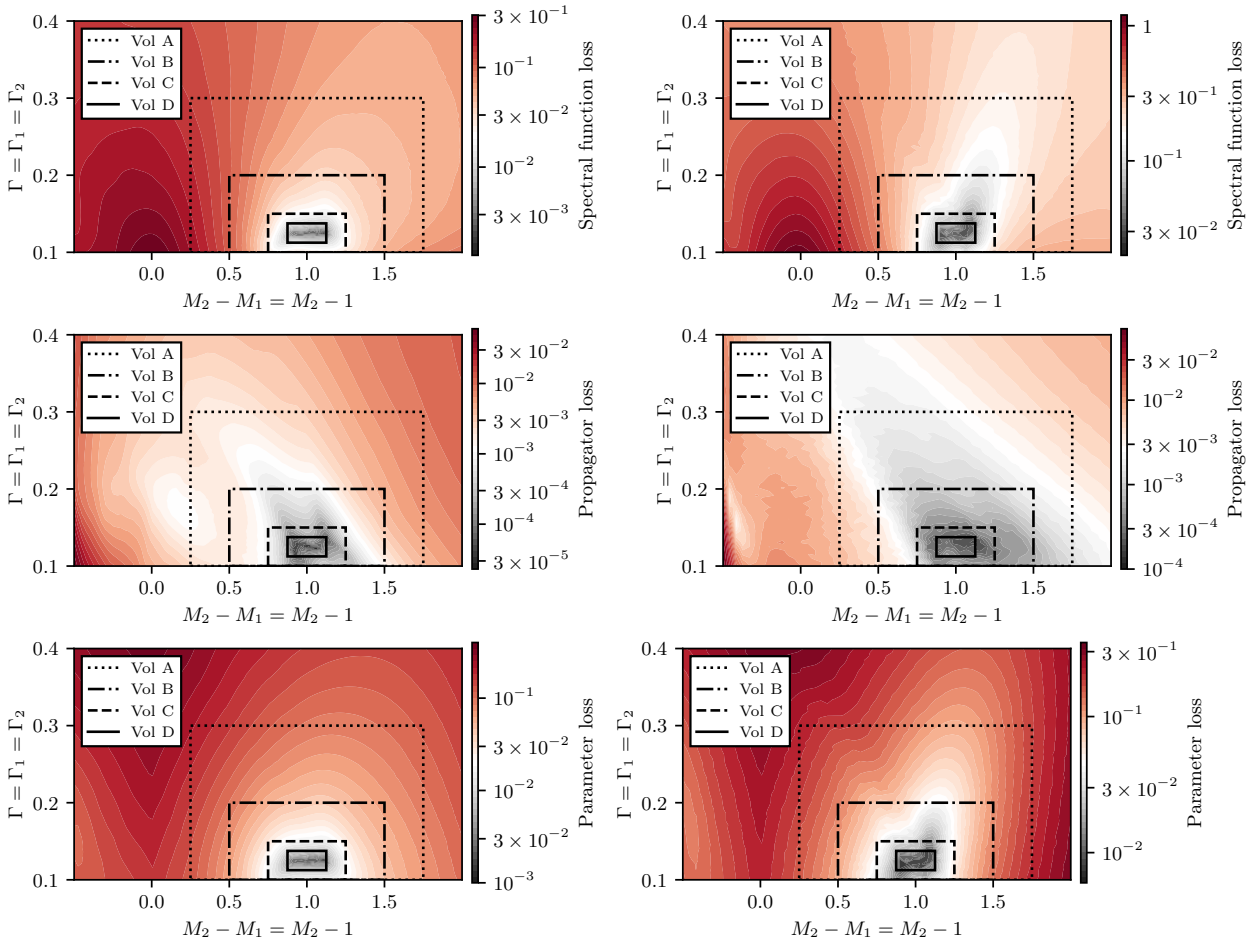

Figure F.4: **Performance outside of the training region -** Performance of the Conv PaNet trained on the smallest volume Vol D for data that lies outside of the training region with a noise width of 10^{-5} (left column) and a noise width of 10^{-3} (right column). As expected, the prediction quality decreases with distance from the boundaries of Vol D. However, we emphasise that there is no immediate sharp transition at the boundary. Instead, we observe at first only a gradual decrease of the prediction quality, indicating that the network can generalise slightly beyond the trained region to varying degrees, depending on which parameters and error metrics are considered.

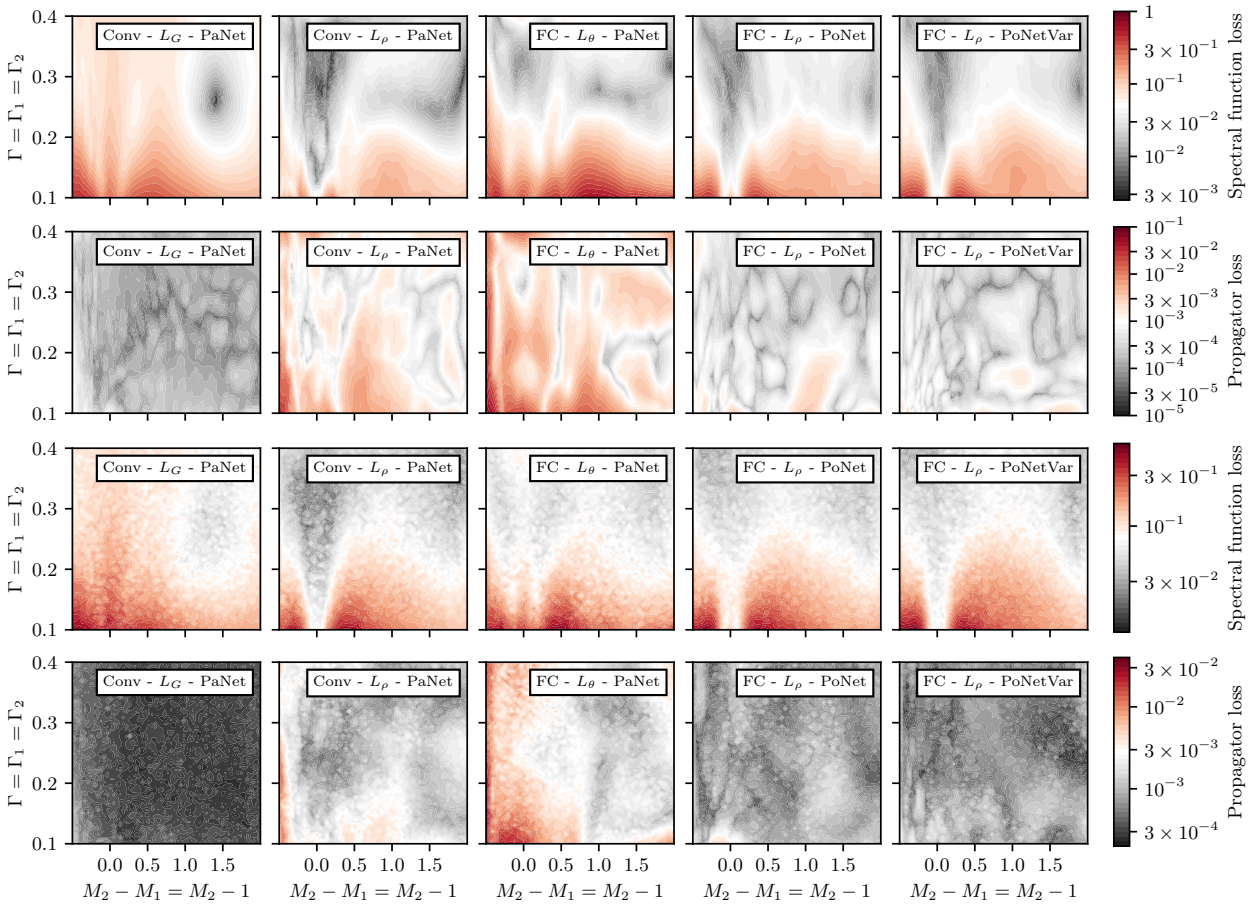

Figure F.5: **Comparison of the parameter net and the point net -** Root-mean-squared-deviations are compared between the parameter net and the point net, trained on two Breit-Wigner like structures (PoNet) and trained on a variable number of Breit-Wigners (PoNetVar), with respect to different loss functions. The two upper rows correspond to results from input propagators with a noise width of 10^{-5} and the two lower ones with a noise width of 10^{-3}. Problems concerning a varying severity of the inverse problem and concerning an information loss caused by the additive noise remain independent of the chosen basis for the representation of the spectral function.

Name	CenterModule	Number of parameters
FC	FC(6700) ⇒ ReLU ⇒ FC(12168) ⇒ ReLU ⇒ FC(1024)	95×10^6
Deep FC	FC(512) ⇒ ReLU ⇒ FC(1024) ⇒ ReLU ⇒ $(\text{FC}(4056) \Rightarrow \text{ReLU})^3 \Rightarrow (\text{FC}(2056) \Rightarrow \text{ReLU})^2$	50×10^6
Narrow Deep FC	FC(512) ⇒ ReLU ⇒ $(\text{FC}(1024) \Rightarrow \text{ReLU})^3 \Rightarrow (\text{FC}(2056) \Rightarrow \text{ReLU})^5 \Rightarrow (\text{FC}(1024) \Rightarrow \text{ReLU})^3$ ⇒ FC(512) ⇒ ReLU ⇒ FC(256)	96×10^6
Straight FC	$(\text{FC}(4112) \Rightarrow \text{BatchNorm1D} \Rightarrow \text{ReLU} \Rightarrow \text{Dropout}(0.2))^7$	102×10^6
Conv	Conv(64, 10) ⇒ ReLU ⇒ Conv(256, 10) ⇒ ReLU ⇒ $(\text{FC}(4096) \Rightarrow \text{ReLU})^2$ ⇒ FC(1024)	41×10^6

Table F.2: Details on the implemented network architectures. The argument of FC denotes the number of output neurons. The numbers in the argument of Conv correspond to the number of output channels and to the kernel size. The general setup is: Input(100) ⇒ ReLU ⇒ CenterModule ⇒ ReLU ⇒ FC(3/6/9/500) ⇒ Output, where the CenterModule is given along with the associated name in the table. The size of the output layer is determined by the use of a parameter net or a point net and the considered number of Breit-Wigners.

Figure	Network type	Architecture	Loss function	Training set	Test set	Noise width	Number of BWs
Fig. 12.1	PaNet	FC	L_θ	Vol O	Noise samples on same propagator	10^{-3}	1-3
Fig. 12.3	PaNet	Various (a) / Conv (b)	L_θ (a) / Various (b)	Vol O	Vol O	10^{-3} / 10^{-5}	2
Fig. 12.4	PaNet	Conv	L_θ	Various	Noise samples on same propagator	10^{-3}	2
Fig. 12.5	PaNet	Conv / Conv PP	L_θ	Various	Vol D	Various	2
Fig. 12.6	PaNet	Conv	L_θ	Vol D	Various	Various	2
Fig. 12.7	Various	FC	Various	Vol O	Vol O	Various	1-3
Fig. 12.8	PaNet	Conv	L_θ	Vol B	Noise samples on same propagator	Various	2
Fig. 12.9	PaNet	FC / FC PP	L_θ	Vol O	Vol O	Various	1-3
Fig. 12.10	PaNet	FC	L_θ	Vol O	Contour - Vol O	10^{-3}	2
Fig. 12.11	PaNet	Conv	L_θ	See Tab. F.1	Specific sets	10^{-3}	1-3
Fig. F.1	PaNet	Various	L_θ	Vol O	Contour - Vol O	10^{-3} / 10^{-5}	2
Fig. F.2	PaNet	Conv	Various	Vol O	Contour - Vol O	10^{-3} / 10^{-5}	2
Fig. F.3	PaNet	Conv	L_θ	Various	Contour - Vol D	10^{-3} / 10^{-5}	2
Fig. F.4	PaNet	Conv	L_θ	Vol D	Contour - Vol O	10^{-3} / 10^{-5}	2
Fig. F.5	Various	Conv / FC	Various	Vol O	Contour - Vol O	10^{-3} / 10^{-5}	2

Table F.3: List of figures that contains details about the training of the different networks and about the dataset used for evaluation/validation.

Acknowledgements

First of all, I want to thank my supervisor Jan M. Pawlowski. The freedom he gave me to follow my own and our shared research interests is truly appreciated. He always guided me into the right directions by his expertise and astonishing understanding of physics and by his ability to quickly enter and comprehend new topics and related problems.

I am grateful to Prof. Dr. Manfred Salmhofer for being my second referee.

I would like to thank the people of the Electronic Visions group, in particular, Andreas Baumbach and Mihai M. Petrovici, for their great support and insightful discussions about the BrainScaleS system. Furthermore, I would also like to thank my collaborators Stefanie Czischek, Thomas Gasenzer, Martin Gärttner and all the other members of CP5 for long and interesting meetings about quantum systems, complex Langevin dynamics and neuromorphic systems.

I would like to thank the members of the machine learning and the lattice meeting group, including Stefan Blücher and Nils Strodthoff, for insightful and versatile discussions.

I would like to thank Ulrich Köthe, Manfred Salmhofer, Julian Urban and Jan M. Pawlowski for enjoyable discussions about machine learning and the renormalization group in the garden of the institute.

I would like to thank the Institute for Pure and Applied Mathematics, UCLA, and the organisers of the long program "Machine Learning for Physics and the Physics of Learning" for giving me the opportunity to spend time in a great environment and to have inspiring discussions with people from all over the world.

I would like to thank all the people in the group of Jan M. Pawlowski for the interesting talks during the coffee breaks, the barbecue evenings at the institute and for making my time in the roof chamber of the Philosophenweg 16 enjoyable.

I am grateful to Andreas Baumbach, Marc Bauer, Bruno Faigle-Cedzich, Konstantin Gerbig, Jan Horak, Markus Heller, Moritz Hoffmann, Klaus Kades, Daniela Moratscheck, Eric Müller, Coralie Schneider, Kirill Shmilovich and Nicolas Wink for proofreading my thesis. Special thanks go to Johannes Lumma for giving valuable feedback on the introduction, the abstract and the conclusion.

I would like to thank Kirill Shmilovich and Marc Stieffenhofer for being great flatmates during our stay in Los Angeles.

Lastly, I want to thank my family, all my friends and my girlfriend for their endless support.

Bibliography

[1] L. Kades and J. M. Pawlowski, Discrete langevin machine: Bridging the gap between thermodynamic and neuromorphic systems, Phys. Rev. E **101**, 063304 (2020).

[2] L. Kades, J. M. Pawlowski, A. Rothkopf, M. Scherzer, J. M. Urban, S. J. Wetzel, N. Wink, and F. P. G. Ziegler, Spectral reconstruction with deep neural networks, Phys. Rev. D **102**, 096001 (2020).

[3] S. Blücher, L. Kades, J. M. Pawlowski, N. Strodthoff, and J. M. Urban, Towards novel insights in lattice field theory with explainable machine learning, Phys. Rev. D **101**, 094507 (2020).

[4] S. Czischek, A. Baumbach, S. Billaudelle, B. Cramer, L. Kades, J. M. Pawlowski, M. K. Oberthaler, J. Schemmel, M. A. Petrovici, T. Gasenzer, and M. Gärttner, Spiking neuromorphic chip learns entangled quantum states (2021), arXiv:2008.01039 [cs.ET] .

[5] L. Kades, M. Gärttner, T. Gasenzer, and J. M. Pawlowski, Towards sampling complex actions (2021), arXiv:2106.09367 [hep-lat] .

[6] A. Baumbach, L. Kades, J. M. Pawlowski, M. A. Petrovici, and J. Schemmel, Complex Langevin dynamics on a neuromorphic device, (in preparation).

[7] L. Kades and K. Shmilovich, Unsupervised neural graph embedding, (in preparation).

[8] A. Hosak, L. Kades, and J. M. Pawlowski, Step-wise reweighting criterion for correctness of complex Langevin dynamics, (in preparation).

[9] M. Bauer, L. Kades, and J. M. Pawlowski, Self-consistent sampling of complex actions, (in preparation).

[10] L. Kades and J. M. Pawlowski, An easy to use Markov chain Monte Carlo sampling framework for lattice field theories, (in preparation).

[11] K. Höfling, L. Kades, M. Reichert, J. M. Pawlowski, and F. Sadlo, Visualizing the functional renormalization group - Finding fixed points in high-dimensional spaces, (in preparation).

[12] G. G. Batrouni, G. R. Katz, A. S. Kronfeld, G. P. Lepage, B. Svetitsky, and K. G. Wilson, Langevin simulations of lattice field theories, Phys. Rev. D **32**, 2736–2747 (1985).

[13] P. H. Damgaard and H. Huffel, Stochastic quantization, Phys. Rept. **152**, 227 (1987).

[14] G. Parisi and Y.-S. Wu, Perturbation theory without gauge fixing, Sci. Sin. **24**, 483–496 (1981).

[15] J. R. Klauder, Stochastic quantization, in *Recent Developments in High-Energy Physics*, edited by H. Mitter and C. B. Lang (Springer Vienna, Vienna, 1983) pp. 251–281.

[16] M. Namiki, *Stochastic quantization*, Lecture Notes in Physics Monographs (Springer-Verlag Berlin Heidelberg, 1992).

[17] S. Furber, Large-scale neuromorphic computing systems, Journal of Neural Engineering **13**, 051001 (2016).

[18] C. S. Thakur, J. L. Molin, G. Cauwenberghs, G. Indiveri, K. Kumar, N. Qiao, J. Schemmel, R. Wang, E. Chicca, J. Olson Hasler, J. Seo, S. Yu, Y. Cao, A. van Schaik, and R. Etienne-Cummings, Large-scale neuromorphic spiking array processors: A quest to mimic the brain, Front. Neurosci. **12**, 891 (2018).

[19] K. Roy, A. Jaiswal, and P. Panda, Towards spike-based machine intelligence with neuromorphic computing, Nature **575**, 607–617 (2019).

[20] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, *et al.*, Loihi: A neuromorphic manycore processor with on-chip learning, IEEE Micro **38**, 82–99 (2018).

[21] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, *et al.*, A million spiking-neuron integrated circuit with a scalable communication network and interface, Science **345**, 668–673 (2014).

[22] J. Pei, L. Deng, S. Song, M. Zhao, Y. Zhang, S. Wu, G. Wang, Z. Zou, Z. Wu, W. He, *et al.*, Towards artificial general intelligence with hybrid Tianjic chip architecture, Nature **572**, 106–111 (2019).

[23] F. Cai, J. M. Correll, S. H. Lee, Y. Lim, V. Bothra, Z. Zhang, M. P. Flynn, and W. D. Lu, A fully integrated reprogrammable memristor–cmos system for efficient multiply–accumulate operations, Nature Electronics **2**, 290–299 (2019).

[24] I. Boybat, M. Le Gallo, S. Nandakumar, T. Moraitis, T. Parnell, T. Tuma, B. Rajendran, Y. Leblebici, A. Sebastian, and E. Eleftheriou, Neuromorphic computing with multi-memristive synapses, Nature communications **9**, 1–12 (2018).

[25] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs), IEEE transactions on biomedical circuits and systems **12**, 106–122 (2017).

[26] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, The Spinnaker project, Proceedings of the IEEE **102**, 652–665 (2014).

[27] M. A. Petrovici, S. Schmitt, J. Klähn, D. Stöckel, A. Schroeder, G. Bellec, J. Bill, O. Breitwieser, I. Bytschok, A. Grübl, *et al.*, Pattern representation and recognition with accelerated analog neuromorphic systems, in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)* (IEEE, 2017) pp. 1–4.

[28] S. Billaudelle, Y. Stradmann, K. Schreiber, B. Cramer, A. Baumbach, D. Dold, J. Göltz, A. F. Kungl, T. C. Wunderlich, A. Hartel, *et al.*, Versatile emulation of spiking neural networks on an accelerated neuromorphic substrate, in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)* (IEEE, 2020) pp. 1–5.

[29] W. Maass, Noise as a resource for computation and learning in networks of spiking neurons, Proceedings of the IEEE **102**, 860–880 (2014).

[30] E. Neftci, S. Das, B. Pedroni, K. Kreutz-Delgado, and G. Cauwenberghs, Event-driven contrastive divergence for spiking neuromorphic systems, Frontiers in Neuroscience **7**, 272 (2014).

[31] E. O. Neftci, B. U. Pedroni, S. Joshi, M. Al-Shedivat, and G. Cauwenberghs, Stochastic synapses enable efficient brain-inspired learning machines, Frontiers in Neuroscience **10**, 241 (2016).

[32] B. U. Pedroni, S. Das, E. Neftci, K. Kreutz-Delgado, and G. Cauwenberghs, Neuromorphic adaptations of restricted Boltzmann machines and deep belief networks, in *The 2013 International Joint Conference on Neural Networks (IJCNN)* (2013) pp. 1–6.

[33] L. Leng, R. Martel, O. Breitwieser, I. Bytschok, W. Senn, J. Schemmel, K. Meier, and M. A. Petrovici, Spiking neurons with short-term synaptic plasticity form superior generative networks, Scientific Reports **8**, 10651 (2018).

[34] G. E. Hinton, Training products of experts by minimizing contrastive divergence, Neural Comput. **14**, 1771–1800 (2002).

[35] 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**, 1–25 (2011).

[36] 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**, 1–30 (2013).

[37] S. Czischek, M. Gärttner, and T. Gasenzer, Quenches near ising quantum criticality as a challenge for artificial neural networks, Phys. Rev. B **98**, 024311 (2018).

[38] G. Carleo and M. Troyer, Solving the quantum many-body problem with artificial neural networks, Science **355**, 602–606 (2017).

[39] X. Gao and L.-M. Duan, Efficient representation of quantum many-body states with deep neural networks, Nature Communications **8**, 662 (2017).

[40] W. Severa, R. Lehoucq, O. Parekh, and J. B. Aimone, Spiking neural algorithms for markov process random walk (2018), arXiv:1805.00509 [cs.NE] .

[41] J. D. Smith, W. Severa, A. J. Hill, L. Reeder, B. Franke, R. B. Lehoucq, O. D. Parekh, and J. B. Aimone, Solving a steady-state pde using spiking networks and neuromorphic hardware, in *International Conference on Neuromorphic Systems 2020*, ICONS 2020 (Association for Computing Machinery, New York, NY, USA, 2020).

[42] J. Schemmel, J. Fieres, and K. Meier, Wafer-scale integration of analog neural networks, in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)* (2008) pp. 431–438.

[43] 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 *2010 IEEE International Symposium on Circuits and Systems (ISCAS)* (2010) pp. 1947–1950.

[44] D. Drubach, *The brain explained* (Prentice Hall Health, Upper Saddle River, N.J, 2000).

[45] W. S. McCulloch and W. Pitts, A logical calculus of the ideas immanent in nervous activity, The bulletin of mathematical biophysics **5**, 115–133 (1943).

[46] D. O. Hebb, *The organization of behavior: A neuropsychological theory* (Wiley, New York, 1949).

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

[48] P. J. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, Ph.D. thesis, Harvard University (1974).

[49] J. Schmidhuber, Deep learning in neural networks: An overview, Neural Networks **61**, 85–117 (2015).

[50] W. Maass, Networks of spiking neurons: The third generation of neural network models, Neural Networks **10**, 1659 – 1671 (1997).

[51] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity* (Cambridge University Press, 2002).

[52] Y. Huang and R. P. N. Rao, Neurons as Monte Carlo samplers: Bayesian inference and learning in spiking networks, in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14 (MIT Press, Cambridge, MA, USA, 2014) p. 1943–1951.

[53] D. Jimenez Rezende and W. Gerstner, Stochastic variational learning in recurrent spiking networks, Frontiers in Computational Neuroscience **8**, 38 (2014).

[54] V. Kornijcuk, H. Lim, J. Y. Seok, G. Kim, S. K. Kim, I. Kim, B. J. Choi, and D. S. Jeong, Leaky integrate-and-fire neuron circuit based on floating-gate integrator, Frontiers in Neuroscience **10**, 212 (2016).

[55] A. H. Marblestone, G. Wayne, and K. P. Kording, Toward an integration of deep learning and neuroscience, Frontiers in Computational Neuroscience **10**, 94 (2016).

[56] W. Gerstner, H. Sprekeler, and G. Deco, Theory and simulation in neuroscience, Science **338**, 60–65 (2012).

[57] W. Severa, O. Parekh, K. D. Carlson, C. D. James, and J. B. Aimone, Spiking network algorithms for scientific computing, in *2016 IEEE International Conference on Rebooting Computing (ICRC)* (2016) pp. 1–8.

[58] J. Kwisthout and N. Donselaar, On the computational power and complexity of spiking neural networks, in *Proceedings of the Neuro-Inspired Computational Elements Workshop*, NICE '20 (Association for Computing Machinery, New York, NY, USA, 2020).

[59] W. Gerstner, A. K. Kreiter, H. Markram, and A. V. M. Herz, Neural codes: Firing rates and beyond, Proceedings of the National Academy of Sciences **94**, 12740–12741 (1997).

[60] E. Fuchs and G. Flügge, Adult neuroplasticity: more than 40 years of research, Neural plasticity **2014**, 541870–541870 (2014), 24883212[pmid].

[61] M. Pfeiffer and T. Pfeil, Deep learning with spiking neurons: Opportunities and challenges, Frontiers in Neuroscience **12**, 774 (2018).

[62] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, Deep learning in spiking neural networks, Neural Networks **111**, 47–63 (2019).

[63] R. Brette, Philosophy of the spike: Rate-based vs. spike-based theories of the brain, Frontiers in Systems Neuroscience **9**, 151 (2015).

[64] M. Li and J. Z. Tsien, Neural code—neural self-information theory on how cell-assembly code rises from spike time and neuronal variability, Frontiers in Cellular Neuroscience **11**, 236 (2017).

[65] H. Wen, J. Shi, Y. Zhang, K.-H. Lu, J. Cao, and Z. Liu, Neural Encoding and Decoding with Deep Learning for Dynamic Natural Vision, Cerebral Cortex **28**, 4136–4160 (2017).

[66] A. Azarfar, N. Calcini, C. Huang, F. Zeldenrust, and T. Celikel, Neural coding: A single neuron's perspective, Neuroscience & Biobehavioral Reviews **94**, 238–247 (2018).

[67] A. Almomani, M. Alauthman, M. Alweshah, O. Dorgham, and F. Albalas, A comparative study on spiking neural network encoding schema: implemented with cloud computing, Cluster Computing **22**, 419–433 (2019).

[68] R. Brette, Is coding a relevant metaphor for the brain?, Behavioral and Brain Sciences **42**, e215 (2019).

[69] J. Göltz, A. Baumbach, S. Billaudelle, A. F. Kungl, O. Breitwieser, K. Meier, J. Schemmel, L. Kriener, and M. A. Petrovici, Fast and deep neuromorphic learning with first-spike coding, in *Proceedings of the Neuro-Inspired Computational Elements Workshop*, NICE '20 (Association for Computing Machinery, New York, NY, USA, 2020).

[70] W. Guo, M. E. Fouda, A. M. Eltawil, and K. N. Salama, Neural coding in spiking neural networks: A comparative study for robust neuromorphic systems, Frontiers in Neuroscience **15**, 212 (2021).

[71] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank, A survey of neuromorphic computing and neural networks in hardware, arXiv:1705.06963 .

[72] J. Zhu, T. Zhang, Y. Yang, and R. Huang, A comprehensive review on emerging artificial neuromorphic devices, Applied Physics Reviews **7**, 011312 (2020).

[73] D. V. Christensen, R. Dittmann, B. Linares-Barranco, A. Sebastian, M. L. Gallo, A. Redaelli, S. Slesazeck, T. Mikolajick, S. Spiga, S. Menzel, I. Valov, G. Milano, C. Ricciardi, S.-J. Liang, F. Miao, M. Lanza, T. J. Quill, S. T. Keene, A. Salleo, J. Grollier, D. Marković, A. Mizrahi, P. Yao, J. J. Yang, G. Indiveri, J. P. Strachan, S. Datta, E. Vianello, A. Valentian, J. Feldmann, X. Li, W. H. P. Pernice, H. Bhaskaran, E. Neftci, S. Ramaswamy, J. Tapson, F. Scherr, W. Maass, P. Panda, Y. Kim, G. Tanaka, S. Thorpe, C. Bartolozzi, T. A. Cleland, C. Posch, S.-C. Liu, A. N. Mazumder, M. Hosseini, T. Mohsenin, E. Donati, S. Tolu, R. Galeazzi, M. E. Christensen, S. Holm, D. Ielmini, and N. Pryds, 2021 roadmap on neuromorphic computing and engineering (2021), arXiv:2105.05956 [cs.ET] .

[74] H. Markram, The human brain project, Scientific American **306**, 50–55 (2012).

[75] D. Dold, I. Bytschok, A. F. Kungl, A. Baumbach, O. Breitwieser, W. Senn, J. Schemmel, K. Meier, and M. A. Petrovici, Stochasticity from function — Why the Bayesian brain may need no noise, Neural Networks **119**, 200 – 213 (2019).

[76] A. F. Kungl, S. Schmitt, J. Klähn, P. Müller, A. Baumbach, D. Dold, A. Kugele, E. Müller, C. Koke, M. Kleider, C. Mauch, O. Breitwieser, L. Leng, N. Gürtler, M. Güttler, D. Husmann, K. Husmann, A. Hartel, V. Karasenko, A. Grübl, J. Schemmel, K. Meier, and M. A. Petrovici, Accelerated physical emulation of bayesian inference in spiking neural networks, Front. Neurosci. **13**, 1201 (2019).

[77] M. A. Petrovici, J. Bill, I. Bytschok, J. Schemmel, and K. Meier, Stochastic inference with spiking neurons in the high-conductance state, Phys. Rev. E **94**, 042312 (2016).

[78] D. S. Lemons and A. Gythiel, Paul Langevin's 1908 paper "On the Theory of Brownian Motion" ["Sur la théorie du mouvement brownien," C. R. Acad. Sci. (Paris) 146, 530-533 (1908)], Am. J. Phys. **65**, 1079–1081 (1997).

[79] G. Aarts and I.-O. Stamatescu, Stochastic quantization at finite chemical potential, J. High Energ. Phys. **2008**, 018–018 (2008).

[80] G. Aarts, F. A. James, J. M. Pawlowski, E. Seiler, D. Sexty, and I.-O. Stamatescu, Stability of complex Langevin dynamics in effective models, J. High Energ. Phys. **2013**, 73 (2013).

[81] G. Aarts, Complex Langevin dynamics and other approaches at finite chemical potential, *Proceedings of the 30th International Symposium on Lattice Field Theory (Lattice 2012): Cairns, Australia, June 24-29, 2012*, PoS **LATTICE2012**, 017 (2012).

[82] M. Welling and Y. W. Teh, Bayesian learning via stochastic gradient langevin dynamics, in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11 (Omnipress, Madison, WI, USA, 2011) p. 681–688.

[83] J. Weis, P. Spilger, S. Billaudelle, Y. Stradmann, A. Emmel, E. Müller, O. Breitwieser, A. Grübl, J. Ilmberger, V. Karasenko, M. Kleider, C. Mauch, K. Schreiber, and J. Schemmel, Inference with artificial neural networks on analog neuromorphic hardware, in *IoT Streams for Data-Driven Predictive Maintenance and IoT, Edge, and Mobile for Embedded Machine Learning*, edited by J. Gama, S. Pashami, A. Bifet, M. Sayed-Mouchawe, H. Fröning, F. Pernkopf, G. Schiele, and M. Blott (Springer International Publishing, Cham, 2020) pp. 201–212.

[84] P. Spilger, E. Müller, A. Emmel, A. Leibfried, C. Mauch, C. Pehle, J. Weis, O. Breitwieser, S. Billaudelle, S. Schmitt, T. C. Wunderlich, Y. Stradmann, and J. Schemmel, hxtorch: Pytorch for brainscales-2 – perceptrons on analog neuromorphic hardware (2020), arXiv:2006.13138 [cs.NE] .

[85] R. P. Feynman, Simulating physics with computers, Int. J. Theor. Phys. **21**, 467–488 (1982).

[86] A. Peres, *Quantum Theory: Concepts and Methods* (Springer, Dordrecht, 2002).

[87] J. Carrasquilla, G. Torlai, R. G. Melko, and L. Aolita, Reconstructing quantum states with generative models, Nat. Mach. Intell. **1**, 155–161 (2019).

[88] J. Berges, S. Borsányi, D. Sexty, and I.-O. Stamatescu, Lattice simulations of real-time quantum fields, Phys. Rev. D **75**, 045007 (2007).

[89] G. Cohen, E. Gull, D. R. Reichman, and A. J. Millis, Taming the dynamical sign problem in real-time evolution of quantum many-body problems, Phys. Rev. Lett. **115**, 266802 (2015).

[90] A. Alexandru, G. m. c. Başar, P. F. Bedaque, S. Vartak, and N. C. Warrington, Monte Carlo study of real time dynamics on the lattice, Phys. Rev. Lett. **117**, 081602 (2016).

[91] G. Kanwar and M. L. Wagman, Real-time lattice gauge theory actions: unitarity, convergence, and path integral contour deformations (2021), arXiv:2103.02602 [hep-lat] .

[92] E. Y. Loh, J. E. Gubernatis, R. T. Scalettar, S. R. White, D. J. Scalapino, and R. L. Sugar, Sign problem in the numerical simulation of many-electron systems, Phys. Rev. B **41**, 9301–9307 (1990).

[93] D. J. Scalapino, Numerical studies of the 2D Hubbard model, in *Handbook of High-Temperature Superconductivity: Theory and Experiment*, edited by J. R. Schrieffer and J. S. Brooks (Springer New York, New York, NY, 2007) pp. 495–526.

[94] J. P. F. LeBlanc, A. E. Antipov, F. Becca, I. W. Bulik, G. K.-L. Chan, C.-M. Chung, Y. Deng, M. Ferrero, T. M. Henderson, C. A. Jiménez-Hoyos, E. Kozik, X.-W. Liu, A. J. Millis, N. V. Prokof'ev, M. Qin, G. E. Scuseria, H. Shi, B. V. Svistunov, L. F. Tocchio, I. S. Tupitsyn, S. R. White, S. Zhang, B.-X. Zheng, Z. Zhu, and E. Gull (Simons Collaboration on the Many-Electron Problem), Solutions of the two-dimensional Hubbard model: benchmarks and results from a wide range of numerical algorithms, Phys. Rev. X **5**, 041041 (2015).

[95] M. Ulybyshev, C. Winterowd, and S. Zafeiropoulos, Taming the sign problem of the finite density Hubbard model via Lefschetz thimbles (2019), arXiv:1906.02726 [cond-mat.str-el] .

[96] M. V. Ulybyshev, V. I. Dorozhinskii, and O. V. Pavlovskii, The use of neural networks to solve the sign problem in physical models, Phys. Part. Nuclei **51**, 363–379 (2020).

[97] C. Berger, L. Rammelmüller, A. Loheac, F. Ehmann, J. Braun, and J. Drut, Complex Langevin and other approaches to the sign problem in quantum many-body physics, Phys. Rept. **892**, 1–54 (2021).

[98] J. Braun, J.-W. Chen, J. Deng, J. E. Drut, B. Friman, C.-T. Ma, and Y.-D. Tsai, Imaginary polarization as a way to surmount the sign problem in ab initio calculations of spin-imbalanced Fermi gases, Phys. Rev. Lett. **110**, 130404 (2013).

[99] K. Gubbels and H. Stoof, Imbalanced Fermi gases at unitarity, Phys. Rept. **525**, 255–313 (2013).

[100] L. Rammelmüller, W. J. Porter, J. E. Drut, and J. Braun, Surmounting the sign problem in nonrelativistic calculations: A case study with mass-imbalanced fermions, Phys. Rev. D **96**, 094506 (2017).

[101] A. Alexandru, P. F. Bedaque, and N. C. Warrington, Spin polarized nonrelativistic fermions in $1+1$ dimensions, Phys. Rev. D **98**, 054514 (2018).

[102] L. Rammelmüller, J. E. Drut, and J. Braun, Pairing patterns in one-dimensional spin- and mass-imbalanced Fermi gases, SciPost Phys. **9**, 14 (2020).

[103] A. H. Castro Neto, F. Guinea, N. M. R. Peres, K. S. Novoselov, and A. K. Geim, The electronic properties of graphene, Rev. Mod. Phys. **81**, 109–162 (2009).

[104] M. V. Ulybyshev, P. V. Buividovich, M. I. Katsnelson, and M. I. Polikarpov, Monte Carlo study of the semimetal-insulator phase transition in monolayer graphene with a realistic interelectron interaction potential, Phys. Rev. Lett. **111**, 056801 (2013).

[105] D. Smith and L. von Smekal, Monte Carlo simulation of the tight-binding model of graphene with partially screened Coulomb interactions, Phys. Rev. B **89**, 195429 (2014).

[106] A. Hasenfratz and D. Toussaint, Canonical ensembles and nonzero density quantum chromodynamics, Nucl. Phys. B **371**, 539–549 (1992).

[107] S. Muroya, A. Nakamura, C. Nonaka, and T. Takaishi, Lattice QCD at finite density: an introductory review, Prog. Theor. Phys. **110**, 615–668 (2003).

[108] M. Stephanov, QCD phase diagram: an overview, PoS **LAT2006**, 024 (2006).

[109] P. de Forcrand, Simulating QCD at finite density, PoS **LAT2009**, 010 (2010).

[110] E. Seiler, D. Sexty, and I.-O. Stamatescu, Gauge cooling in complex Langevin for lattice QCD with heavy quarks, Phys. Lett. B **723**, 213–216 (2013).

[111] D. Sexty, Simulating full QCD at nonzero density using the complex Langevin equation, Phys. Lett. B **729**, 108 – 111 (2014).

[112] Y. Mori, K. Kashiwa, and A. Ohnishi, Application of a neural network to the sign problem via the path optimization method, Prog. Theor. Exp. Phys. **2018**, 023B04 (2018).

[113] A. Joseph and A. Kumar, Complex Langevin simulations of zero-dimensional supersymmetric quantum field theories, Phys. Rev. D **100**, 074507 (2019).

[114] K. Kashiwa, Y. Mori, and A. Ohnishi, Controlling the model sign problem via the path optimization method: Monte Carlo approach to a QCD effective model with Polyakov loop, Phys. Rev. D **99**, 014033 (2019).

[115] A. Alexandru, G. Basar, P. F. Bedaque, and N. C. Warrington, Complex paths around the sign problem, (2020), arXiv:2007.05436 [hep-lat] .

[116] F. Attanasio, B. Jäger, and F. P. G. Ziegler, Complex Langevin simulations and the QCD phase diagram: recent developments, Eur. Phys. J. A **56**, 251 (2020).

[117] G. Parisi, On complex probabilities, Phys. Lett. B **131**, 393–395 (1983).

[118] M. Troyer and U.-J. Wiese, Computational complexity and fundamental limitations to fermionic quantum Monte Carlo simulations, Phys. Rev. Lett. **94**, 170201 (2005).

[119] J. Ambjørn and S.-K. Yang, Numerical problems in applying the Langevin equation to complex effective actions, Phys. Lett. B **165**, 140–146 (1985).

[120] G. Aarts, F. A. James, E. Seiler, and I.-O. Stamatescu, Adaptive stepsize and instabilities in complex Langevin dynamics, Phys. Lett. B **687**, 154 – 159 (2010).

[121] F. Attanasio and B. Jäger, Dynamical stabilisation of complex Langevin simulations of QCD, Eur. Phys. J. C **79**, 16 (2019).

[122] E. Seiler, Status of complex Langevin, EPJ Web Conf. **175**, 01019 (2018).

[123] D. Guest, K. Cranmer, and D. Whiteson, Deep Learning and its Application to LHC Physics, Ann. Rev. Nucl. Part. Sci. **68**, 161–181 (2018), arXiv:1806.11484 [hep-ex] .

[124] A. Radovic, M. Williams, D. Rousseau, M. Kagan, D. Bonacorsi, A. Himmel, A. Aurisano, K. Terao, and T. Wongjirad, Machine learning at the energy and intensity frontiers of particle physics, Nature **560**, 41–48 (2018).

[125] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, Nature **521**, 436–444 (2015).

[126] A. Graves, A.-r. Mohamed, and G. Hinton, Speech recognition with deep recurrent neural networks, in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (2013) pp. 6645–6649.

[127] A. Krizhevsky, I. Sutskever, and G. E. Hinton, Imagenet classification with deep convolutional neural networks, in *Advances in Neural Information Processing Systems*, Vol. 25, edited by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Curran Associates, Inc., 2012).

[128] S. Ren, K. He, R. Girshick, and J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, in *Advances in Neural Information Processing Systems*, Vol. 28, edited by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Curran Associates, Inc., 2015).

[129] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016) pp. 770–778.

[130] L. Wang, Discovering phase transitions with unsupervised learning, Phys. Rev. B **94**, 195105 (2016).

[131] P. Broecker, J. Carrasquilla, R. G. Melko, and S. Trebst, Machine learning quantum phases of matter beyond the fermion sign problem, Scientific Reports **7**, 8823 (2017).

[132] L.-G. Pang, K. Zhou, N. Su, H. Petersen, H. Stöcker, and X.-N. Wang, An equation-of-state-meter of qcd transition from deep learning, (2016), arXiv:1612.04262 [hep-ph] .

[133] S. J. Wetzel, Unsupervised learning of phase transitions: From principal component analysis to variational autoencoders, Phys. Rev. E **96**, 022140 (2017).

[134] S. J. Wetzel and M. Scherzer, Machine learning of explicit order parameters: From the ising model to su(2) lattice gauge theory, Phys. Rev. B **96**, 184410 (2017).

[135] M. Cristoforetti, G. Jurman, A. I. Nardelli, and C. Furlanello, Towards meaningful physics from generative models, (2017), arXiv:1705.09524 [hep-lat] .

[136] K. Ch'ng, J. Carrasquilla, R. G. Melko, and E. Khatami, Machine learning phases of strongly correlated fermions, Phys. Rev. X **7**, 031038 (2017).

[137] W. Hu, R. R. P. Singh, and R. T. Scalettar, Discovering phases, phase transitions, and crossovers through unsupervised machine learning: A critical examination, Phys. Rev. E **95**, 062122 (2017).

[138] Z. Liu, S. P. Rodrigues, and W. Cai, Simulating the ising model with a deep convolutional generative adversarial network, (2017), arXiv:1710.04987 [cond-mat.dis-nn] .

[139] E. P. L. van Nieuwenburg, Y.-H. Liu, and S. D. Huber, Learning phase transitions by confusion, Nature Physics **13**, 435–439 (2017).

[140] P. Ponte and R. G. Melko, Kernel methods for interpretable machine learning of order parameters, Phys. Rev. B **96**, 205146 (2017).

[141] J. Carrasquilla and R. G. Melko, Machine learning phases of matter, Nature Physics **13**, 431–434 (2017).

[142] A. Morningstar and R. G. Melko, Deep learning the ising model near criticality, J. Mach. Learn. Res. **18**, 5975–5991 (2017).

[143] A. Tanaka and A. Tomiya, Towards reduction of autocorrelation in HMC by machine learning, (2017), arXiv:1712.03893 [hep-lat] .

[144] A. Tanaka and A. Tomiya, Detection of phase transition via convolutional neural networks, Journal of the Physical Society of Japan **86**, 063001 (2017).

[145] L. Huang and L. Wang, Accelerated monte carlo simulations with restricted boltzmann machines, Phys. Rev. B **95**, 035105 (2017).

[146] J. Liu, Y. Qi, Z. Y. Meng, and L. Fu, Self-learning monte carlo method, Phys. Rev. B **95**, 041101 (2017).

[147] D. Wu, L. Wang, and P. Zhang, Solving statistical mechanics using variational autoregressive networks, Phys. Rev. Lett. **122**, 080602 (2019).

[148] K. Zhou, G. Endrődi, L.-G. Pang, and H. Stöcker, Regressive and generative neural networks for scalar field theory, Phys. Rev. D **100**, 011501 (2019).

[149] P. E. Shanahan, D. Trewartha, and W. Detmold, Machine learning action parameters in lattice quantum chromodynamics, Phys. Rev. D **97**, 094506 (2018).

[150] P. Suchsland and S. Wessel, Parameter diagnostics of phases and phase transition learning by neural networks, Phys. Rev. B **97**, 174435 (2018).

[151] J. M. Pawlowski and J. M. Urban, Reducing autocorrelation times in lattice simulations with generative adversarial networks, Machine Learning: Science and Technology **1**, 045011 (2020).

[152] J. Karpie, K. Orginos, A. Rothkopf, and S. Zafeiropoulos, Reconstructing parton distribution functions from ioffe time data: from bayesian methods to neural networks, Journal of High Energy Physics **2019**, 57 (2019).

[153] F. Noé, S. Olsson, J. Köhler, and H. Wu, Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning, Science **365**, eaaw1147 (2019).

[154] K. Nicoli, P. Kessel, N. Strodthoff, W. Samek, K.-R. Müller, and S. Nakajima, Comment on "Solving Statistical Mechanics Using VANs": Introducing saVANt - VANs Enhanced by Importance and MCMC Sampling, (2019), arXiv:1903.11048 [cond-mat.stat-mech] .

[155] M. S. Albergo, G. Kanwar, and P. E. Shanahan, Flow-based generative models for markov chain monte carlo in lattice field theory, Phys. Rev. D **100**, 034515 (2019).

[156] K. Kashiwa, Y. Kikuchi, and A. Tomiya, Phase transition encoded in neural network, Progress of Theoretical and Experimental Physics **2019**, 10.1093/ptep/ptz082 (2019), 083A04.

[157] K. Liu, J. Greitemann, and L. Pollet, Learning multiple order parameters with interpretable machines, Phys. Rev. B **99**, 104410 (2019).

[158] C. Casert, T. Vieijra, J. Nys, and J. Ryckebusch, Interpretable machine learning for inferring the phase boundaries in a nonequilibrium system, Phys. Rev. E **99**, 023304 (2019).

[159] W. Rządkowski, N. Defenu, S. Chiacchiera, A. Trombettoni, and G. Bighin, Detecting composite orders in layered models via machine learning, New Journal of Physics **22**, 093026 (2020).

[160] K. A. Nicoli, S. Nakajima, N. Strodthoff, W. Samek, K.-R. Müller, and P. Kessel, Asymptotically unbiased estimation of physical observables with neural samplers, Phys. Rev. E **101**, 023304 (2020).

[161] E. Greplova, A. Valenti, G. Boschung, F. Schäfer, N. Lörch, and S. D. Huber, Unsupervised identification of topological phase transitions using predictive models, New Journal of Physics **22**, 045003 (2020).

[162] J. Shlomi, P. Battaglia, and J.-R. Vlimant, Graph neural networks in particle physics, Machine Learning: Science and Technology **2**, 021001 (2021).

[163] P. Mehta, M. Bukov, C.-H. Wang, A. G. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, A high-bias, low-variance introduction to machine learning for physicists, Phys. Rept. **810**, 1–124 (2019).

[164] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, Machine learning and the physical sciences, Rev. Mod. Phys. **91**, 045002 (2019).

[165] Y. Bengio, A. Courville, and P. Vincent, Representation learning: A review and new perspectives, IEEE Transactions on Pattern Analysis and Machine Intelligence **35**, 1798–1828 (2013).

[166] N. Tishby, F. C. Pereira, and W. Bialek, The information bottleneck method, in *Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing* (1999) pp. 368–377.

[167] T.-W. Lee, M. Girolami, A. Bell, and T. Sejnowski, A unifying information-theoretic framework for independent component analysis, Computers & Mathematics with Applications **39**, 1–21 (2000).

[168] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, Information-theoretic metric learning, in *Proceedings of the 24th International Conference on Machine Learning*, ICML '07 (Association for Computing Machinery, New York, NY, USA, 2007) p. 209–216.

[169] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, Extracting and composing robust features with denoising autoencoders, in *Proceedings of the 25th International Conference on Machine Learning*, ICML '08 (Association for Computing Machinery, New York, NY, USA, 2008) p. 1096–1103.

[170] D. P. Kingma and M. Welling, Auto-encoding variational bayes, in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, edited by Y. Bengio and Y. LeCun (2014).

[171] N. Tishby and N. Zaslavsky, Deep learning and the information bottleneck principle, in *2015 IEEE Information Theory Workshop (ITW)* (2015) pp. 1–5.

[172] M. D. Hoffmann and M. J. Johnson, Elbo surgery: yet another way to carve up the variational evidence lower bound, in *NIPS 2016 Workshop on Advances in Approximate Bayesian Inference* (2016).

[173] A. A. Alemi, B. Poole, I. Fischer, J. V. Dillon, R. A. Saurous, and K. Murphy, An information-theoretic analysis of deep latent-variable models, CoRR **abs/1711.00464** (2017), arXiv:1711.00464 .

[174] A. Alemi, B. Poole, I. Fischer, J. Dillon, R. A. Saurous, and K. Murphy, Fixing a broken ELBO, in *Proceedings of the 35th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 80, edited by J. Dy and A. Krause (PMLR, 2018) pp. 159–168.

[175] R. Shwartz-Ziv and N. Tishby, Opening the black box of deep neural networks via information, CoRR **abs/1703.00810** (2017), arXiv:1703.00810 .

[176] M. Gabrié, A. Manoel, C. Luneau, J. Barbier, N. Macris, F. Krzakala, and L. Zdeborová, Entropy and mutual information in models of deep neural networks, in *NeurIPS* (2018) pp. 1826–1836.

[177] A. M. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. D. Tracey, and D. D. Cox, On the information bottleneck theory of deep learning, Journal of Statistical Mechanics: Theory and Experiment **2019**, 124020 (2019).

[178] A. Kolchinsky, B. D. Tracey, and D. H. Wolpert, Nonlinear information bottleneck, Entropy **21**, 10.3390/e21121181 (2019).

[179] V. Crescimanna and B. Graham, An information theoretic approach to the autoencoder, in *Recent Advances in Big Data and Deep Learning*, edited by L. Oneto, N. Navarin, A. Sperduti, and D. Anguita (Springer International Publishing, Cham, 2020) pp. 99–108.

[180] L. Ardizzone, R. Mackowiak, C. Rother, and U. Köthe, Training normalizing flows with the information bottleneck for competitive generative classification, in *Advances in Neural Information Processing Systems*, Vol. 33, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Curran Associates, Inc., 2020) pp. 7828–7840.

[181] R. Linsker, Self-organization in a perceptual network, Computer **21**, 105–117 (1988).

[182] A. J. Bell and T. J. Sejnowski, An Information-Maximization Approach to Blind Separation and Blind Deconvolution, Neural Computation **7**, 1129–1159 (1995).

[183] D. Barber and F. Agakov, Information maximization in noisy channels : A variational approach, in *Advances in Neural Information Processing Systems*, Vol. 16, edited by S. Thrun, L. Saul, and B. Schölkopf (MIT Press, 2004).

[184] L. Paninski, Estimation of entropy and mutual information, Neural Comput. **15**, 1191–1253 (2003).

[185] A. Kraskov, H. Stögbauer, and P. Grassberger, Estimating mutual information, Phys. Rev. E **69**, 066138 (2004).

[186] M. I. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, Y. Bengio, A. Courville, and D. Hjelm, Mutual information neural estimation, in *Proceedings of the 35th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 80, edited by J. Dy and A. Krause (PMLR, 2018) pp. 531–540.

[187] Z. Goldfeld, E. Van Den Berg, K. Greenewald, I. Melnyk, N. Nguyen, B. Kingsbury, and Y. Polyanskiy, Estimating information flow in deep neural networks, in *Proceedings of the 36th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 97, edited by K. Chaudhuri and R. Salakhutdinov (PMLR, 2019) pp. 2299–2308.

[188] D. McAllester and K. Stratos, Formal limitations on the measurement of mutual information, in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, Vol. 108, edited by S. Chiappa and R. Calandra (PMLR, 2020) pp. 875–884.

[189] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, Learning deep representations by mutual information estimation and maximization, in *International Conference on Learning Representations* (2019).

[190] P. Bachman, R. D. Hjelm, and W. Buchwalter, Learning representations by maximizing mutual information across views, in *Advances in Neural Information Processing Systems*, Vol. 32, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc., 2019).

[191] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Generative adversarial nets, in *Advances in Neural Information Processing Systems*, Vol. 27, edited by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger (Curran Associates, Inc., 2014).

[192] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, Infogan: Interpretable representation learning by information maximizing generative adversarial nets, in *Advances in Neural Information Processing Systems*, Vol. 29, edited by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Curran Associates, Inc., 2016).

[193] D. J. Rezende, S. Mohamed, and D. Wierstra, Stochastic backpropagation and approximate inference in deep generative models, in *Proceedings of the 31st International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 32, edited by E. P. Xing and T. Jebara (PMLR, Bejing, China, 2014) pp. 1278–1286.

[194] D. Qian and W. K. Cheung, Enhancing variational autoencoders with mutual information neural estimation for text generation, in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (Association for Computational Linguistics, Hong Kong, China, 2019) pp. 4047–4057.

[195] S. Zhao, J. Song, and S. Ermon, Infovae: Balancing learning and inference in variational autoencoders, Proceedings of the AAAI Conference on Artificial Intelligence **33**, 5885–5892 (2019).

[196] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, Deep graph infomax, in *International Conference on Learning Representations* (2019).

[197] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, Neural message passing for quantum chemistry, in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17 (JMLR.org, 2017) p. 1263–1272.

[198] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, Geometric deep learning: Going beyond euclidean data, IEEE Signal Processing Magazine **34**, 18–42 (2017).

[199] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, Graph neural networks: A review of methods and applications (2018), arXiv:1812.08434 [cs.LG] .

[200] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, Relational inductive biases, deep learning, and graph networks (2018), arXiv:1806.01261 [cs.LG] .

[201] H. Cai, V. W. Zheng, and K. Chang, A comprehensive survey of graph embedding: Problems, techniques, and applications, IEEE Transactions on Knowledge & Data Engineering **30**, 1616–1637 (2018).

[202] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, A comprehensive survey on graph neural networks, IEEE Transactions on Neural Networks and Learning Systems , 1–21 (2020).

[203] I. Chami, S. Abu-El-Haija, B. Perozzi, C. Ré, and K. Murphy, Machine learning on graphs: A model and comprehensive taxonomy (2020), arXiv:2005.03675 [cs.LG] .

[204] G. Ma, N. K. Ahmed, T. L. Willke, and P. S. Yu, Deep graph similarity learning: a survey, Data Mining and Knowledge Discovery **35**, 688–725 (2021).

[205] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, Explaining explanations: An overview of interpretability of machine learning, in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)* (2018) pp. 80–89.

[206] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai, Information Fusion **58**, 82–115 (2020).

[207] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, Explainable ai: A review of machine learning interpretability methods, Entropy **23**, 10.3390/e23010018 (2021).

[208] M. Jarrell and J. E. Gubernatis, Bayesian inference and the analytic continuation of imaginary-time quantum Monte Carlo data, Phys. Rept. **269**, 133–195 (1996).

[209] M. Asakawa, Y. Nakahara, and T. Hatsuda, Maximum entropy analysis of the spectral functions in lattice qcd, Progress in Particle and Nuclear Physics **46**, 459–508 (2001).

[210] Y. Burnier and A. Rothkopf, Bayesian approach to spectral function reconstruction for euclidean quantum field theories, Phys. Rev. Lett. **111**, 182003 (2013).

[211] A. Rothkopf, Bayesian techniques and applications to QCD, in *Proceedings of XIII Quark Confinement and the Hadron Spectrum — PoS(Confinement2018)*, Vol. 336 (2019) p. 026.

[212] V. Shah and C. Hegde, Solving linear inverse problems using gan priors: An algorithm with provable guarantees, in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018) pp. 4609–4613.

[213] H. Li, J. Schwab, S. Antholzer, and M. Haltmeier, NETT: solving inverse problems with deep neural networks, Inverse Problems **36**, 065005 (2020).

[214] R. Anirudh, J. J. Thiagarajan, B. Kailkhura, and T. Bremer, An unsupervised approach to solving inverse problems using generative adversarial networks, CoRR **abs/1805.07281** (2018).

[215] L. Ardizzone, J. Kruse, C. Rother, and U. Köthe, Analyzing inverse problems with invertible neural networks, in *International Conference on Learning Representations* (2019).

[216] G. Aarts, P. Giudice, and E. Seiler, Localised distributions and criteria for correctness in complex Langevin dynamics, Ann. Phys. (N. Y.) **337**, 238–260 (2013).

[217] K. Nagata, J. Nishimura, and S. Shimasaki, Argument for justification of the complex Langevin method and the condition for correct convergence, Phys. Rev. D **94**, 114515 (2016).

[218] G. Aarts, Can stochastic quantization evade the sign problem? The relativistic Bose gas at finite chemical potential, Phys. Rev. Lett. **102**, 131601 (2009).

[219] G. Aarts, E. Seiler, and I.-O. Stamatescu, Complex Langevin method: when can it be trusted?, Phys. Rev. D **81**, 054508 (2010).

[220] L. L. Salcedo, Does the complex Langevin method give unbiased results?, Phys. Rev. D **94**, 114505 (2016).

[221] G. Aarts, F. A. James, E. Seiler, and I.-O. Stamatescu, Complex Langevin: etiology and diagnostics of its main problem, Eur. Phys. J. C **71**, 1756 (2011).

[222] G. Aarts, K. Boguslavski, M. Scherzer, E. Seiler, D. Sexty, and I.-O. Stamatescu, Getting even with CLE, EPJ Web Conf. **175**, 14007 (2018).

[223] K. Nagata, J. Nishimura, and S. Shimasaki, Testing the criterion for correct convergence in the complex Langevin method, J. High Energ. Phys. **2018**, 4 (2018).

[224] M. Scherzer, E. Seiler, D. Sexty, and I.-O. Stamatescu, Complex Langevin and boundary terms, Phys. Rev. D **99**, 014512 (2019).

[225] E. Seiler, Complex Langevin: boundary terms at poles, Phys. Rev. D **102**, 094507 (2020).

[226] G. Aarts, E. Seiler, D. Sexty, and I.-O. Stamatescu, Complex Langevin dynamics and zeroes of the fermion determinant, J. High Energ. Phys. **2017**, 44 (2017).

[227] M. Scherzer, E. Seiler, D. Sexty, and I.-O. Stamatescu, Controlling complex Langevin simulations of lattice models by boundary term analysis, Phys. Rev. D **101**, 014501 (2020).

[228] J. Nishimura and S. Shimasaki, New insights into the problem with a singular drift term in the complex Langevin method, Phys. Rev. D **92**, 011501 (2015).

[229] M. E. J. Newman and G. T. Barkema, *Monte Carlo methods in statistical physics* (Clarendon Press, Oxford, 1999).

[230] C. F. Baillie and D. A. Johnston, Metropolis and langevin time, Phys. Rev. D **39**, 1246–1248 (1989).

[231] P. Meakin, H. Metiu, R. G. Petschek, and D. J. Scalapino, The simulation of spinodal decomposition in two dimensions: A comparison of Monte Carlo and Langevin dynamics, J. Chem. Phys. **79**, 1948–1954 (1983).

[232] R. Ettelaie and M. A. Moore, Comparison of langevin and monte carlo dynamics, Journal of Physics A: Mathematical and General **17**, 3505 (1984).

[233] A. Destexhe, M. Rudolph, and D. Paré, The high-conductance state of neocortical neurons in vivo, Nature Reviews Neuroscience **4**, 739–751 (2003).

[234] M. A. Petrovici, *Form versus function: Theory and models for neuronal substrates* (Springer International Publishing, Cham, 2016).

[235] G. Hinton, P. Dayan, B. Frey, and R. Neal, The "wake-sleep" algorithm for unsupervised neural networks, Science **268**, 1158–1161 (1995).

[236] N. Gürtler, *A Markovian Model of LIF Networks*, Masterarbeit, Universität Heidelberg (2018).

[237] M. A. Petrovici, A. Schroeder, O. Breitwieser, A. Grübl, J. Schemmel, and K. Meier, Robustness from structure: Inference with hierarchical spiking networks on analog neuromorphic hardware, in *2017 International Joint Conference on Neural Networks (IJCNN)* (2017) pp. 2209–2216.

[238] G. E. Hinton and R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science **313**, 504–507 (2006).

[239] R. Salakhutdinov, A. Mnih, and G. Hinton, Restricted boltzmann machines for collaborative filtering, in *Proceedings of the 24th International Conference on Machine Learning*, ICML '07 (Association for Computing Machinery, New York, NY, USA, 2007) p. 791–798.

[240] N. Le Roux and Y. Bengio, Representational Power of Restricted Boltzmann Machines and Deep Belief Networks, Neural Computation **20**, 1631–1649 (2008).

[241] G. Torlai and R. G. Melko, Learning thermodynamics with boltzmann machines, Phys. Rev. B **94**, 165134 (2016).

[242] R. G. Melko, G. Carleo, J. Carrasquilla, and J. I. Cirac, Restricted boltzmann machines in quantum physics, Nature Physics **15**, 887–892 (2019).

[243] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, A learning algorithm for Boltzmann machines, Cogn. Sci. **9**, 147–169 (1985).

[244] A. Fischer and C. Igel, An introduction to restricted boltzmann machines, in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, edited by L. Alvarez, M. Mejail, L. Gomez, and J. Jacobo (Springer Berlin Heidelberg, Berlin, Heidelberg, 2012) pp. 14–36.

[245] A. Baumbach, *Magnetic Phenomena in Spiking Neural Networks*, Masterarbeit, Universität Heidelberg (2016).

[246] B. Cramer, S. Billaudelle, S. Kanya, A. Leibfried, A. Grübl, V. Karasenko, C. Pehle, K. Schreiber, Y. Stradmann, J. Weis, J. Schemmel, and F. Zenke, Surrogate gradients for analog neuromorphic computing (2021), arXiv:2006.07239 [cs.NE] .

[247] T. C. Wunderlich and C. Pehle, Event-based backpropagation can compute exact gradients for spiking neural networks, Scientific Reports **11**, 12829 (2021).

[248] M. Vuffray, S. Misra, and A. Lokhov, Efficient learning of discrete graphical models, in *Advances in Neural Information Processing Systems*, Vol. 33, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Curran Associates, Inc., 2020) pp. 13575–13585.

[249] A. Jayakumar, A. Lokhov, S. Misra, and M. Vuffray, Learning of discrete graphical models with neural networks, in *Advances in Neural Information Processing Systems*, Vol. 33, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Curran Associates, Inc., 2020) pp. 5610–5620.

[250] E. O. Neftci, H. Mostafa, and F. Zenke, Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks, IEEE Signal Processing Magazine **36**, 51–63 (2019).

[251] S. Billaudelle, B. Cramer, M. A. Petrovici, K. Schreiber, D. Kappel, J. Schemmel, and K. Meier, Structural plasticity on an accelerated analog neuromorphic hardware system, Neural Networks **133**, 11–20 (2021).

[252] T. Wunderlich, A. F. Kungl, E. Müller, A. Hartel, Y. Stradmann, S. A. Aamir, A. Grübl, A. Heimbrecht, K. Schreiber, D. Stöckel, C. Pehle, S. Billaudelle, G. Kiene, C. Mauch, J. Schemmel, K. Meier, and M. A. Petrovici, Demonstrating advantages of neuromorphic computation: A pilot study, Front. Neurosci. **13**, 260 (2019).

[253] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, Equation of State Calculations by Fast Computing Machines, J. Chem. Phys. **21**, 1087–1092 (1953).

[254] R. B. Potts, *The mathematical investigation of some cooperative phenomena*, Ph.D. thesis (1951).

[255] R. B. Potts, Some generalized order - disorder transformations, Proc. Cambridge Phil. Soc. **48**, 106–109 (1952).

[256] F. Y. Wu, The Potts model, Rev. Mod. Phys. **54**, 235–268 (1982).

[257] C. M. Lapilli, P. Pfeifer, and C. Wexler, Universality away from critical points in two-dimensional phase transitions, Phys. Rev. Lett. **96**, 140603 (2006).

[258] P. S. Neelakanta, R. Sudhakar, and D. Degroff, Langevin machine: A neural network based on stochastically justifiable sigmoidal function, Biol. Cybern. **65**, 331–338 (1991).

[259] A. Baumbach, *From microscopic dynamics to ensemble behavior in spiking neural networks*, Ph.D. thesis, Universität Heidelberg (2021).

[260] 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**, 1–22 (2011).

[261] 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**, 13 (2015).

[262] J. Jordan, M. A. Petrovici, O. Breitwieser, J. Schemmel, K. Meier, M. Diesmann, and T. Tetzlaff, Deterministic networks for probabilistic computing, Scientific Reports **9**, 18303 (2019).

[263] E. Ising, Beitrag zur Theorie des Ferromagnetismus, Zeitschrift für Physik **31**, 253–258 (1925).

[264] L. Onsager, Crystal statistics. 1. A Two-dimensional model with an order disorder transition, Phys. Rev. **65**, 117–149 (1944).

[265] I. Bytschok, D. Dold, J. Schemmel, K. Meier, and M. A. Petrovici, Spike-based probabilistic inference with correlated noise, in *BMC Neuroscience 2017*, Vol. 18 (Organization for Computational Neurosciences, 2017) p. 200.

[266] G. Montúfar, Restricted Boltzmann machines: introduction and review, in *Information Geometry and Its Applications*, edited by N. Ay, P. Gibilisco, and F. Matúš (Springer International Publishing, Cham, 2018) pp. 75–115.

[267] S. Kullback and R. A. Leibler, On information and sufficiency, Ann. Math. Statist. **22**, 79–86 (1951).

[268] Y. Hamada, Dynamics of the noise-induced phase transition of the verhulst model, Progress of Theoretical Physics **65**, 850–860 (1981).

[269] A. L. Barbera and B. Spagnolo, Spatio-temporal patterns in population dynamics, Physica A: Statistical Mechanics and its Applications **314**, 120 – 124 (2002).

[270] D. Valenti, A. Fiasconaro, and B. Spagnolo, Pattern formation and spatial correlation induced by the noise in two competing species, Acta Physica Polonica B **35**, 1481 – 1489 (2004).

[271] A. Fiasconaro, D. Valenti, and B. Spagnolo, Nonmonotonic behavior of spatiotemporal pattern formation in a noisy Lotka-Volterra system, Acta Physica Polonica B **35**, 1491 – 1500 (2004).

[272] K. Kaneko, Overview of coupled map lattices, Chaos: An Interdisciplinary Journal of Nonlinear Science **2**, 279–282 (1992).

[273] M. Girardi-Schappo, M. Tragtenberg, and O. Kinouchi, A brief history of excitable map-based neurons and neural networks, Journal of Neuroscience Methods **220**, 116 – 130 (2013).

[274] R. FitzHugh, Mathematical models of threshold phenomena in the nerve membrane, The bulletin of mathematical biophysics **17**, 257 (1955).

[275] J. Nagumo, S. Arimoto, and S. Yoshizawa, An active pulse transmission line simulating nerve axon, Proceedings of the IRE **50**, 2061 (1962).

[276] M. E. Yamakou, T. D. Tran, L. H. Duc, and J. Jost, The stochastic Fitzhugh-Nagumo neuron model in the excitable regime embeds a leaky integrate-and-fire model, Journal of mathematical biology **79**, 509 (2019).

[277] M. A. Zaks, X. Sailer, L. Schimansky-Geier, and A. B. Neiman, Noise induced complexity: From subthreshold oscillations to spiking in coupled excitable systems, Chaos: An Interdisciplinary Journal of Nonlinear Science **15**, 026117 (2005).

[278] K.-E. Lee, M. A. Lopes, J. F. F. Mendes, and A. V. Goltsev, Critical phenomena and noise-induced phase transitions in neuronal networks, Phys. Rev. E **89**, 012701 (2014).

[279] K. Hornik, M. Stinchcombe, and H. White, Multilayer feedforward networks are universal approximators, Neural Networks **2**, 359–366 (1989).

[280] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Networks **4**, 251–257 (1991).

[281] J. S. Bell, *Speakable and Unspeakable in Quantum Mechanics: Collected Papers on Quantum Philosophy*, 2nd ed. (Cambridge University Press, 2004).

[282] J. F. Clauser, M. A. Horne, A. Shimony, and R. A. Holt, Proposed experiment to test local hidden-variable theories, Phys. Rev. Lett. **23**, 880–884 (1969).

[283] A. Aspect, P. Grangier, and G. Roger, Experimental tests of realistic local theories via Bell's theorem, Phys. Rev. Lett. **47**, 460–463 (1981).

[284] A. Cabello, A. Feito, and A. Lamas-Linares, Bell's inequalities with realistic noise for polarization-entangled photons, Phys. Rev. A **72**, 052112 (2005).

[285] D. M. Greenberger, M. A. Horne, and A. Zeilinger, Going beyond Bell's theorem, in *Bell's Theorem, Quantum Theory and Conceptions of the Universe*, edited by M. Kafatos (Springer Netherlands, Dordrecht, 1989) pp. 69–72.

[286] D. Leibfried, E. Knill, S. Seidelin, J. Britton, R. B. Blakestad, J. Chiaverini, D. B. Hume, W. M. Itano, J. D. Jost, C. Langer, R. Ozeri, R. Reichle, and D. J. Wineland, Creation of a six-atom 'Schrödinger cat' state, Nature **438**, 639–642 (2005).

[287] N. Le Roux and Y. Bengio, Representational power of restricted Boltzmann machines and deep belief networks, Neural Comput. **20**, 1631–1649 (2008).

[288] S. Aaronson, The learnability of quantum states, P. Roy. Soc. A-Math. Phy. **463**, 3089–3114 (2007).

[289] C. Wetterich, Quantum computing with classical bits, Nucl. Phys. B **948**, 114776 (2019).

[290] S. Czischek, J. M. Pawlowski, T. Gasenzer, and M. Gärttner, Sampling scheme for neuromorphic simulation of entangled quantum systems, Phys. Rev. B **100**, 195120 (2019).

[291] S. Bluecher, J. M. Pawlowski, M. Scherzer, M. Schlosser, I.-O. Stamatescu, S. Syrkowski, and F. P. G. Ziegler, Reweighting Lefschetz thimbles, SciPost Phys. **5**, 44 (2018).

[292] D. Alvestad, R. Larsen, and A. Rothkopf, Stable solvers for real-time complex Langevin (2021), arXiv:2105.02735 [hep-lat] .

[293] S. Duane, A. Kennedy, B. J. Pendleton, and D. Roweth, Hybrid Monte Carlo, Phys. Lett. B **195**, 216 – 222 (1987).

[294] P. Smolensky, Information processing in dynamical systems: Foundations of harmony theory, in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations* (MIT Press, Cambridge, MA, USA, 1986) p. 194–281.

[295] J. Bloch, Reweighting complex Langevin trajectories, Phys. Rev. D **95**, 054509 (2017).

[296] A. M. Ferrenberg and R. H. Swendsen, New monte carlo technique for studying phase transitions, Phys. Rev. Lett. **61**, 2635–2638 (1988).

[297] R. Iwami, S. Ejiri, K. Kanaya, Y. Nakagawa, D. Yamamoto, and T. Umeda (WHOT-QCD Collaboration), Multipoint reweighting method and its applications to lattice qcd, Phys. Rev. D **92**, 094507 (2015).

[298] Bloch, Jacques, Glesaaen, Jonas, Philipsen, Owe, Verbaarschot, Jacobus, and Zafeiropoulos, Savvas, Complex langevin simulations of a finite density matrix model for qcd, EPJ Web Conf. **137**, 07030 (2017).

[299] A. Hosak, *Examining the one link SU3 model using complex Langevin and reweighting*, Bachelor's thesis, Universität Heidelberg (2021).

[300] M. Gori, G. Monfardini, and F. Scarselli, A new model for learning in graph domains, in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, Vol. 2 (2005) pp. 729–734 vol. 2.

[301] Y. Li, D. Tarlow, M. Brockschmidt, and R. S. Zemel, Gated graph sequence neural networks, in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, edited by Y. Bengio and Y. LeCun (2016).

[302] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, The graph neural network model, IEEE Transactions on Neural Networks **20**, 61–80 (2009).

[303] T. N. Kipf and M. Welling, Semi-Supervised Classification with Graph Convolutional Networks, in *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17 (2017).

[304] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, Graph convolutional networks: a comprehensive review, Computational Social Networks **6**, 11 (2019).

[305] W. L. Hamilton, R. Ying, and J. Leskovec, Representation learning on graphs: Methods and applications, IEEE Data Eng. Bull. **40**, 52–74 (2017).

[306] P. Goyal and E. Ferrara, Graph embedding techniques, applications, and performance: A survey, Knowledge-Based Systems **151**, 78–94 (2018).

[307] B. Li and D. Pi, Network representation learning: a systematic literature review, Neural Computing and Applications **32**, 16647–16679 (2020).

[308] S. Berretti, A. Del Bimbo, and E. Vicario, Efficient matching and indexing of graph models in content-based retrieval, IEEE Transactions on Pattern Analysis and Machine Intelligence **23**, 1089–1105 (2001).

[309] E. Bengoetxea, *Inexact Graph Matching Using Estimation of Distribution Algorithms*, Ph.D. thesis, Ecole Nationale Supérieure des Télécommunications, Paris, France (2002).

[310] R. Dijkman, M. Dumas, and L. García-Bañuelos, Graph matching algorithms for business process model similarity search, in *Business Process Management*, edited by U. Dayal, J. Eder, J. Koehler, and H. A. Reijers (Springer Berlin Heidelberg, Berlin, Heidelberg, 2009) pp. 48–63.

[311] J. B. Kruskal, Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis, Psychometrika **29**, 1–27 (1964).

[312] J. B. Tenenbaum, V. d. Silva, and J. C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science **290**, 2319–2323 (2000).

[313] B. Perozzi, R. Al-Rfou, and S. Skiena, Deepwalk: Online learning of social representations, in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14 (Association for Computing Machinery, New York, NY, USA, 2014) p. 701–710.

[314] D. Wang, P. Cui, and W. Zhu, Structural deep network embedding, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16 (Association for Computing Machinery, New York, NY, USA, 2016) p. 1225–1234.

[315] I. Chami, Z. Ying, C. Ré, and J. Leskovec, Hyperbolic graph convolutional neural networks, in *Advances in Neural Information Processing Systems*, Vol. 32, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc., 2019).

[316] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, Convolutional networks on graphs for learning molecular fingerprints, in *Advances in Neural Information Processing Systems*, Vol. 28, edited by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Curran Associates, Inc., 2015).

[317] Q. Liu, M. Allamanis, M. Brockschmidt, and A. Gaunt, Constrained graph variational autoencoders for molecule design, in *Advances in Neural Information Processing Systems*, Vol. 31, edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Curran Associates, Inc., 2018).

[318] Y. Bai, H. Ding, S. Bian, T. Chen, Y. Sun, and W. Wang, Simgnn: A neural network approach to fast graph similarity computation, in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19 (Association for Computing Machinery, New York, NY, USA, 2019) p. 384–392.

[319] Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli, Graph matching networks for learning the similarity of graph structured objects, in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, Proceedings of Machine Learning Research, Vol. 97, edited by K. Chaudhuri and R. Salakhutdinov (PMLR, 2019) pp. 3835–3845.

[320] R. Al-Rfou, D. Zelle, and B. Perozzi, Ddgk: Learning graph representations for deep divergence graph kernels, in *Proceedings of the 2019 World Wide Web Conference on World Wide Web* (2019).

[321] Z. Zeng, A. K. H. Tung, J. Wang, J. Feng, and L. Zhou, Comparing stars: On approximating graph edit distance, Proc. VLDB Endow. **2**, 25–36 (2009).

[322] K. Riesen and H. Bunke, Approximate graph edit distance computation by means of bipartite graph matching, Image and Vision Computing **27**, 950–959 (2009), 7th IAPR-TC15 Workshop on Graph-based Representations (GbR 2007).

[323] X. Gao, B. Xiao, D. Tao, and X. Li, A survey of graph edit distance, Pattern Analysis and Applications **13**, 113–129 (2010).

[324] M. Simonovsky and N. Komodakis, Graphvae: Towards generation of small graphs using variational autoencoders, in *Artificial Neural Networks and Machine Learning – ICANN 2018*, edited by V. Kůrková, Y. Manolopoulos, B. Hammer, L. Iliadis, and I. Maglogiannis (Springer International Publishing, Cham, 2018) pp. 412–422.

[325] H. Xiao, K. Rasul, and R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms (2017), cs.LG/1708.07747 .

[326] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. von Lilienfeld, Quantum chemistry structures and properties of 134 kilo molecules, Scientific Data **1**, 140022 (2014).

[327] S. J. Wetzel, R. G. Melko, J. Scott, M. Panju, and V. Ganesh, Discovering symmetry invariants and conserved quantities by interpreting siamese neural networks, Phys. Rev. Research **2**, 033499 (2020).

[328] M. Ancona, E. Ceolini, C. Öztireli, and M. Gross, Towards better understanding of gradient-based attribution methods for deep neural networks, in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings* (OpenReview.net, 2018).

[329] G. Montavon, W. Samek, and K.-R. Müller, Methods for interpreting and understanding deep neural networks, Digital Signal Processing **73**, 1–15 (2018).

[330] K. Simonyan, A. Vedaldi, and A. Zisserman, Deep inside convolutional networks: Visualising image classification models and saliency maps, in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*, edited by Y. Bengio and Y. LeCun (2014).

[331] M. Sundararajan, A. Taly, and Q. Yan, Axiomatic attribution for deep networks, in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17 (JMLR.org, 2017) p. 3319–3328.

[332] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation, PLOS ONE **10**, 1–46 (2015).

[333] A. Shrikumar, P. Greenside, and A. Kundaje, Learning important features through propagating activation differences, in *Proceedings of the 34th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 70, edited by D. Precup and Y. W. Teh (PMLR, 2017) pp. 3145–3153.

[334] M. D. Zeiler and R. Fergus, Visualizing and understanding convolutional networks, in *Computer Vision – ECCV 2014*, edited by D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars (Springer International Publishing, Cham, 2014) pp. 818–833.

[335] Fraunhofer HHI and TU Berlin, Heatmapping, accessed: 2021-08-23.

[336] K. A. Nicoli, P. Kessel, M. Gastegger, and K. T. Schütt, Analysis of atomistic representations using weighted skip-connections (2018), arXiv:1810.09751 [physics.comp-ph] .

[337] R. Fournier, L. Wang, O. V. Yazyev, and Q. Wu, An Artificial Neural Network Approach to the Analytic Continuation Problem, arXiv:1810.00913 [physics.comp-ph] .

[338] H. Yoon, J.-H. Sim, and M. J. Han, Analytic continuation via domain knowledge free machine learning, Phys. Rev. B **98**, 245101 (2018).

[339] A. K. Cyrol, J. M. Pawlowski, A. Rothkopf, and N. Wink, Reconstructing the gluon, SciPost Phys. **5**, 65 (2018).

[340] R. Oehme and W. Zimmermann, Gauge Field Propagator and the Number of Fermion Fields, Phys. Rev. **D21**, 1661 (1980).

[341] R. Oehme, On superconvergence relations in quantum chromodynamics, Phys. Lett. **B252**, 641–646 (1990).

[342] G. Cuniberti, E. De Micheli, and G. A. Viano, Reconstructing the thermal green functions at real times from those at imaginary times, Communications in Mathematical Physics **216**, 59–83 (2001).

[343] Y. Burnier, M. Laine, and L. Mether, A test on analytic continuation of thermal imaginary-time data, The European Physical Journal C **71**, 1619 (2011).

[344] C. N. dos Santos, K. Wadhawan, and B. Zhou, Learning loss functions for semi-supervised learning via discriminative adversarial networks (2017), arXiv:1707.02198 [cs.LG] .

[345] L. Wu, F. Tian, Y. Xia, Y. Fan, T. Qin, J. Lai, and T.-Y. Liu, Learning to Teach with Dynamic Loss Functions, arXiv:1810.12081 [cs.LG] .

[346] J. Horak, J. M. Pawlowski, J. Rodríguez-Quintero, J. Turnwald, J. M. Urban, N. Wink, and S. Zafeiropoulos, Reconstructing QCD Spectral Functions with Gaussian Processes (2021), arXiv:2107.13464 [hep-ph] .

[347] C. Yi, On the first passage time distribution of an Ornstein–Uhlenbeck process, Quantitative Finance **10**, 957–960 (2010).

[348] R. M. Neal, MCMC using Hamiltonian dynamics, Handbook of Markov Chain Monte Carlo **54**, 113–162 (2010).

[349] M. Betancourt, A Conceptual Introduction to Hamiltonian Monte Carlo (2018), arXiv:1701.02434 [stat.ME] .

[350] Y. Freund and D. Haussler, Unsupervised learning of distributions on binary vectors using two layer networks, in *Proceedings of the 4th International Conference on Neural Information Processing Systems*, NIPS'91 (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1991) p. 912–919.

[351] R. F. Werner, Quantum states with Einstein-Podolsky-Rosen correlations admitting a hidden-variable model, Phys. Rev. A **40**, 4277–4281 (1989).

[352] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, edited by Y. Bengio and Y. LeCun (2015).

[353] J. Carrasquilla, Machine learning for quantum matter, Advances in Physics: X **5**, 1797528 (2020).

[354] G. Torlai and R. G. Melko, Machine-learning quantum states in the nisq era, Annual Review of Condensed Matter Physics **11**, 325–344 (2020).

[355] O. Breitwieser, A. Baumbach, A. Korcsak-Gorzo, J. Klähn, M. Brixner, and M. Petrovici, sbs: Spike-based sampling (v1.8.2) (2020), This open source software code was developed in part in the Human Brain Project, funded from the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreement No. 720270 (HBP SGA1) and 785907 (HBP SGA2).

[356] D. von Suchodoletz, B. Wiebelt, K. Meier, and M. Janczyk, Flexible hpc: bwforcluster nemo, Proceedings of the 3rd bwHPCSymposium: Heidelberg (2016).

[357] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, Pytorch: An imperative style, high-performance deep learning library, in *Advances in Neural Information Processing Systems 32*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc., 2019) pp. 8024–8035.

[358] M. Fey and J. E. Lenssen, Fast graph representation learning with PyTorch Geometric, in *ICLR Workshop on Representation Learning on Graphs and Manifolds* (2019).

[359] Stan Development Team, Pystan: the python interface to stan, version 2.17.1.0, `http://mc-stan.org` (2018).

[360] B. Carpenter, A. Gelman, M. D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell, Stan: A probabilistic programming language, Journal of statistical software **76** (2017).