SOLVING THE QUANTUM MANY-BODY PROBLEM WITH NEURAL-NETWORK QUANTUM STATES

By

Jane Mee Kim

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Physics — Doctor of Philosophy
Computational Mathematics, Science and Engineering — Dual Major

2023

ABSTRACT

Quantum many-body systems provide a rich framework for exploring and understanding the fundamental laws of physics. By studying the collective behavior and emergent phenomena that arise from the intricate microscopic interactions among particles, we can deepen our understanding of quantum mechanics and gain insight into its broader implications for macroscopic observations. However, quantum many-body systems pose significant computational challenges, as the information contained in the many-body wave function grows exponentially with the size of the system. This exponential scaling, coupled with the presence of strong interparticle correlations, makes the accurate description of these systems difficult, if not impossible, for traditional analytical or perturbative techniques. In this interdisciplinary approach, we aim to solve the quantum many-body problem by representing the trial wave function of a variational Monte Carlo calculation by a so-called neural-network quantum state. These states, as their name suggests, are rooted in artificial neural networks and serve as a novel alternative to conventional parameterizations of the trial wave function.

In addition to reviewing key concepts in quantum many-body theory and machine learning, we investigate a diverse set of continuous-space systems with varying levels of complexity. Starting from a pedagogical overview of an exactly solvable system of bosons in one dimension, we work our way up to strongly-interacting fermionic systems in three dimensions, including dilute neutron matter and ultra-cold Fermi gases. The highly non-perturbative interactions featured in these systems motivate the development of innovative neural-network quantum states, capable of discovering strong correlations while maintaining required symmetries and boundary conditions. We accompany this study with the description of two distinct implementations of neural-network quantum states, each

with their unique goals and strategies. Our findings indicate that neural-network quantum states provide a powerful and flexible strategy for investigating a wide range of quantum phenomena, without relying on prior assumptions about the underlying physics as in traditional approaches.

This thesis is dedicated to my parents, who have been a constant source of love and support despite their distance, and to Elliot, my life partner and best friend, who fills my days with joy and laughter.

ACKNOWLEDGMENTS

Meeting Morten Hjorth-Jensen during my senior year at MSU was an incredibly fortunate event that profoundly influenced my academic journey. Knowing he would make an excellent mentor, he was a large part of the reason I decided to stay at MSU to pursue my PhD. His expertise, patience, and dedication have played a pivotal role in fostering my passion and shaping my scientific identity. Morten, I am forever grateful for your kindness, as well as your ability to strike a delicate balance between providing steadfast support and encouraging independent thinking. Please know that you have had an immense impact on my growth as a researcher and a human being.

Scott Pratt, I distinctly remember approaching your office one day after encountering the statement `"x = x + 1"` and thinking to myself, "Well, that's simply not true!" It seems trivial now, but this moment marked my very first introduction to coding. What's more, I knew nothing about research at the time, but you convinced me to give it a shot. Thank you for taking a chance on me, and thank you for your many words of wisdom.

To my esteemed committee members, Filomena Nunes, Artemis Spyrou, Huey-Wen Lin, Michael Murillo, and Stuart Tessmer (and a special mention to Matthew Hirn, who is no longer at MSU), I am grateful for your support and guidance throughout the years. You have not only fulfilled your roles as committee members but also served as invaluable role models for me. I would also like to express my gratitude to the nuclear theory group, particularly Dean Lee, Scott Bogner, and Heiko Hergert, for the countless stimulating discussions we have had over the years. I hope that our paths will cross often in the future.

Alessandro Lovato and Bryce Fore, it has been an absolute delight collaborating with both of you. I have learned so much during this past year and a half, and I eagerly look forward to our continued collaborations in the future. Gabriel Pescia, Jannes Nys, and

Giuseppe Carleo, I am deeply appreciative of the opportunity to work alongside each of you and to have had the unforgettable experience of meeting in person. The time spent together in Lausanne holds a special place in my heart, and I hope to have the chance to visit again in the future.

Throughout my nearly decade-long journey at MSU, my interests and career aspirations have undergone complete transformations. In the face of these changes, I am deeply grateful for the presence of Kim Crosslan in the P&A department. Kim, speaking not only for myself but for all graduate students, you are an incredibly important source of stability, and your impact is immeasurable. Thank you for the many times you have saved me.

My dearest Elliot, I can not imagine what graduate school would have been like if you had not been there for every second of it. Thank you for being you, and for helping me through stressful times. Umma and Appa, I am excited for you to finally see what I've been working on for the past few years. Thank you for always encouraging me to work through tough obstacles to strive towards my goals, hearing 'hwaiting!' is often exactly what I needed. Therese (Erin), thank you for your supportive words over the years. I miss you all deeply.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

1.1 Challenges of the Quantum Many-Body Problem

One of the fundamental challenges in addressing the quantum many-body problem lies in the extensive amount of information required to fully characterize a quantum state. In contrast to classical mechanics, where the focus is to determine the trajectories of the particles in the system, in quantum mechanics, our objective is to determine the relative probabilities of *all* potential states occurring for all particles. This distinction arises due to the probabilistic nature of quantum mechanics, where particles can exist in multiple states simultaneously and their behaviors are described by the evolution of wave functions.

All possible states of a quantum system can be represented as vectors in a many-body Hilbert space. As the number of particles increases, the dimension of the Hilbert space grows exponentially, resulting in a staggering number of possible states. This exponential growth poses a formidable computational challenge, as the explicit representation and manipulation of the wave function becomes intractable even for systems of modest size.

Consequently, quantum many-body theory is driven by the use of efficient and physically motivated approximations. Specific strategies vary depending on the type of problem at hand, but ideally, we want to encode as much relevant information about the system as possible in the fewest number of free parameters or degrees of freedom. In continuous space, the exponential scaling problem is magnified by an infinite basis. Nonetheless, resolving the spatial distribution of a quantum system remains of utmost interest, as it is often more intuitive to interpret physical phenomena in terms of position and spatial relationships.

1.2 Neural-Network Quantum States

In this study, we solve continuous-space quantum many-body problems by enhancing an established computational approach, called variational Monte Carlo, through the integration of machine learning—a category of models that learns from data rather than relying on predefined instructions. More specifically, we parameterize the trial wave function with artificial neural networks, computational models inspired by the structure of the human brain, and optimize their parameters by minimizing the energy of the system. Trial wave functions that involve these artificial neural networks are aptly named "neural-network quantum states".

Since their initial application by Carleo and Troyer in 2017 [1], neural-network quantum states have outperformed traditional variational Monte Carlo calculations in fields ranging from condensed matter physics [2] to quantum chemistry [3] and nuclear physics [4]. Remarkably, in many instances [2, 5], they have even surpassed diffusion Monte Carlo, a computational method widely regarded as the gold standard of quantum many-body methods. The rapid growth of this field underscores the tremendous potential demonstrated by neural-network quantum states. As we continue to develop and refine these techniques, new and exciting opportunities emerge for further breakthroughs that could reshape our understanding of complex quantum systems.

1.3 Algorithm Development Through the Lens of Nuclear Theory

Advancements in neural-network quantum states often find inspiration from established techniques in machine learning. While the prospect of adapting unexplored machine learning methods sparks curiosity, it is equally intriguing to explore the advantages that a background in physics can bring to overcoming these challenges.

In particular, the distinctive nature of nuclear interactions provides a compelling reason to design neural-network quantum states with broader applicability compared to states tailored for interactions commonly encountered in condensed matter systems. Nuclear Hamiltonians, which describe the interactions between protons and neutrons in real space, offer profound insights into the intricate mechanisms governing the universe—from the microscopic domain of atomic nuclei to the expansive reaches of neutron stars.

With the intention of eventually applying our neural-network quantum states to nuclear systems, we thoughtfully design the wave functions to ensure compatibility with the exchange of spin and isospin. For instance, in our investigation of ultra-cold Fermi gases in Sec. 9, we adopt a generalized approach to the Pfaffian wave function [6, 7], a type of wave function used in the presence of strong pairing correlations. Even though ultra-cold Fermi gases technically fall under the condensed matter category, our Pfaffian neural-network quantum state can be applied immediately to nuclear systems as well. By embracing a more inclusive approach, we discover a remarkably versatile and scalable ansatz.

1.4 Structure of Dissertation

This dissertation begins with a comprehensive overview of the quantum many-body problem in Chapter 2, including the establishment of notation and vocabulary used in later chapters. In Chapter 3, we discuss Quantum Monte Carlo methods, serving as a foundational framework to facilitate the incorporation of machine learning techniques for enhanced flexibility. Then in Chapter 4, we provide a broad background on machine learning, focusing on the mathematics of artificial neural networks. Chapter 5 presents two distinct approaches for the implementation of neural-network quantum states: one in C++ and the other in Python. The former allows for a deeper understanding of how the neural-network quantum states are trained, as gradients are computed analytically in terms of the trainable parameters, while the latter affords greater freedom in design, as gradients are computed numerically. Chapters 6-9 provide illustrative examples of neural-network quantum states employed in different quantum systems, with the latter three chapters being a collection of articles. Each of these examples will provide discussion on the power of neural-network quantum states as they are applied in the specific study. Finally, Chapter 10 includes conclusions and perspectives on the field as a whole.

2 The Quantum Many-Body Problem

The collective behavior of a quantum many-body system often displays remarkable deviations from the simple summation of the individual constituents. Strong correlations and intricate interactions between particles lead to emergent phenomena that cannot be adequately described by classical mechanics alone. These phenomena, such as phase transitions and the formation of superfluid states, underscore the importance of quantum mechanics in unraveling the complex behaviors and novel properties of quantum many-body systems.

In this study, our objective is to solve the time-independent Schrödinger equation for a nonrelativistic system of identical particles in continuous space. We will focus on determining the ground state, which plays a pivotal role in understanding the system's stability, symmetries, and equilibrium properties. Furthermore, the ground state serves as a reference point for all excited states, enabling meaningful comparisons to experimental observations.

The purpose of this chapter is to establish the foundation for our investigation. We will cover the essential aspects, significance, and challenges associated with quantum many-body problems, as well as examples of commonly employed ab initio methods. This discussion will help provide a comprehensive and contextualized understanding of the field and motivate our subsequent efforts to enhance quantum Monte Carlo methods with neural networks.

2.1 Formalism

Dirac notation, also known as bra-ket notation, offers a concise and abstract representation of quantum states, operators, and measurements. It is intimately linked to the concept of Hilbert space, a complex inner product space denoted as $\mathcal{H}$. A general quantum state is represented as a ket vector that lives in Hilbert space $|\Psi\rangle \in \mathcal{H}$. For each ket $|\Psi\rangle$, there exists a corresponding linear map from Hilbert space to the field of complex numbers $\langle\Psi| : \mathcal{H} \to \mathbb{C}$, which we call a bra vector. One can interpret a bra vector as a measurement device or probing tool that extracts useful information regarding a particular state. By applying a bra $\langle\Psi|$ to the left-hand side of another ket $|\Phi\rangle$, we define the inner product $\langle\Psi|\Phi\rangle$, a complex scalar that characterizes the overlap between the states $|\Psi\rangle$ and $|\Phi\rangle$. This inner product is essential for computing probabilities and expectation values of observables, as it enables us to properly define the notions of distance and orthogonality between states in Hilbert space.

Operators are linear transformations $\hat{A} : \mathcal{H} \to \mathcal{H}$ that act on a state to produce another state. While some operators, like time-evolution or rotation operators, do not correspond directly to measurable quantities, they still have mathematical and physical significance for describing the dynamics and symmetries of quantum states. Observables, such as position, momentum, and spin, are represented by Hermitian operators. Formally, they satisfy $\langle\hat{A}\Psi|\Phi\rangle = \langle\Psi|\hat{A}\Phi\rangle$ for any $|\Psi\rangle, |\Phi\rangle \in \mathcal{H}$. Hermitian operators have two important features:

1. *Real eigenvalues.* All possible measurements are real, implying that all expectation values $\langle\hat{A}\rangle \equiv \langle\Psi|\hat{A}|\Psi\rangle/\langle\Psi|\Psi\rangle$ are also real. The denominator ensures that the average is taken over a normalized probability distribution. These real expectation values play a crucial role in making comparisons and predictions with experimental results.

2. *Orthogonal eigenvectors.* The eigenvectors corresponding to distinct eigenvalues form

a set of orthonormal basis vectors $\{|\alpha\rangle\}$ that span the Hilbert space, yielding the completeness relation

$$\hat{1} = \sum_{\alpha} |\alpha\rangle\langle\alpha|. \tag{2.1}$$

Any state can be expanded as a linear combination of the eigenvectors

$$|\Psi\rangle = \left(\sum_{\alpha} |\alpha\rangle\langle\alpha|\right)|\Psi\rangle = \sum_{\alpha} \langle\alpha|\Psi\rangle|\alpha\rangle. \tag{2.2}$$

If the states are continuous, we convert the sum into an integral

$$\hat{1} = \int d\alpha |\alpha\rangle\langle\alpha|, \quad |\Psi\rangle = \int d\alpha \langle\alpha|\Psi\rangle|\alpha\rangle. \tag{2.3}$$

Since we will work with both continuous and discrete states, this distinction is of minor significance. It will be left to the reader to convert sums into integrals or vice versa, whenever they are not explicitly written.

Therefore, one should identify the observables which provide the most convenient set of basis states $\{|\alpha\rangle\}$ to represent the many-body state $|\Psi\rangle$. Determining the coefficients (or functions) $\langle\alpha|\Psi\rangle$ is equivalent to finding a representation of the abstract state $|\Psi\rangle$.

The best choice of basis for a many-body system depends on the specific nature of the particles and the chosen method for solving the problem. In preparation for our later treatment of N particles in d spatial dimensions, let us define our single-particle basis states as

$$|\boldsymbol{x}_i\rangle = \begin{cases} |\boldsymbol{r}_i\rangle, & \text{for bosons,} \\ |\boldsymbol{r}_i\rangle \otimes |\boldsymbol{s}_i\rangle \equiv |\boldsymbol{r}_i, \boldsymbol{s}_i\rangle, & \text{for fermions,} \end{cases} \tag{2.4}$$

where $\boldsymbol{r}_i \in \mathbb{R}^d$ are the coordinates of the ith particle, $\boldsymbol{s}_i \in \mathbb{R}^s$ contains the spin-like degrees of freedom (if there are any), and $i = 1, 2, ..., N$. The spatial parts are eigenstates of the single-particle position operator[1] $\hat{\boldsymbol{r}}_i$,

$$\hat{\boldsymbol{r}}_i|\boldsymbol{r}_i\rangle = \boldsymbol{r}_i|\boldsymbol{r}_i\rangle, \tag{2.5}$$

where the inner product between any two eigenstates is a d-dimensional Dirac delta function

$$\langle \boldsymbol{r}_i|\boldsymbol{r}_i'\rangle = \delta(\boldsymbol{r}_i - \boldsymbol{r}_i'). \tag{2.6}$$

The spin parts are written as

$$|\boldsymbol{s}_i\rangle = \begin{cases} |s_i^z\rangle, & \text{for spin-1/2 fermions,} \\ |s_i^z, t_i^z\rangle = |s_i^z\rangle \otimes |t_i^z\rangle, & \text{for nucleons,} \end{cases} \tag{2.7}$$

where $|s_i^z\rangle$ are the eigenstates of the single-particle spin-z operator $\hat{s}_i^z$,

$$\hat{s}_i^z|s_i^z\rangle = s_i^z|s_i^z\rangle, \quad \langle s_i^z|s_i^{z\prime}\rangle = \delta_{s_i^z s_i^{z\prime}}, \tag{2.8}$$

and $|t_i^z\rangle$ are the eigenstates of the single-particle isospin-z operator $\hat{t}_i^z$,

$$\hat{t}_i^z|t_i^z\rangle = t_i^z|t_i^z\rangle, \quad \langle t_i^z|t_i^{z\prime}\rangle = \delta_{t_i^z t_i^{z\prime}}, \tag{2.9}$$

[1] The position operator for the ith particle is commonly denoted as $\hat{\boldsymbol{X}}_i$ or $\hat{\boldsymbol{x}}_i$, but we chose $\hat{\boldsymbol{r}}_i$ to stay consistent with our definition of $\boldsymbol{x}_i$ and $\boldsymbol{r}_i$.

and δ_{ij} denotes the Kronecker delta. The resulting single-particle Hilbert spaces are

$$\mathcal{H}_i = \begin{cases} L^2(\mathbb{R}^d), & \text{for spin-0 bosons,} \\ L^2(\mathbb{R}^d) \otimes \mathbb{C}^2, & \text{for spin-1/2 fermions,} \\ L^2(\mathbb{R}^d) \otimes \mathbb{C}^2 \otimes \mathbb{C}^2, & \text{for nucleons,} \end{cases} \tag{2.10}$$

and a many-body configuration can be written as the tensor product of the single-particle basis states

$$|\boldsymbol{X}\rangle = |\boldsymbol{x}_1\rangle \otimes |\boldsymbol{x}_2\rangle \otimes \cdots \otimes |\boldsymbol{x}_N\rangle \equiv |\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\rangle \in \mathcal{H}, \tag{2.11}$$

living in a tensor product of single-particle Hilbert spaces

$$\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \cdots \otimes \mathcal{H}_N. \tag{2.12}$$

Assuming the dimensions of all the single-particle Hilbert spaces are the same, it is now clear that the dimension of the many-body Hilbert space scales exponentially with the number of particles

$$\dim(\mathcal{H}) = \prod_{i=1}^{N} \dim(\mathcal{H}_i) = \dim(\mathcal{H}_1)^N. \tag{2.13}$$

This is the primary source of difficulty in quantum many-body problems. The exponential scaling of the Hilbert space dimension applies for even the simplest spin systems in which $\dim(\mathcal{H}_1) = 2$. Our challenge is exacerbated by the fact that the particles in our systems of interest have continuous spatial degrees of freedom, resulting in an infinite $\dim(\mathcal{H}_1)$. Accordingly, quantum many-body methods are often built around different ways of

mitigating this problem, such as truncating Hilbert space to a finite subspace or exploring only relevant areas of Hilbert space stochastically.

Our many-body state $|\Psi\rangle$ can be expanded in terms of the many-body configurations defined in Eq. (2.11),

$$|\Psi\rangle = \int d\boldsymbol{X} \langle \boldsymbol{X}|\Psi\rangle |\boldsymbol{X}\rangle \tag{2.14}$$

where our goal is to determine the wave function $\Psi(\boldsymbol{X}) \equiv \langle \boldsymbol{X}|\Psi\rangle$, the probability amplitude of finding the system in a certain configuration $|\boldsymbol{X}\rangle$. Since we may have a mixture of continuous and discrete degrees of freedom, it is sometimes convenient to decompose $|\boldsymbol{X}\rangle$ as

$$|\boldsymbol{X}\rangle = |\boldsymbol{R}\rangle \otimes |\boldsymbol{S}\rangle, \tag{2.15}$$

$$|\boldsymbol{R}\rangle = |\boldsymbol{r}_1\rangle \otimes |\boldsymbol{r}_2\rangle \otimes \cdots \otimes |\boldsymbol{r}_N\rangle \equiv |\boldsymbol{r}_1, \boldsymbol{r}_2, ...\boldsymbol{r}_N\rangle, \tag{2.16}$$

$$|\boldsymbol{S}\rangle = |\boldsymbol{s}_1\rangle \otimes |\boldsymbol{s}_2\rangle \otimes \cdots \otimes |\boldsymbol{s}_N\rangle \equiv |\boldsymbol{s}_1, \boldsymbol{s}_2, ...\boldsymbol{s}_N\rangle, \tag{2.17}$$

instead of how it was originally expressed in Eq. (2.11). Then we can easily convert the integral over X into a discrete sum over the spin degrees of freedom and an integral over the continuous spatial degrees of freedom

$$\int d\boldsymbol{X} \longrightarrow \sum_{\boldsymbol{S}} \int d\boldsymbol{R} = \sum_{\boldsymbol{s}_1} \sum_{\boldsymbol{s}_2} \cdots \sum_{\boldsymbol{s}_N} \int d^d\boldsymbol{r}_1 d^d\boldsymbol{r}_2 \cdots d^d\boldsymbol{r}_N, \tag{2.18}$$

with the latter omitted in the bosonic case.

The single-particle momentum operator

$$\hat{\boldsymbol{p}}_i = -i\hbar \boldsymbol{\nabla}_i, \tag{2.19}$$

where $\boldsymbol{\nabla}_i \equiv \frac{\partial}{\partial \boldsymbol{r}_i}$ is the gradient with respect to position the ith particle, acts on the wave function as

$$\hat{\boldsymbol{p}}_i|\Psi\rangle = \int dX \langle X|\hat{\boldsymbol{p}}_i|\Psi\rangle|X\rangle = \int dX \Big(-i\hbar\boldsymbol{\nabla}_i\langle X|\Psi\rangle\Big)|X\rangle \tag{2.20}$$

The single-particle eigenstates are

$$\hat{\boldsymbol{p}}_i|\boldsymbol{p}_i\rangle = \boldsymbol{p}_i|\boldsymbol{p}_i\rangle, \quad \langle \boldsymbol{p}_i|\boldsymbol{p}_i'\rangle = \delta(\boldsymbol{p}_i - \boldsymbol{p}_i'), \tag{2.21}$$

with the continuous momenta $\boldsymbol{p}_i \in \mathbb{R}^d$ becoming discrete in the presence of periodicity. Expressed in position space, the momentum eigenfunctions are plane waves

$$\varphi_{\boldsymbol{p}_i}(\boldsymbol{x}_i) \equiv \langle \boldsymbol{x}_i|\boldsymbol{p}_i\rangle = \frac{1}{(2\pi\hbar)^{d/2}}\, e^{i\boldsymbol{p}_i\cdot\boldsymbol{r}_i/\hbar}, \tag{2.22}$$

which will become relevant in our studies of infinite matter.

Just as we can expand states in terms of our working basis, we can expand operators as well,

$$\hat{A} = \left(\int d\boldsymbol{X}|\boldsymbol{X}\rangle\langle\boldsymbol{X}|\right)\hat{A}\left(\int d\boldsymbol{X}'|\boldsymbol{X}'\rangle\langle\boldsymbol{X}'|\right) = \int d\boldsymbol{X}\int d\boldsymbol{X}'\langle\boldsymbol{X}|\hat{A}|\boldsymbol{X}'\rangle|\boldsymbol{X}\rangle\langle\boldsymbol{X}'|, \tag{2.23}$$

where the coefficients $\langle\boldsymbol{X}|\hat{A}|\boldsymbol{X}'\rangle$ are called matrix elements of the operator $\hat{A}$, even though our basis contains continuous components. If the operator is diagonal in our basis, we can

write

$$\begin{aligned}\hat{A} &= \int d\boldsymbol{X} \int d\boldsymbol{X}' \langle \boldsymbol{X}|\hat{A}|\boldsymbol{X}'\rangle |\boldsymbol{X}\rangle\langle \boldsymbol{X}'| \\ &= \int d\boldsymbol{X} \int d\boldsymbol{X}' \langle \boldsymbol{X}|\hat{A}|\boldsymbol{X}\rangle \delta(\boldsymbol{X}-\boldsymbol{X}')|\boldsymbol{X}\rangle\langle \boldsymbol{X}'| \\ &= \int d\boldsymbol{X} \langle \boldsymbol{X}|\hat{A}|\boldsymbol{X}\rangle |\boldsymbol{X}\rangle\langle \boldsymbol{X}| \\ &= \int d\boldsymbol{X} \hat{A}(\boldsymbol{X})|\boldsymbol{X}\rangle\langle \boldsymbol{X}|\end{aligned} \tag{2.24}$$

with $\delta(\boldsymbol{X}-\boldsymbol{X}') \equiv \delta(\boldsymbol{R}-\boldsymbol{R}')\delta_{\boldsymbol{S},\boldsymbol{S}'}$ representing the product of a Dirac delta function over the spatial degrees of freedom and a Kronecker delta over the spin degrees of freedom, if it applies. It is common to leave off the explicit dependence on $\boldsymbol{X}$ in the the function $\hat{A}(\boldsymbol{X}) \equiv \langle \boldsymbol{X}|\hat{A}|\boldsymbol{X}\rangle$, since this representation of $\hat{A}$ is particularly simple.

2.2 Indistinguishable Particles

The behavior of quantum systems exhibit fundamental differences from classical systems, primarily due to the concept of particle indistinguishability. In classical mechanics, particles are always distinguishable; even when they possess identical masses, charges, or other properties, they can be assigned different labels to unambiguously describe and predict their motion. As a result, classical systems feature well-defined trajectories for each particle, albeit often involving highly nonlinear coupled differential equations.

When we established our mathematical framework in the previous section, we implicitly assumed that the particles were distinguishable. This is evident in Eq. (2.11) and Eq. (2.12), for example, where each particle and single-particle Hilbert space was given an index corresponding to their ordering. Now, let us consider what happens if we assume the particles are indistinguishable. More specifically, we assume that the expectation value of

any observable $\hat{A}$ is invariant under any permutation $\hat{P} \in S_N$ of the N particle indices

$$\frac{\langle\Psi|\hat{A}|\Psi\rangle}{\langle\Psi|\Psi\rangle} = \frac{\langle\hat{P}\Psi|\hat{A}|\hat{P}\Psi\rangle}{\langle\hat{P}\Psi|\hat{P}\Psi\rangle} = \frac{\langle\Psi|\hat{P}^{-1}\hat{A}\hat{P}|\Psi\rangle}{\langle\Psi|\Psi\rangle}, \tag{2.25}$$

where we have used that $\hat{P}^\dagger = \hat{P}^{-1}$ in the last equality. The above equation implies that $[\hat{A}, \hat{P}] = 0$.

Since any permutation in the symmetric group S_N can be written as a product of transpositions, it is sufficient to investigate the action of the transposition operator $\hat{P}_{ij}$, also known as the particle exchange operator, which trades the labeling of the ith and jth particle.

$$\hat{P}_{ij}|\boldsymbol{X}\rangle = \hat{P}_{ij}|\boldsymbol{x}_1, ..., \boldsymbol{x}_i, ..., \boldsymbol{x}_j, ..., \boldsymbol{x}_N\rangle = |\boldsymbol{x}_1, ..., \boldsymbol{x}_j, ..., \boldsymbol{x}_i, ..., \boldsymbol{x}_N\rangle. \tag{2.26}$$

Applying the same particle exchange operator twice restores the original ordering

$$\hat{P}_{ij}^2|\boldsymbol{X}\rangle = \hat{P}_{ij}|\boldsymbol{x}_1, ..., \boldsymbol{x}_j, ..., \boldsymbol{x}_i, ..., \boldsymbol{x}_N\rangle = |\boldsymbol{x}_1, ..., \boldsymbol{x}_i, ..., \boldsymbol{x}_j, ..., \boldsymbol{x}_N\rangle = |\boldsymbol{X}\rangle, \tag{2.27}$$

implying the eigenvalues of $\hat{P}_{ij}$ are ± 1. Thus, the possible effects of exchanging the particle ordering

$$\hat{P}_{ij}|\Psi_{\mathcal{S}}\rangle = |\Psi_{\mathcal{S}}\rangle, \qquad \text{for bosons,} \tag{2.28}$$

$$\hat{P}_{ij}|\Psi_{\mathcal{A}}\rangle = -|\Psi_{\mathcal{A}}\rangle, \qquad \text{for fermions,} \tag{2.29}$$

give rise to two fundamentally different classes of identical particles: bosons, which have purely symmetric states and obey Bose-Einstein statistics, and fermions, which have purely antisymmetric states and obey Fermi-Dirac statistics. The subscripts $\mathcal{S}$ and $\mathcal{A}$ stand for

symmetric and antisymmetric, respectively. By the spin-statistics theorem, the difference between the two categories is attributed to an intrinsic property called spin. Bosons, such as photons and gluons, have integer values of spin, while fermions, such as electrons, protons, and neutrons, have half-integer values. We will only consider spin-0 and spin-1/2 particles in this work, as shown already in Eq. (2.10). In Sec. 9, we will discover that the line distinguishing bosons and fermions is sometimes less distinct than it initially appears.

It is straightforward to see how the action of the particle exchange operator can be generalized to an arbitrary permutation $\hat{P} \in S_N$. The sign, or parity, of the permutation is $\sigma(P) = (-1)^p$, where p is the number of transpositions required to decompose the permutation into a product of transpositions[2]. Then the permutation acts as

$$\hat{P}|\Psi_{\mathcal{S}}\rangle = |\Psi_{\mathcal{S}}\rangle, \qquad \text{for bosons,} \tag{2.30}$$

$$\hat{P}|\Psi_{\mathcal{A}}\rangle = \sigma(P)|\Psi_{\mathcal{A}}\rangle, \qquad \text{for fermions,} \tag{2.31}$$

for the two cases. Our working basis states $|X\rangle$ are not eigenstates themselves, but we can construct symmetric and antisymmetric states from them. In fact, we can define the symmetrization operator

$$\hat{\mathcal{S}} = \frac{1}{N!} \sum_{\hat{P} \in S_N} \hat{P}, \tag{2.32}$$

and the antisymmetrization operator

$$\hat{\mathcal{A}} = \frac{1}{N!} \sum_{\hat{P} \in S_N} \sigma(P) \hat{P}, \tag{2.33}$$

[2]There exists an infinite number of ways to decompose a permutation into a product of transpositions. However, an odd permutation will invariably be decomposed into an odd number of transpositions, and an even permutation will be decomposed into an even number of transpositions.

that will map any many-body state $|\Psi\rangle \in \mathcal{H}$ to the equivalent symmetric and antisymmetric states

$$\hat{\mathcal{S}}|\Psi\rangle \equiv |\Psi_{\mathcal{S}}\rangle, \quad \hat{\mathcal{A}}|\Psi\rangle \equiv |\Psi_{\mathcal{A}}\rangle. \tag{2.34}$$

Both of these operators are idempotent,

$$\hat{\mathcal{S}}^2 = \hat{\mathcal{S}}, \quad \hat{\mathcal{A}}^2 = \hat{\mathcal{A}}, \tag{2.35}$$

meaning they project generic states in Hilbert space to symmetric and antisymmetric subspaces, respectively,

$$|\Psi_{\mathcal{S}}\rangle \in \mathcal{H}_{\mathcal{S}}, \quad |\Psi_{\mathcal{A}}\rangle \in \mathcal{H}_{\mathcal{A}}. \tag{2.36}$$

Applying the symmetrization operator after already applying the antisymmetrization operator destroys any state, and vice versa

$$\hat{\mathcal{S}}\hat{\mathcal{A}} = \hat{\mathcal{A}}\hat{\mathcal{S}} = 0, \tag{2.37}$$

implying the subspaces $\mathcal{H}_{\mathcal{S}}$ and $\mathcal{H}_{\mathcal{A}}$ are entirely distinct. Futhermore, any two states that are related by a permutation

$$|\Psi'\rangle = \hat{P}'|\Psi\rangle, \quad \hat{P}' \in S_N, \tag{2.38}$$

will map to the same symmetric state

$$|\Psi'_S\rangle = \hat{\mathcal{S}}|\Psi'\rangle = \hat{\mathcal{S}}|\Psi\rangle = |\Psi_S\rangle, \tag{2.39}$$

and to the same antisymmetric state except for a possible sign flip

$$|\Psi'_A\rangle = \hat{\mathcal{A}}|\Psi'\rangle = \sigma(P')\hat{\mathcal{A}}|\Psi\rangle = \sigma(P')|\Psi_A\rangle. \tag{2.40}$$

Therefore, the many-body Hilbert space $\mathcal{H}$ that we established in Eq. (2.12) for distinguishable particles is typically much larger than the direct sum of the subspaces for indistinguishable particles

$$\mathcal{H}_{\mathcal{S}} \oplus \mathcal{H}_{\mathcal{A}} \subseteq \mathcal{H}. \tag{2.41}$$

This is spectacular news! The physically realizable quantum states of our system will either be in $\mathcal{H}_{\mathcal{S}}$ or $\mathcal{H}_{\mathcal{A}}$, which is smaller than the entire Hilbert space $\mathcal{H}$ by a factor that scales as $N!$, the order of the symmetric group S_N. While this does not alleviate the dimensionality problem entirely because our Hilbert space is still infinite dimensional, it does help make the many-body problem more tractable. Enforcing other symmetries, such as parity and time-reversal, can similarly help reduce the effective size of our Hilbert space.

Now that we are able to construct both symmetric and antisymmetric states from a generic one, let us consider what happens if two (or more) indistinguishable particles occupy the same exact single-particle state. Then there exists a particle exchange operator $\hat{P}_{ij}$ for some $i \neq j$ that generates a symmetry of the many-body state,

$$\hat{P}_{ij}|\Psi_{ij}\rangle = |\Psi_{ij}\rangle. \tag{2.42}$$

Here, we have denoted the state as $|\Psi_{ij}\rangle$ to specify that it is invariant under $i \leftrightarrow j$. No issues arise for the corresponding bosonic system according to Eq. (2.39), so bosons are fine with being on top of their identical neighbors. However, for the fermionic system, Eq. (2.40)

implies a contradiction

$$\hat{\mathcal{A}}|\Psi_{ij}\rangle = -\hat{\mathcal{A}}|\Psi_{ij}\rangle. \tag{2.43}$$

The above equation can only be possible if $\hat{\mathcal{A}}|\Psi_{ij}\rangle = 0$, leading to the well-known Pauli exclusion principle. This fundamental principle asserts that no two identical fermions can occupy the same quantum state simultaneously. As a result, fermions exhibit distinct behavior from bosons, particularly at low temperatures.

2.3 The Schrödinger Equation

The behavior of a non-relativistic quantum system is governed by the Schrödinger equation

$$i\hbar\frac{\partial}{\partial t}|\Psi\rangle = \hat{H}|\Psi\rangle, \tag{2.44}$$

where the Hamiltonian $\hat{H}$ is a Hermitian operator that represents the total energy of the system. We will exclusively work with a time-independent Hamiltonian that is diagonal in our working basis $|\boldsymbol{X}\rangle$, meaning

$$\hat{H} = \int d\boldsymbol{X}\hat{H}(\boldsymbol{X})|\boldsymbol{X}\rangle\langle\boldsymbol{X}|, \tag{2.45}$$

where $\hat{H}(\boldsymbol{X}) \equiv \langle\boldsymbol{X}|\hat{H}|\boldsymbol{X}\rangle$ as defined in Eq. (2.24). Our Hamiltonian takes the form

$$\hat{H} = \hat{K} + \hat{V}, \tag{2.46}$$

where the non-relativistic kinetic energy is defined as

$$\hat{K} = \frac{1}{2m}\sum_{i=1}^{N}\hat{\boldsymbol{p}}_i^2 = -\frac{\hbar^2}{2m}\sum_{i=1}^{N}\boldsymbol{\nabla}_i^2 \tag{2.47}$$

and the potential energy $\hat{V} = V(\boldsymbol{X})$ may include external potentials, two- or three-body interactions, and spin- or isospin-dependence. While the above Hamiltonian is not dependent on time, the state $|\Psi\rangle$ still could be. We have suppressed this time-dependence thus far, but we will now make it explicit by writing

$$|\Psi(t)\rangle = \int d\boldsymbol{X}\langle\boldsymbol{X}|\Psi(t)\rangle|\boldsymbol{X}\rangle, \tag{2.48}$$

where the time-dependent wave function can be decomposed as

$$\langle\boldsymbol{X}|\Psi(t)\rangle \equiv \Psi(\boldsymbol{X},t) = \psi(\boldsymbol{X})\phi(t), \tag{2.49}$$

following the separation of variables method of solving partial differential equations. Then the Schrödinger equation reads

$$i\hbar\frac{\partial}{\partial t}\Psi(\boldsymbol{X},t) = \hat{H}(\boldsymbol{X})\Psi(\boldsymbol{X},t) \quad\Longrightarrow\quad i\hbar\psi(\boldsymbol{X})\frac{\partial}{\partial t}\phi(t) = \phi(t)\hat{H}(\boldsymbol{X})\psi(\boldsymbol{X}). \tag{2.50}$$

Rearranging all of the t-dependent components on the left and the X-dependent components on the right,

$$\frac{i\hbar}{\phi(t)}\frac{\partial}{\partial t}\phi(t) = \frac{1}{\psi(\boldsymbol{X})}\hat{H}(\boldsymbol{X})\psi(\boldsymbol{X}) = E, \tag{2.51}$$

we find that each side of the equation must be equal to some constant E. The left-hand side can be solved immediately

$$i\hbar\frac{\partial}{\partial t}\phi(t) = E\phi(t) \quad \Longrightarrow \quad \phi(t) = \phi(0)e^{-iEt/\hbar}. \tag{2.52}$$

The right-hand side, however, is an eigenvalue problem

$$\hat{H}(\boldsymbol{X})\psi(\boldsymbol{X}) = E\psi(\boldsymbol{X}), \tag{2.53}$$

also known as the time-independent Schrödinger equation.

Eigenvalue problems are challenging because they are inherently nonlinear. What's more, the eigenvectors in our situation are actually continuous eigenfunctions rather than finite-dimensional vectors. However, once we identify the solutions of the time-independent Schrödinger equation above, we will automatically obtain the solutions of the full time-dependent Schrödinger equation as well. Let us index the possible solutions with α, which can either label the discrete energy spectrum for a bound system or the continuous spectrum for an unbound system. Then the so-called stationary states are

$$\Psi_\alpha(\boldsymbol{X}, t) = \psi_\alpha(\boldsymbol{X})e^{-iE_\alpha t/\hbar}, \tag{2.54}$$

where we omit writing the constant $\phi(0)$ from Eq. (2.52) because it can be absorbed into the coefficients c_α of the general solution, a linear combination of stationary states

$$\Psi(\boldsymbol{X}, t) = \sum_\alpha c_\alpha \Psi_\alpha(\boldsymbol{X}, t). \tag{2.55}$$

The stationary states are called "stationary" because their associated probability distributions do not depend on time

$$|\Psi_\alpha(\boldsymbol{X},t)|^2 = \psi_\alpha^*(\boldsymbol{X})e^{iE\alpha t/\hbar}\psi_\alpha(\boldsymbol{X})e^{-iE\alpha t/\hbar} = |\psi_\alpha(\boldsymbol{X})|^2. \tag{2.56}$$

Given that

$$|\Psi(\boldsymbol{X},t)|^2 = \sum_\alpha \sum_\beta c_\alpha^* c_\beta \psi_\alpha^*(\boldsymbol{X})\psi_\beta(\boldsymbol{X})e^{i(E_\alpha - E_\beta)t/\hbar}, \tag{2.57}$$

the same cannot be said about the general solution.

2.4 The Variational Principle

While the solutions of the time-dependent Schrödinger equation are vital for understanding the dynamics of the quantum system, it is a necessary step to first solve the time-independent Schrödinger equation. However, the exact time-independent solutions and their corresponding energies are still difficult, if not impossible, to obtain. One valuable tool for finding approximate solutions is called the variational principle, which states that the expectation value of the Hamiltonian is an upper bound for the true ground state energy of the system E_0,

$$E[\Psi] \equiv \langle \hat{H} \rangle = \frac{\langle \Psi | \hat{H} | \Psi \rangle}{\langle \Psi | \Psi \rangle} \geq E_0, \tag{2.58}$$

for any state $|\Psi\rangle$. It is simple to prove. First, we write the eigenstates of the Hamiltonian as $|\Psi_\alpha\rangle = |\alpha\rangle$ and their corresponding energies as E_α. The eigenstates form a complete orthonormal basis with $\langle \alpha | \beta \rangle = \delta_{\alpha\beta}$. If we assume $|\Psi\rangle$ is normalized, $\langle \Psi | \Psi \rangle = 1$, then the

expectation value of the energy can be expanded as

$$\begin{aligned} E[\Psi] &= \langle\Psi|\hat{H}|\Psi\rangle \\ &= \sum_{\alpha}\sum_{\beta}\langle\Psi|\alpha\rangle\langle\alpha|\hat{H}|\beta\rangle\langle\beta|\Psi\rangle \\ &= \sum_{\alpha}\sum_{\beta}\langle\Psi|\alpha\rangle\langle\beta|\Psi\rangle E_{\alpha}\delta_{\alpha\beta} \\ &= \sum_{\alpha} E_{\alpha}|\langle\alpha|\Psi\rangle|^2. \end{aligned} \tag{2.59}$$

The ground state energy E_0, by definition, is smaller than all the other energy eigenvalues $\{E_\alpha\}_{\alpha\neq 0}$, leading to the result

$$E[\Psi] = \sum_{\alpha} E_{\alpha}|\langle\alpha|\Psi\rangle|^2 \geq E_0, \tag{2.60}$$

where equality can only hold for the ground state $|\Psi_0\rangle$.

Similar variational principles can be developed for the excited states. First, let us order our energy eigenstates such that $E_0 \leq E_1 \leq E_2...$, where E_1 is the energy of the first excited state $|\Psi_1\rangle$, E_2 is the energy of the second excited state $|\Psi_2\rangle$, and so on. Then the variational principle for the nth excited state is given by

$$E[\Psi] = \frac{\langle\Psi|\hat{H}|\Psi\rangle}{\langle\Psi|\Psi\rangle} \geq E_n, \tag{2.61}$$

if our state $|\Psi\rangle$ is orthogonal to all the lower-lying states,

$$\langle\Psi_m|\Psi\rangle = 0 \text{ for all } m < n. \tag{2.62}$$

This statement can be proved by the same method as Eq. (2.59). We have used the Latin indicies m and n in place of the Greek indicies α and β to emphasize that our eigenstates have been ordered according to their corresponding energies.

Another useful fact is that the variance of energy is only zero if and only if our state is an eigenstate of the Hamiltonian

$$\sigma_E^2 = 0 \iff |\Psi\rangle = |\Psi_\alpha\rangle \text{ for some } \alpha. \tag{2.63}$$

The backward direction of the statement above is trivial to prove, so we will just prove the forward direction here. Expanding the variance of the energy in terms of the energy eigenstates, we obtain

$$\begin{aligned}
\sigma_E^2 &= \langle (\hat{H} - \langle \hat{H} \rangle)^2 \rangle = \langle \hat{H}^2 \rangle - \langle \hat{H} \rangle^2 \\
&= \sum_\alpha E_\alpha^2 |\langle \alpha | \Psi \rangle|^2 - \left(\sum_\alpha E_\alpha |\langle \alpha | \Psi \rangle|^2 \right) \left(\sum_\beta E_\beta |\langle \beta | \Psi \rangle|^2 \right) \\
&= \sum_\alpha E_\alpha^2 |\langle \alpha | \Psi \rangle|^2 - \sum_\alpha \sum_\beta E_\alpha E_\beta |\langle \alpha | \Psi \rangle|^2 |\langle \beta | \Psi \rangle|^2 \\
&= \sum_\alpha E_\alpha |\langle \alpha | \Psi \rangle|^2 \left(E_\alpha - \sum_\beta E_\beta |\langle \beta | \Psi \rangle|^2 \right).
\end{aligned} \tag{2.64}$$

There are only two ways that the above can equal zero *for all* α: either $|\langle \alpha | \Psi \rangle|^2 = 0$ or

$$E_\alpha = \sum_\beta E_\beta |\langle \beta | \Psi \rangle|^2 = E_\alpha |\langle \alpha | \Psi \rangle|^2 \implies |\langle \alpha | \Psi \rangle|^2 = 1. \tag{2.65}$$

Therefore, the state $|\Psi\rangle$ must be an eigenstate of the Hamiltonian for some α. A nonzero variance of the energy indicates that the system is not in a well-defined state, but rather, a

superposition of eigenstates.

We must address an important caveat regarding the variational principle. Specifically, the variational principle only holds true if the state adheres to the correct symmetries of the system. The many-body Hamiltonian often does not provide information about whether the system consists of bosons or fermions. For example, when describing a system of identical charged particles, the Coulomb interaction only depends on the charge and distances between the pairs. If one employs a state intended for bosons when computing the expectation value of the energy for a fermionic system, it can lead to energies much lower than the true ground state energy of the system. Such erroneous results are not violations of the variational principle; instead, they arise due to incorrect implementation.

2.5 Ab Initio Methods

A many-body method is called ab initio (or "from first principles") if it only relies on fundamental principles and established laws of nature. It involves solving the Schrödinger equation starting from a microscopic Hamiltonian rather than a Hamiltonian derived from empirical or experimental data. In this section, we will provide examples of common ab initio methods. While some these methods are not explicitly formulated for real-space systems, they can often be adapted and applied to such systems with the appropriate modifications.

2.5.1 Configuration Interaction

Full Configuration Interaction is an exact method that aims to solve the time-independent Schrödinger equation through diagonalization of the Hamiltonian in matrix form. We begin by expanding the wave function as a linear combination of orthonormal many-body basis

states $\{|\alpha\rangle\}$,

$$|\Psi\rangle = \sum_{\alpha} c_\alpha |\alpha\rangle, \tag{2.66}$$

where c_α are the coefficients of the expansion and the states $|\alpha\rangle$ are usually taken to be all possible antisymmetrized products of single-particle basis states, also known as Slater determinants, for fermionic systems. By inserting this expansion and the completeness relation into the Schrödinger equation, we obtain

$$\hat{H}|\Psi\rangle = E|\Psi\rangle, \tag{2.67}$$

$$\sum_{\alpha} \hat{H} c_\alpha |\alpha\rangle = E \sum_{\alpha} c_\alpha |\alpha\rangle, \tag{2.68}$$

$$\sum_{\alpha}\sum_{\beta} \langle\beta|\hat{H}|\alpha\rangle c_\alpha |\beta\rangle = E \sum_{\alpha} c_\alpha |\alpha\rangle. \tag{2.69}$$

Since the basis states are orthogonal, multiplying on the left by $\langle\gamma|$ yields

$$\sum_{\alpha} \langle\gamma|\hat{H}|\alpha\rangle c_\alpha = E c_\gamma, \tag{2.70}$$

which can be written in a more convenient form by organizing the coefficients c_α into a vector and $\langle\gamma|\hat{H}|\alpha\rangle$ into a matrix

$$\begin{bmatrix} \langle 1|\hat{H}|1\rangle & \langle 1|\hat{H}|2\rangle & \cdots & \langle 1|\hat{H}|\alpha\rangle & \cdots \\ \langle 2|\hat{H}|1\rangle & \langle 2|\hat{H}|2\rangle & \cdots & \langle 2|\hat{H}|\alpha\rangle & \cdots \\ \vdots & \vdots & \ddots & \vdots & \cdots \\ \langle \alpha|\hat{H}|1\rangle & \langle \alpha|\hat{H}|2\rangle & \cdots & \langle \alpha|\hat{H}|\alpha\rangle & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_\alpha \\ \vdots \end{bmatrix} = E \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_\alpha \\ \vdots \end{bmatrix} \tag{2.71}$$

Clearly, solving the eigenvalue problem above is only imaginable when the matrix is of finite size. However, even in such cases, the dimensionality of the problem often poses significant challenges and renders it computationally intractable.

One alternative approach is to truncate the many-body basis to a finite, manageable subset $\{|\alpha_1\rangle, |\alpha_2\rangle, ..., |\alpha_n\rangle\}$, leading to the more commonly-employed method called Configuration Interaction. In truncating our many-body basis, it becomes important to consider which states we include in our subset. Typically, the most relevant states are constructed by identifying a suitable reference state, like the Hartree-Fock solution, and constructing excitations or configurations around it.

2.5.2 Hartree-Fock Theory

Hartree-Fock theory is one of the simplest approximate methods for solving the many-body Schrödinger equation. It provides a mean-field description of the system, where each particle moves in an effective average field created by all other particles. This approach essentially decouples the two-body interaction, resulting in a tractable computational scheme, but it neglects correlation effects, which can be significant in systems with strong interactions.

Let us assume that the Hamiltonian of a fermionic system contains up to two-body interactions

$$\hat{H}(\boldsymbol{X}) = \hat{H}_0(\boldsymbol{X}) + \hat{V}(\boldsymbol{X}), \tag{2.72}$$

$$\hat{H}_0(\boldsymbol{X}) = \sum_{i=1}^{N} \hat{h}_0(\boldsymbol{x}_i), \tag{2.73}$$

$$\hat{V}(\boldsymbol{X}) = \sum_{i<j}^{N} \hat{v}(\boldsymbol{x}_{ij}), \tag{2.74}$$

where we have divided the terms in the Hamiltonian such that $\hat{H}_0$ contains only one-body contributions and $\hat{V}$ contains the pair-wise interactions. Accordingly they can be decomposed into individual one-body and two-body operators. The notation $\boldsymbol{x}_{ij}$ is meant to imply that the two-body interactions are symmetric with respect to i and j.

To achieve the best mean-field description of the interacting problem, our goal is to find an effective one-body Hamiltonian that approximates the original one. Namely,

$$\hat{H}(\boldsymbol{X}) \approx \sum_{i=1}^{N} \hat{f}(\boldsymbol{x}_i), \tag{2.75}$$

where $\hat{f}$ can be further decomposed as $\hat{f}(\boldsymbol{x}_i) = \hat{h}_0(\boldsymbol{x}_i) + \hat{v}^{HF}(\boldsymbol{x}_i)$, for some choice of $\hat{v}^{HF}$ that is to be determined. Then eigenstates of this mysterious one-body Hamiltonian must satisfy

$$\hat{f}(\boldsymbol{x})\varphi_\alpha(\boldsymbol{x}) = \varepsilon_\alpha \varphi_\alpha(\boldsymbol{x}), \tag{2.76}$$

with the corresponding single-particle energies ε_α.

In the Hartree-Fock approximation, the many-body wave function $\Phi^{HF}(\boldsymbol{X})$ is taken to be a Slater determinant of these single-particle eigenstates. The Slater determinant is a natural result of applying the antisymmetrization operator (Eq. (2.33)) to a product of single-particle wave functions

$$\begin{aligned}
\Phi^{HF}(\boldsymbol{X}) &= \sqrt{N!}\, \hat{\mathcal{A}}\{\varphi_1(\boldsymbol{x}_1)\varphi_2(\boldsymbol{x}_2)\cdots\varphi_N(\boldsymbol{x}_N)\} \\
&= \frac{1}{\sqrt{N!}} \det \begin{bmatrix} \varphi_1(\boldsymbol{x}_1) & \varphi_1(\boldsymbol{x}_2) & \cdots & \varphi_1(\boldsymbol{x}_N) \\ \varphi_2(\boldsymbol{x}_1) & \varphi_2(\boldsymbol{x}_2) & \cdots & \varphi_2(\boldsymbol{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_N(\boldsymbol{x}_1) & \varphi_N(\boldsymbol{x}_2) & \cdots & \varphi_N(\boldsymbol{x}_N) \end{bmatrix}.
\end{aligned} \tag{2.77}$$

The above assumption only applies for fermions, but an equivalent assumption can be made for bosons by applying the symmetrization operator (Eq. (2.32)) to the product, resulting in a permanent instead of a determinant. To simplify the notation moving forward, let us define $\Phi(\boldsymbol{X}) \equiv \varphi_1(\boldsymbol{x}_1)\varphi_2(\boldsymbol{x}_2)\cdots\varphi_N(\boldsymbol{x}_N)$ to mean the product of single-particle eigenstates before they have been antisymmetrized. Then $\Phi^{HF}(\boldsymbol{X}) = \sqrt{N!}\hat{\mathcal{A}}\Phi(\boldsymbol{X})$.

According to the variational principle, the best set of orthogonal single-particle states $\varphi_\alpha(\boldsymbol{x})$ would minimize the energy using the Hartree-Fock ansatz $\Phi^{HF}(\boldsymbol{X})$. The expectation value of the energy is

$$E[\Phi^{HF}] = \langle\Phi^{HF}|\hat{H}|\Phi^{HF}\rangle = \langle\Phi^{HF}|\hat{H}_0|\Phi^{HF}\rangle + \langle\Phi^{HF}|\hat{V}|\Phi^{HF}\rangle, \tag{2.78}$$

where we have assumed $\Phi^{HF}(\boldsymbol{X})$ is normalized. Each of these terms can be evaluated by using the known properties of the antisymmetrization operator and the orthogonality of the single-particle states $\varphi_\alpha(\boldsymbol{x}) = \langle\boldsymbol{x}|\alpha\rangle$. The calculations for both are as follows:

$$\begin{aligned}
&\langle\Phi^{HF}|\hat{H}_0|\Phi^{HF}\rangle \\
&= \int d\boldsymbol{X}\Phi^{HF*}(\boldsymbol{X})\hat{H}_0(\boldsymbol{X})\Phi^{HF}(\boldsymbol{X}) \\
&= N!\int d\boldsymbol{X}\Phi^*(\boldsymbol{X})\big[\hat{\mathcal{A}}\hat{H}_0(\boldsymbol{X})\big]\hat{\mathcal{A}}\Phi(\boldsymbol{X}) = N!\int d\boldsymbol{X}\Phi^*(\boldsymbol{X})\big[\hat{H}_0(\boldsymbol{X})\hat{\mathcal{A}}\big]\hat{\mathcal{A}}\Phi(\boldsymbol{X}) \\
&= N!\int d\boldsymbol{X}\Phi^*(\boldsymbol{X})\hat{H}_0(\boldsymbol{X})\big[\hat{\mathcal{A}}\hat{\mathcal{A}}\big]\Phi(\boldsymbol{X}) = N!\int d\boldsymbol{X}\Phi^*(\boldsymbol{X})\hat{H}_0(\boldsymbol{X})\hat{\mathcal{A}}\Phi(\boldsymbol{X}) \\
&= \sum_{i=1}^{N}\sum_{\hat{P}\in S_N}\sigma(P)\int d\boldsymbol{X}\Phi^*(\boldsymbol{X})\hat{h}_0(\boldsymbol{x}_i)\hat{P}\Phi(\boldsymbol{X}) = \sum_{i=1}^{N}\int d\boldsymbol{X}\Phi^*(\boldsymbol{X})\hat{h}_0(\boldsymbol{x}_i)\Phi(\boldsymbol{X}) \\
&= \sum_{\alpha=1}^{N}\int d\boldsymbol{x}\varphi_\alpha^*(\boldsymbol{x})\hat{h}_0(\boldsymbol{x})\varphi_\alpha(\boldsymbol{x}) = \sum_{\alpha=1}^{N}\langle\alpha|\hat{h}_0|\alpha\rangle,
\end{aligned} \tag{2.79}$$

$$
\begin{aligned}
&\langle \Phi^{HF}|\hat{V}|\Phi^{HF}\rangle \\
&= \int d\boldsymbol{X}\Phi^{HF*}(\boldsymbol{X})\hat{V}(\boldsymbol{X})\Phi^{HF}(\boldsymbol{X}) \\
&= N!\int d\boldsymbol{X}\Phi^{*}(\boldsymbol{X})\big[\hat{\mathcal{A}}\hat{V}(\boldsymbol{X})\big]\hat{\mathcal{A}}\Phi(\boldsymbol{X}) = N!\int d\boldsymbol{X}\Phi^{*}(\boldsymbol{X})\big[\hat{V}(\boldsymbol{X})\hat{\mathcal{A}}\big]\hat{\mathcal{A}}\Phi(\boldsymbol{X}) \\
&= N!\int d\boldsymbol{X}\Phi^{*}(\boldsymbol{X})\hat{V}(\boldsymbol{X})\big[\hat{\mathcal{A}}\hat{\mathcal{A}}\big]\Phi(\boldsymbol{X}) = N!\int d\boldsymbol{X}\Phi^{*}(\boldsymbol{X})\hat{V}(\boldsymbol{X})\hat{\mathcal{A}}\Phi(\boldsymbol{X}) \\
&= \sum_{i<j}^{N}\sum_{\hat{P}\in S_N}\sigma(P)\int d\boldsymbol{X}\Phi^{*}(\boldsymbol{X})\hat{v}(\boldsymbol{x}_{ij})\hat{P}\Phi(\boldsymbol{X}) = \sum_{i<j}^{N}\int d\boldsymbol{X}\Phi^{*}(\boldsymbol{X})\hat{v}(\boldsymbol{x}_{ij})(1-\hat{P}_{ij})\Phi(\boldsymbol{X}) \\
&= \frac{1}{2}\sum_{\alpha\beta}^{N}\int d\boldsymbol{x}_1 d\boldsymbol{x}_2\Big[\varphi_\alpha^*(\boldsymbol{x}_1)\varphi_\beta^*(\boldsymbol{x}_2)\hat{v}(\boldsymbol{x}_{12})\varphi_\alpha(\boldsymbol{x}_1)\varphi_\beta(\boldsymbol{x}_2) - \varphi_\alpha^*(\boldsymbol{x}_1)\varphi_\beta^*(\boldsymbol{x}_2)\hat{v}(\boldsymbol{x}_{12})\varphi_\alpha(\boldsymbol{x}_2)\varphi_\beta(\boldsymbol{x}_1)\Big] \\
&= \frac{1}{2}\sum_{\alpha\beta}^{N}\Big[\langle\alpha\beta|\hat{v}|\alpha\beta\rangle - \langle\alpha\beta|\hat{v}|\beta\alpha\rangle\Big] \equiv \frac{1}{2}\sum_{\alpha\beta}^{N}\langle\alpha\beta|\hat{v}|\alpha\beta\rangle_{\mathcal{A}}.
\end{aligned}
\tag{2.80}
$$

Now that we know

$$
E[\Phi^{HF}] = \sum_{\alpha=1}^{N}\langle\alpha|\hat{h}_0|\alpha\rangle + \frac{1}{2}\sum_{\alpha\beta}^{N}\langle\alpha\beta|\hat{v}|\alpha\beta\rangle_{\mathcal{A}}, \tag{2.81}
$$

we can now vary the orbitals $\varphi_\alpha(\boldsymbol{x})$ in order to find the minimum $E[\Phi^{HF}]$. Alternatively, we can perform a unitary transformation on our basis, expanding $|\alpha\rangle$ as a linear combination of well-known, orthogonal basis states $|i\rangle$, and vary the coefficients. We will choose the latter method, specifically with the basis states coming from the non-interacting problem,

$$
|\alpha\rangle = \sum_i C_{\alpha i}|i\rangle, \tag{2.82}
$$

where

$$\hat{f}|\alpha\rangle = \varepsilon_\alpha|\alpha\rangle, \tag{2.83}$$

$$\hat{h}_0|i\rangle = \varepsilon_i^0|i\rangle. \tag{2.84}$$

In terms of these expansion coefficients, the expectation value of the energy becomes

$$E[\Phi^{HF}] = \sum_{\alpha=1}^{N}\sum_{ij} C_{\alpha i}^* C_{\alpha j}\langle i|\hat{h}_0|j\rangle + \frac{1}{2}\sum_{\alpha\beta}^{N}\sum_{ijk\ell} C_{\alpha i}^* C_{\beta j}^* C_{\alpha k} C_{\beta \ell}\langle ij|\hat{v}|k\ell\rangle_{\mathcal{A}} \tag{2.85}$$

Recognizing that what we really desire is Eq. (2.75), we can then write

$$E[\Phi^{HF}] = \sum_{\alpha=1}^{N}\langle\alpha|\hat{f}|\alpha\rangle = \sum_{\alpha=1}^{N}\sum_{ij} C_{\alpha i}^* C_{\alpha j}\langle i|\hat{f}|j\rangle. \tag{2.86}$$

By comparing this equation with the one above it, we find

$$\langle i|\hat{f}|j\rangle = \langle i|\hat{h}_0|j\rangle + \sum_{\alpha}\sum_{k\ell} C_{\alpha k}^* C_{\beta \ell}\langle ik|\hat{v}|jl\rangle_{\mathcal{A}}, \tag{2.87}$$

which simplifies to

$$\langle i|\hat{f}|j\rangle = \varepsilon_i^0 + \sum_{\alpha}\sum_{k\ell} C_{\alpha k}^* C_{\beta \ell}\langle ik|\hat{v}|jl\rangle_{\mathcal{A}}, \tag{2.88}$$

because the states $|i\rangle$ are eigenstates of $\hat{h}_0$. Next, by expanding Eq. (2.83) in terms of $|j\rangle$ and projecting on the left with $\langle i|$, we find that we can write

$$\sum_j C_{\alpha j}\langle i|\hat{f}|j\rangle = \varepsilon_\alpha C_{\alpha i}. \tag{2.89}$$

This means that if we organize Eq. (2.86) into a matrix and the coefficients $C_{\alpha i}$ into a vector

for each α, the above can be written as a normal eigenvalue problem $FC_\alpha = \varepsilon_\alpha C_\alpha$.

The Hartree-Fock eigenvalue problem can be solved iteratively. First, we provide guesses for the initial $C_\alpha^{(0)}$, where the supercript indicates the iteration. Then for each iteration, we construct the matrix $F^{(t)}$, which has elements $F_{ij}^{(t)} = \langle i|\hat{f}|j\rangle$ using the previous coefficients $C_\alpha^{(t-1)}$. By solving the eigenvalue problem through diagonalization, we obtain new eigenvectors $C_\alpha^{(t)}$ and their corresponding eigenvalues $\varepsilon_\alpha^{(t)}$. We continue this process until the eigenvalues converge.

2.5.3 Many-Body Perturbation Theory

Many-body perturbation theory offers a systematic framework for incorporating the effects of interparticle correlations beyond the Hartree-Fock level. The Hamiltonian is assumed to consist of two parts, an unperturbed Hamiltonian $\hat{H}_0$ and an interacting Hamiltonian $\hat{H}_I$,

$$\hat{H} = \hat{H}_0 + \hat{H}_I, \tag{2.90}$$

where the solutions for the unperturbed case are easy to find

$$\hat{H}_0|\Phi_n\rangle = W_n|\Phi_n\rangle. \tag{2.91}$$

Let us split the completeness relation such that the projector onto the subspace spanned by the ground state is separated from the rest,

$$\hat{1} = \sum_{n=0}^{\infty}|\Phi_n\rangle\langle\Phi_n| = |\Phi_0\rangle\langle\Phi_0| + \sum_{n=1}^{\infty}|\Phi_n\rangle\langle\Phi_n| = \hat{P} + \hat{Q}. \tag{2.92}$$

Then we can expand the ground state $|\Psi_0\rangle$ of the full Hamiltonian $\hat{H}$ as

$$|\Psi_0\rangle = (\hat{P} + \hat{Q})|\Psi_0\rangle = |\Phi_0\rangle + \hat{Q}|\Psi_0\rangle = |\Phi_0\rangle + \sum_{n=1}^{\infty} \langle \Phi_n|\Psi_0\rangle |\Phi_n\rangle, \tag{2.93}$$

where we have made the assumption $\langle \Phi_0|\Psi_0\rangle = 1$. By applying the full Hamiltonian on the above state and projecting with $\langle \Phi_0|$, we obtain

$$\langle \Phi_0|\hat{H}|\Psi_0\rangle = \langle \Phi_0|\hat{H}_0|\Phi_0\rangle + \langle \Phi_0|\hat{H}_I|\Psi_0\rangle = W_0 + \langle \Phi_0|\hat{H}_I|\Psi_0\rangle, \tag{2.94}$$

due to the orthogonality of the eigenstates $|\Phi_n\rangle$. In addition, by applying $\langle \Phi_0|$ to the Schrödinger equation for the full Hamiltonian, we obtain

$$\langle \Phi_0|\hat{H}|\Psi_0\rangle = E_0 \langle \Phi_0|\Psi_0\rangle = E_0, \tag{2.95}$$

implying that $E_0 = W_0 + \Delta E$, where $\Delta E \equiv \langle \Phi_0|\hat{H}_I|\Psi_0\rangle$. Through simple manipulations of the Schrödinger equation, we find

$$\begin{aligned} \hat{H}|\Psi_0\rangle &= \hat{H}_0|\Psi_0\rangle + \hat{H}_I|\Psi_0\rangle = E_0\hat{H}|\Psi_0\rangle \\ (\hat{H}_0 - W_0)|\Psi_0\rangle &= (E_0 - W_0 - \hat{H}_I)|\Psi_0\rangle \\ (W_0 - \hat{H}_0)|\Psi_0\rangle &= (\hat{H}_I - E_0)|\Psi_0\rangle \end{aligned} \tag{2.96}$$

Since $\hat{Q}$ is idempotent and commutes with the Hamiltonian, applying $\hat{Q}$ to both sides of the above equation yields

$$\begin{aligned}
\hat{Q}(\hat{H}_I - \Delta E)|\Psi_0\rangle &= \hat{Q}(W_0 - \hat{H}_0)|\Psi_0\rangle \\
&= \hat{Q}\hat{Q}(W_0 - \hat{H}_0)|\Psi_0\rangle \\
&= \hat{Q}(W_0 - \hat{H}_0)\hat{Q}|\Psi_0\rangle \\
&= \sum_{m,n=1}^{\infty} |\Phi_m\rangle\langle\Phi_m|(W_0 - \hat{H}_0)|\Phi_n\rangle\langle\Phi_n|\Psi_0\rangle \\
&= \sum_{m,n=1}^{\infty} (W_0 - W_m)\delta_{mn}|\Phi_m\rangle\langle\Phi_n|\Psi_0\rangle \\
&= \sum_{n=1}^{\infty} (W_0 - W_n)|\Phi_n\rangle\langle\Phi_n|\Psi_0\rangle \\
&= (W_0 - \hat{H}_0)\hat{Q}|\Psi_0\rangle.
\end{aligned} \tag{2.97}$$

Inserting this equation into Eq. (2.93) yields

$$|\Psi_0\rangle = \frac{\hat{Q}}{W_0 - \hat{H}_0}(\hat{H}_I - \Delta E)|\Psi_0\rangle, \tag{2.98}$$

which can be recursively inserted into itself

$$|\Psi_0\rangle = \sum_{n=0}^{\infty} \left[\frac{\hat{Q}}{W_0 - \hat{H}_0}(\hat{H}_I - \Delta E)\right]^n |\Phi_0\rangle. \tag{2.99}$$

Likewise, the implications for the correlation energy is as follows:

$$\Delta E = \sum_{n=0}^{\infty} \langle\Phi_0| \left[\frac{\hat{Q}}{W_0 - \hat{H}_0}(\hat{H}_I - \Delta E)\right]^n |\Phi_0\rangle. \tag{2.100}$$

By organizing the terms in the above by order in $\hat{H}_I$, the correlation energy can be written as

$$\Delta E = \sum_{n=1}^{\infty} \Delta E^{(n)}, \tag{2.101}$$

where the first few terms in the expansion are

$$\Delta E^{(1)} = \langle \Phi_0 | \hat{H}_I | \Phi_0 \rangle \tag{2.102}$$

$$\Delta E^{(2)} = \langle \Phi_0 | \hat{H}_I \frac{\hat{Q}}{W_0 - \hat{H}_0} \hat{H}_I | \Phi_0 \rangle \tag{2.103}$$

$$\Delta E^{(3)} = \langle \Phi_0 | \hat{H}_I \frac{\hat{Q}}{W_0 - \hat{H}_0} \hat{H}_I \frac{\hat{Q}}{W_0 - \hat{H}_0} \hat{H}_I | \Phi_0 \rangle \tag{2.104}$$

$$- \langle \Phi_0 | \hat{H}_I \frac{\hat{Q}}{W_0 - \hat{H}_0} \langle \Phi_0 | \hat{H}_I | \Phi_0 \rangle \frac{\hat{Q}}{W_0 - \hat{H}_0} \hat{H}_I | \Phi_0 \rangle \tag{2.105}$$

3 Quantum Monte Carlo

In this chapter, we introduce quantum Monte Carlo (QMC), a diverse family of ab initio many-body methods based on simulating the quantum system and computing its properties stochastically. We will cover two different flavors of QMC methods: variational Monte Carlo (VMC) and diffusion Monte Carlo (DMC). The former will take center stage in this study, as our neural-network quantum states are designed to push the limits of this method. The latter consistently yields state-of-the-art results, so it serves as a benchmark for our neural-network quantum states.

We will begin by establishing the basics of Monte Carlo sampling and integration. Both VMC and DMC use these techniques extensively, as implied by their names. Then, we will delve into the details of the conventional implementation of variational Monte Carlo. This will not only include an extensive list of commonly employed trial wave functions, but also the optimization algorithms used to update the parameters of the wave function. We will use these same optimization methods to train our neural-network quantum states, discussed throughout the remaining chapters of this thesis. We will end with a brief overview of diffusion Monte Carlo. While these methods can be used to solve for excited states, we will focus only on the ground state in our investigation.

3.1 Monte Carlo Methods

3.1.1 Markov Chain Monte Carlo Sampling

Monte Carlo sampling methods are used to generate random samples from a target probability distribution. In our case, the target distribution $P(\boldsymbol{X})$ is given by the square of

the wave function $|\Psi(\boldsymbol{X})|^2$, which is a function of continuous spatial degrees of freedom, and possibly discrete spin degrees of freedom as well. However, our wave function is not guaranteed to be normalized, so it is more precise to say

$$P(\boldsymbol{X}) = \frac{1}{Z}|\Psi(\boldsymbol{X})|^2, \tag{3.1}$$

where $Z = \int dX|\Psi(\boldsymbol{X})|^2$ is the normalization constant. It is impractical to compute Z directly since it involves a high-dimensional integral over all the spatial degrees of freedom and a sum over all the spin degrees of freedom. Luckily, there are clever tricks we can use to avoid computing Z, leading to a style of sampling called Markov Chain Monte Carlo (MCMC) sampling.

MCMC algorithms, in general, are designed to address the problem of sampling from high-dimensional and nontrivial distributions. Instead of generating independent samples from $P(\boldsymbol{X})$, as done with a standard hit-or-miss algorithm for instance, we can produce a discrete sequence of samples $\{\boldsymbol{X}_1, \boldsymbol{X}_2, ..., \boldsymbol{X}_t, ...\}$, where each sample is dependent only on the previous one. Then the transition probability for the Markov process must have the simple form

$$P(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{X}_{t-2}, ..., \boldsymbol{X}_0) = P(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}), \tag{3.2}$$

for each $t > 0$. Our goal is to accurately approximate the target distribution $P(\boldsymbol{X})$ by the distribution of a finite number of samples

$$P(\boldsymbol{X}) \approx \frac{1}{T}\sum_{t=1}^{T}\delta(\boldsymbol{X} - \boldsymbol{X}_t), \tag{3.3}$$

which is only possible if $P(\boldsymbol{X})$ is invariant under the action of the transition probability. In

other words, $P(\boldsymbol{X})$ is the stationary distribution for the Markov process characterized by $P(\boldsymbol{X}'|\boldsymbol{X})$ if it satisfies

$$P(\boldsymbol{X}') = \int d\boldsymbol{X} P(\boldsymbol{X}'|\boldsymbol{X}) P(\boldsymbol{X}), \tag{3.4}$$

for all $\boldsymbol{X}'$.

In principle, the above stationary condition provides a way to evolve to $P(\boldsymbol{X})$ starting from any initial distribution $P(\boldsymbol{X}_0)$. The distribution at a later time t can be obtained by repeatedly applying the transition probability on $P(\boldsymbol{X}_0)$,

$$P(\boldsymbol{X}_t) = \int d\boldsymbol{X}_{t-1} d\boldsymbol{X}_{t-2} \cdots d\boldsymbol{X}_0 P(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}) P(\boldsymbol{X}_{t-1}|\boldsymbol{X}_{t-2}) \cdots P(\boldsymbol{X}_1|\boldsymbol{X}_0) P(\boldsymbol{X}_0). \tag{3.5}$$

For large enough t, the Markov chain must converge to the stationary distribution

$$\lim_{t\to\infty} P(\boldsymbol{X}_t) = P(\boldsymbol{X}), \tag{3.6}$$

regardless of the initial state $\boldsymbol{X}_0$. The algorithm for generating the samples is as follows:

1. *Initialize.* Draw the first sample $\boldsymbol{X}_0$ from any initial probability distribution $P(\boldsymbol{X}_0)$.

2. *Iterate.* For each iteration $t = 1, ..., T$, draw the next sample $\boldsymbol{X}_t$ from $P(\boldsymbol{X}_t)$ in Eq. (3.5).

While it is simple to state the algorithm, the second step is somewhat difficult to complete without additional restrictions.

In order to treat the samples in a Markov chain as independent samples from the target distribution, the chain should be fully equilibrated and the autocorrelation between samples should be low. Let us define τ as the number of steps required to sufficiently reach the

stationary distribution, also known as the equilibration or burn-in time. In addition, let us define γ as the number of steps required to sufficiently reduce the autocorrelation between samples

$$\langle \boldsymbol{X}_t \cdot \boldsymbol{X}_{t+\gamma} \rangle \approx 0. \tag{3.7}$$

Then, our target distribution can be better approximated by

$$P(\boldsymbol{X}) \approx \frac{1}{T} \sum_{t=1}^{T} \delta(\boldsymbol{X} - \boldsymbol{X}_{\tau+\gamma t}), \tag{3.8}$$

for some large, but finite value of T. This approach allows us to faithfully treat each effective sample $\boldsymbol{X}_{\tau+\gamma t}$ as independent samples from $P(\boldsymbol{X})$, at the cost of throwing away $\tau+(\gamma-1)T$ samples out of the total number of samples $\tau + \gamma T$.

3.1.1.1 Metropolis-Hastings

The Metropolis-Hastings algorithm is a specific type of MCMC sampling algorithm that enforces the detailed balance condition, a stricter restriction than the stationary condition in Eq. (3.4). Namely, it assumes the Markov process is reversible

$$P(\boldsymbol{X}'|\boldsymbol{X})P(\boldsymbol{X}) = P(\boldsymbol{X}|\boldsymbol{X}')P(\boldsymbol{X}'), \tag{3.9}$$

for every pair of states $\boldsymbol{X}, \boldsymbol{X}'$. By integrating both sides of the detailed balance condition,

$$\int d\boldsymbol{X} P(\boldsymbol{X}'|\boldsymbol{X})P(\boldsymbol{X}) = \int d\boldsymbol{X} P(\boldsymbol{X}|\boldsymbol{X}')P(\boldsymbol{X}') = P(\boldsymbol{X}') \int d\boldsymbol{X} P(\boldsymbol{X}|\boldsymbol{X}') = P(\boldsymbol{X}'), \tag{3.10}$$

it automatically follows that $P(\boldsymbol{X})$ is the stationary distribution.

In addition, this approach involves decomposing the transition probability into two components

$$P(\boldsymbol{X}'|\boldsymbol{X}) = Q(\boldsymbol{X}'|\boldsymbol{X})A(\boldsymbol{X}'|\boldsymbol{X}), \tag{3.11}$$

where the proposal probability density $Q(\boldsymbol{X}'|\boldsymbol{X})$ suggests a candidate state $\boldsymbol{X}'$ based on the current state $\boldsymbol{X}$, and the acceptance probability $A(\boldsymbol{X}'|\boldsymbol{X})$ determines if the proposed state should be accepted or rejected. Plugging this into the detailed balance equation yields

$$\frac{A(\boldsymbol{X}'|\boldsymbol{X})}{A(\boldsymbol{X}|\boldsymbol{X}')} = \frac{Q(\boldsymbol{X}|\boldsymbol{X}')P(\boldsymbol{X}')}{Q(\boldsymbol{X}'|\boldsymbol{X})P(\boldsymbol{X})}. \tag{3.12}$$

If $A(\boldsymbol{X}|\boldsymbol{X}') = 1$, then $A(\boldsymbol{X}'|\boldsymbol{X}) = \frac{Q(\boldsymbol{X}|\boldsymbol{X}')P(\boldsymbol{X}')}{Q(\boldsymbol{X}'|\boldsymbol{X})P(\boldsymbol{X})}$. Likewise, if $A(\boldsymbol{X}'|\boldsymbol{X}) = 1$ then $A(\boldsymbol{X}'|\boldsymbol{X}) = \frac{Q(\boldsymbol{X}'|\boldsymbol{X})P(\boldsymbol{X})}{Q(\boldsymbol{X}|\boldsymbol{X}')P(\boldsymbol{X}')}$. Thus the choice of $A(\boldsymbol{X}'|\boldsymbol{X})$ that maximizes the acceptance probability is

$$A(\boldsymbol{X}'|\boldsymbol{X}) \equiv \min\left\{1, \frac{Q(\boldsymbol{X}|\boldsymbol{X}')P(\boldsymbol{X}')}{Q(\boldsymbol{X}'|\boldsymbol{X})P(\boldsymbol{X})}\right\}. \tag{3.13}$$

In practice, the Metropolis-Hastings algorithm can be realized by separating the transition step into two parts:

1. *Initialize.* Draw the first sample $\boldsymbol{X}_0$ from an initial probability distribution $P(\boldsymbol{X}_0)$.

2. *Iterate.* For each iteration $t = 1, ..., T$:

   (a) *Propose.* Draw a candidate state $\boldsymbol{X}'$ based on the previous state $\boldsymbol{X}_{t-1}$ according to the proposal probability density $Q(\boldsymbol{X}'|\boldsymbol{X}_{t-1})$.

   (b) *Calculate.* Calculate the acceptance probability of the transition $A(\boldsymbol{X}'|\boldsymbol{X}_{t-1})$.

   (c) *Accept-or-reject.* Draw a uniform random number $r \sim \mathcal{U}(0, 1)$ between 0 and 1. If $r < A(\boldsymbol{X}'|\boldsymbol{X}_{t-1})$, accept the transition by setting $\boldsymbol{X}_t = \boldsymbol{X}'$. Otherwise, reject

the transition by setting $\boldsymbol{X}_t = \boldsymbol{X}_{t-1}$.

Since our states may involve discrete spin projections $\boldsymbol{S}$ in addition to the continuous spatial coordinates $\boldsymbol{R}$, we will further decompose all the probabilities as

$$P(\boldsymbol{X}) = P(\boldsymbol{R})P(\boldsymbol{S}), \tag{3.14}$$

$$Q(\boldsymbol{X}'|\boldsymbol{X}) = Q(\boldsymbol{R}'|\boldsymbol{R})Q(\boldsymbol{S}'|\boldsymbol{S}), \tag{3.15}$$

$$A(\boldsymbol{X}'|\boldsymbol{X}) = A(\boldsymbol{R}'|\boldsymbol{R})A(\boldsymbol{S}'|\boldsymbol{S}), \tag{3.16}$$

and perform Metropolis steps in $\boldsymbol{R}$ and $\boldsymbol{S}$ separately. For a localized system of particles, one can take the initial distribution of the spatial degrees of freedom as

$$P(\boldsymbol{R}_0) \sim \mathcal{N}(\boldsymbol{0}, \sigma_0^2 \boldsymbol{I}), \tag{3.17}$$

with a variance σ_0^2 chosen to minimize the equilibration time τ. For an infinite system, one can take the initial distribution as a uniform distribution

$$P(\boldsymbol{R}_0) \sim \mathcal{U}(-L/2, L/2), \tag{3.18}$$

ranging the entire size of the d-dimensional simulation box of side length L. The proposal probability density for the spatial coordinates is commonly chosen to be a Gaussian distribution

$$Q(\boldsymbol{R}'|\boldsymbol{R}) \sim \mathcal{N}(\boldsymbol{R}, \sigma^2 \boldsymbol{I}), \tag{3.19}$$

with a variance σ^2 that balances a short autocorrelation time γ with a high acceptance probability $A(\boldsymbol{R}'|\boldsymbol{R})$. Since the proposal probability is symmetric, $Q(\boldsymbol{R}'|\boldsymbol{R}) = Q(\boldsymbol{R}|\boldsymbol{R}')$,

the acceptance probability for the move simplifies to

$$A(\boldsymbol{R}'|\boldsymbol{R}) = \min\left\{1, \frac{P(\boldsymbol{R}')}{P(\boldsymbol{R})}\right\} = \min\left\{1, \frac{|\Psi(\boldsymbol{R}', \boldsymbol{S})|^2}{|\Psi(\boldsymbol{R}, \boldsymbol{S})|^2}\right\}. \tag{3.20}$$

For the spin degrees of freedom $\boldsymbol{S}$, it is possible to take a similar strategy as above by initializing a random configuration of spins and proposing random spins to flip rather than coordinates to perturb. However, the Hamiltonians we will use always commute with the total spin projection operator on the z-axis

$$\hat{S}^z = \sum_{i=1}^{N} \hat{s}_i^z, \tag{3.21}$$

meaning the total spin projection $S^z = \sum_{i=1}^{N} s_i^z$ is conserved. Similarly, the total isospin projection on the z-axis T^z is also preserved for systems of nucleons, as the total isospin projection operator

$$\hat{T}^z = \sum_{i=1}^{N} \hat{t}_i^z \tag{3.22}$$

commutes with the nuclear Hamiltonian. To avoid sampling unphysical spin configurations, we can restrict our Metropolis walk to preserve the total spin (and isospin) projection as follows:

1. *Initialize.* Generate any spin configuration $\boldsymbol{S}_0$ with the desired total spin projection S^z (and isospin projection T^z). Spin-up particles are assigned a spin value of $+1$, while the spin-down particles are assigned -1.

2. *Iterate.* For each iteration $t = 1, 2, ..., T$:

   (a) *Propose.* Choose a random *pair* of particles (i, j) to exchange spin degrees of

freedom. The proposal spin configuration $\boldsymbol{S}'$ is the same as $\boldsymbol{S}_{t-1}$, except with the spins of particle i and j exchanged

$$\begin{aligned} \boldsymbol{s}_i' &= \boldsymbol{s}_{j,t-1}, \\ \boldsymbol{s}_j' &= \boldsymbol{s}_{i,t-1}. \end{aligned} \tag{3.23}$$

A different pair can be chosen to exchange the isospin degrees of freedom, if desired.

(b) *Compute.* Compute the acceptance probability of the proposed configuration

$$A(\boldsymbol{S}'|\boldsymbol{S}_{t-1}) = \min\left\{1, \frac{P(\boldsymbol{S}')}{P(\boldsymbol{S}_{t-1})}\right\} = \min\left\{1, \frac{|\Psi(\boldsymbol{R}, \boldsymbol{S}')|^2}{|\Psi(\boldsymbol{R}, \boldsymbol{S}_{t-1})|^2}\right\}. \tag{3.24}$$

(c) *Accept-or-reject.* Draw a uniform random number $r \sim \mathcal{U}(0,1)$ between 0 and 1. If $r < A(\boldsymbol{S}'|\boldsymbol{S}_{t-1})$, accept the spin exchange by setting $\boldsymbol{S}_t = \boldsymbol{S}'$. Otherwise, reject the exchange by setting $\boldsymbol{S}_t = \boldsymbol{S}_{t-1}$.

3.1.1.2 Importance Sampling

Importance sampling, a variant of the Metropolis-Hastings algorithm, draws inspiration from the Fokker-Planck equation—a generalization of the diffusion equation. The basic idea of importance sampling is to guide the proposal probability for the spatial degrees of freedom $Q(\boldsymbol{R}'|\boldsymbol{R})$ towards regions with higher probability $P(\boldsymbol{R})$. For a general, time-dependent probability distribution function, the Fokker-Planck equation of a diffusion process reads

$$\frac{\partial}{\partial t} P(\boldsymbol{R}, t) = D\boldsymbol{\nabla} \cdot (\boldsymbol{\nabla} - \boldsymbol{F}(\boldsymbol{R}))\, P(\boldsymbol{R}, t), \tag{3.25}$$

where $\boldsymbol{\nabla} = (\boldsymbol{\nabla}_1, \boldsymbol{\nabla}_2, ..., \boldsymbol{\nabla}_N)$ is the combination of all gradient operators for the N particles, $D = \frac{\hbar^2}{2m}$ is a constant diffusion coefficient, and $\boldsymbol{F}(\boldsymbol{R})$ is the drift velocity due to an external potential. Here, we have also suppressed all dependence on the spins $\boldsymbol{S}$, as they remain constant during these steps. Our goal is find the drift velocity $\boldsymbol{F}(\boldsymbol{R})$ such that our probability distribution converges to the stationary distribution determined by our wave function

$$P(\boldsymbol{R}, t) = P(\boldsymbol{R}) = |\Psi(\boldsymbol{R})|^2 \implies \frac{\partial}{\partial t} P(\boldsymbol{R}, t) = 0. \tag{3.26}$$

Under this condition, the Fokker-Planck equation simplifies to

$$\boldsymbol{\nabla}^2 P(\boldsymbol{R}) = \boldsymbol{\nabla} \cdot \boldsymbol{F}(\boldsymbol{R}) P(\boldsymbol{R}) = \boldsymbol{F}(\boldsymbol{R}) \cdot \boldsymbol{\nabla} P(\boldsymbol{R}) + P(\boldsymbol{R}) \boldsymbol{\nabla} \cdot \boldsymbol{F}(\boldsymbol{R}) \tag{3.27}$$

For the Laplacian to appear on the right-hand side, the drift force should have the form

$$\boldsymbol{F}(\boldsymbol{R}) = f(P(\boldsymbol{R})) \boldsymbol{\nabla} P(\boldsymbol{R}), \tag{3.28}$$

where f is a scalar function to be determined. Then, Eq. (3.25) can be written as

$$\begin{aligned} \boldsymbol{\nabla}^2 P(\boldsymbol{R}) = {} & f(P(\boldsymbol{R})) \boldsymbol{\nabla} P(\boldsymbol{R}) \cdot \boldsymbol{\nabla} P(\boldsymbol{R}) \\ & + P(\boldsymbol{R}) f(P(\boldsymbol{R})) \boldsymbol{\nabla}^2 P(\boldsymbol{R}) \\ & + P(\boldsymbol{R}) \boldsymbol{\nabla} f(P(\boldsymbol{R})) \cdot \boldsymbol{\nabla} P(\boldsymbol{R}). \end{aligned} \tag{3.29}$$

Matching the Laplacian terms, we find that

$$f(P(\boldsymbol{R})) = \frac{1}{P(\boldsymbol{R})}, \tag{3.30}$$

which also eliminates the gradient terms. Therefore, the drift velocity must have the form

$$\boldsymbol{F}(\boldsymbol{R}) = \frac{1}{P(\boldsymbol{R})}\boldsymbol{\nabla}P(\boldsymbol{R}) = 2\frac{1}{\Psi(\boldsymbol{R})}\boldsymbol{\nabla}\Psi(\boldsymbol{R}), \tag{3.31}$$

where we have assumed $\Psi(\boldsymbol{R})$ is real-valued in the last equality.

The Fokker-Planck equation can be seen as a deterministic description of the stochastic dynamics captured by the corresponding Langevin equation,

$$\frac{\partial}{\partial t}\boldsymbol{R}(t) = D\boldsymbol{F}(\boldsymbol{R}) + \boldsymbol{\xi}, \tag{3.32}$$

a stochastic differential equation that describes the dynamics of a particle under the influence of both deterministic forces and random noise. In the above, $\boldsymbol{\xi} \in \mathbb{R}^{Nd}$ is a random perturbation drawn from a Gaussian distribution $\mathcal{N}(\boldsymbol{0}, 2D\boldsymbol{I})$ and $\boldsymbol{F}(\boldsymbol{R})$ is the drift force evaluated at the initial configuration, defined in Eq. (3.31). Integrating the Langevin equation over a short time interval Δt, we obtain

$$\boldsymbol{R}(t+\Delta t) = \boldsymbol{R}(t) + D\Delta t\boldsymbol{F}(\boldsymbol{R}) + \sqrt{\Delta t}\boldsymbol{\xi}. \tag{3.33}$$

This proposal rule implies the normalized proposal density is given by

$$\begin{aligned} Q(\boldsymbol{R}'|\boldsymbol{R}, \Delta t) &= \mathcal{N}(\boldsymbol{R} + D\Delta t\boldsymbol{F}(\boldsymbol{R}), 2D\delta t\boldsymbol{I}) \\ &= \frac{1}{(4\pi D\Delta t)^{Nd/2}} e^{-(\boldsymbol{R}'-\boldsymbol{R}-D\Delta t\boldsymbol{F}(\boldsymbol{R}))^2/4D\Delta t}, \end{aligned} \tag{3.34}$$

which is no longer symmetric due to the drift term. Then the acceptance probability becomes

$$\begin{aligned} A(\boldsymbol{R}'|\boldsymbol{R},\Delta t) &= \min\left\{1, \frac{P(\boldsymbol{R}')Q(\boldsymbol{R}|\boldsymbol{R}',\Delta t)}{P(\boldsymbol{R})Q(\boldsymbol{R}'|\boldsymbol{R},\Delta t)}\right\} \\ &= \min\left\{1, \frac{|\Psi(\boldsymbol{R}',\boldsymbol{S})|^2}{|\Psi(\boldsymbol{R},\boldsymbol{S})|^2}\frac{e^{-(\boldsymbol{R}-\boldsymbol{R}'-D\Delta t\boldsymbol{F}(\boldsymbol{R}'))^2/4D\Delta t}}{e^{-(\boldsymbol{R}'-\boldsymbol{R}-D\Delta t\boldsymbol{F}(\boldsymbol{R}))^2/4D\Delta t}}\right\}. \end{aligned} \tag{3.35}$$

The additional weight appearing next to the original acceptance probability can be easily simplified by defining the drift velocity $\boldsymbol{v}(\boldsymbol{R}) \equiv \boldsymbol{F}(\boldsymbol{R})/2$,

$$\frac{e^{-(\boldsymbol{R}-\boldsymbol{R}'-D\Delta t\boldsymbol{F}(\boldsymbol{R}'))^2/4D\Delta t}}{e^{-(\boldsymbol{R}'-\boldsymbol{R}-D\Delta t\boldsymbol{F}(\boldsymbol{R}))^2/4D\Delta t}} = e^{-\Big(\boldsymbol{v}(\boldsymbol{R})-\boldsymbol{v}(\boldsymbol{R}')\Big)\Big((\boldsymbol{R}-\boldsymbol{R}')+D\Delta t\Big(\boldsymbol{v}(\boldsymbol{R})-\boldsymbol{v}(\boldsymbol{R}')\Big)\Big)} \tag{3.36}$$

This weight corrects for the sampling bias introduced by our proposal distribution exploring more relevant regions of the probability distribution. An equivalent derivation can be achieved by integrating the Fokker-Planck equation itself for a small time step Δt and using the Green's function method to solve for the time-dependent probability distribution. The resulting Green's function coincides with the proposal distribution in Eq. (3.34).

3.1.2 Integration

Monte Carlo integration is a numerical technique that uses random sampling to efficiently estimate the value of a high-dimensional integral. Recall that we can expand the expectation value of operators $\hat{A}$ in our $|\boldsymbol{X}\rangle$ basis as

$$\langle\hat{A}\rangle = \frac{\langle\Psi|\hat{A}|\Psi\rangle}{\langle\Psi|\Psi\rangle} = \frac{\int d\boldsymbol{X}\langle\Psi|\boldsymbol{X}\rangle\langle\boldsymbol{X}|\hat{A}|\Psi\rangle}{\int d\boldsymbol{X}\langle\Psi|\boldsymbol{X}\rangle\langle\boldsymbol{X}|\Psi\rangle}, \tag{3.37}$$

Inserting $1 = \langle \boldsymbol{X}|\Psi\rangle/\langle \boldsymbol{X}|\Psi\rangle$ into the numerator, we have

$$\langle \hat{A}\rangle = \frac{\int d\boldsymbol{X}|\Psi(\boldsymbol{X})|^2 A(\boldsymbol{X})}{\int d\boldsymbol{X}|\Psi(\boldsymbol{X})|^2}, \tag{3.38}$$

where $A(\boldsymbol{X})$ is the local quantity corresponding to the operator $\hat{A}$

$$A(\boldsymbol{X}) \equiv \frac{\langle \boldsymbol{X}|\hat{A}|\Psi\rangle}{\langle \boldsymbol{X}|\Psi\rangle}. \tag{3.39}$$

Notice that

$$P(\boldsymbol{X}) = \frac{|\Psi(\boldsymbol{X})|^2}{\int dX|\Psi(\boldsymbol{X})|^2} \tag{3.40}$$

is none other than the normalized probability distribution given by our wave function from Eq. (3.1). Then the expectation value becomes

$$\langle \hat{A}\rangle = \int d\boldsymbol{X} P(\boldsymbol{X}) A(\boldsymbol{X}), \tag{3.41}$$

where the integral over $\boldsymbol{X}$ can be appropriately decomposed into a sum over $\boldsymbol{S}$ and an integral over $\boldsymbol{R}$ whenever necessary. Any integral with this form can be approximated as a simple average over local quantities

$$\langle \hat{A}\rangle \approx \frac{1}{T}\sum_{t=1}^{T} A(\boldsymbol{X}_t) \equiv \langle A(\boldsymbol{X})\rangle, \tag{3.42}$$

where the average is weighted over T samples from the probability distribution $\boldsymbol{X}_t \sim P(\boldsymbol{X})$. By the law of large numbers, the average of the local quantities $\langle A(\boldsymbol{X})\rangle$ will converge to the expected value of the operator $\langle \hat{A}\rangle$ as T increases. Notice the notation we have chosen for the expectation value of the operator versus the weighted sum over the local quantities, as it

differs from other texts. We express both with angled brackets, but the explicit dependence on $\boldsymbol{X}$ for the latter is meant to imply $\boldsymbol{X}$ is pulled from our probability distribution. We choose this notation $\langle A(\boldsymbol{X})\rangle$ over the more common $\langle A_L\rangle$ to leave room for other subscripts.

3.2 Variational Monte Carlo

Variational Monte Carlo (VMC) is a direct application of the variational principle from Sec. 2.4 and Monte Carlo integration from Sec. 3.1.2. The core idea is to take some parameterized trial wave function $\Psi_{\boldsymbol{\theta}}(\boldsymbol{X})$, and minimize the corresponding expectation value of the energy with respect to the variational parameters $\boldsymbol{\theta}$

$$E[\Psi_{\boldsymbol{\theta}}] \equiv \frac{\langle\Psi_{\boldsymbol{\theta}}|\hat{H}|\Psi_{\boldsymbol{\theta}}\rangle}{\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle}, \tag{3.43}$$

$$\min_{\boldsymbol{\theta}} E[\Psi_{\boldsymbol{\theta}}] \geq E_0. \tag{3.44}$$

The variational principle guarantees that the variational energy $E[\Psi_{\boldsymbol{\theta}}]$ establishes a strict upperbound on the true ground state energy of the system. To find the optimal parameters, we use gradient descent methods, a family of iterative optimization algorithms discussed in more detail in Sec. 3.2.2. For now, we just compute the gradient of the variational energy with respect to the parameters

$$\begin{aligned}\boldsymbol{\nabla}_{\boldsymbol{\theta}} E[\Psi_{\boldsymbol{\theta}}] &= \frac{1}{\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle}\Big(\langle\boldsymbol{\nabla}_{\boldsymbol{\theta}}\Psi_{\boldsymbol{\theta}}|\hat{H}|\Psi_{\boldsymbol{\theta}}\rangle + \langle\Psi_{\boldsymbol{\theta}}|\hat{H}|\boldsymbol{\nabla}_{\boldsymbol{\theta}}\Psi_{\boldsymbol{\theta}}\rangle\Big) \\ &\quad - \frac{\langle\Psi_{\boldsymbol{\theta}}|\hat{H}|\Psi_{\boldsymbol{\theta}}\rangle}{\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle^2}\Big(\langle\boldsymbol{\nabla}_{\boldsymbol{\theta}}\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle + \langle\Psi_{\boldsymbol{\theta}}|\boldsymbol{\nabla}_{\boldsymbol{\theta}}\Psi_{\boldsymbol{\theta}}\rangle\Big) \\ &= 2\left(\frac{\langle\Psi_{\boldsymbol{\theta}}|\hat{H}|\boldsymbol{\nabla}_{\boldsymbol{\theta}}\Psi_{\boldsymbol{\theta}}\rangle}{\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle} - E[\Psi_{\boldsymbol{\theta}}]\frac{\langle\Psi_{\boldsymbol{\theta}}|\boldsymbol{\nabla}_{\boldsymbol{\theta}}\Psi_{\boldsymbol{\theta}}\rangle}{\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle}\right),\end{aligned} \tag{3.45}$$

where we have assumed our trial wave function is real-valued in the last equality. To approximate all the high-dimensional integrals above, we employ Monte Carlo integration. For the variational energy, we simply write

$$E[\Psi_{\boldsymbol{\theta}}] \approx \frac{1}{T}\sum_{t=1}^{T} E_{\boldsymbol{\theta}}(\boldsymbol{X}_t) \equiv \langle E_{\boldsymbol{\theta}}(\boldsymbol{X})\rangle, \tag{3.46}$$

where the configurations $\boldsymbol{X}_t$ are sampled from $|\Psi_{\boldsymbol{\theta}}(\boldsymbol{X})|^2$, T is the number of samples, and the local energy is defined as

$$E_{\boldsymbol{\theta}}(\boldsymbol{X}) \equiv \frac{\langle \boldsymbol{X}|\hat{H}|\Psi_{\boldsymbol{\theta}}\rangle}{\langle \boldsymbol{X}|\Psi_{\boldsymbol{\theta}}\rangle} = \frac{\hat{H}\Psi_{\boldsymbol{\theta}}(\boldsymbol{X})}{\Psi_{\boldsymbol{\theta}}(\boldsymbol{X})}. \tag{3.47}$$

For the gradient in Eq. (3.45), we first define the local gradient of the wave function as

$$\boldsymbol{O}_{\boldsymbol{\theta}}(\boldsymbol{X}) \equiv \frac{\langle \boldsymbol{X}|\boldsymbol{\nabla}_{\boldsymbol{\theta}}\Psi_{\boldsymbol{\theta}}\rangle}{\langle \boldsymbol{X}|\Psi_{\boldsymbol{\theta}}\rangle} = \frac{\boldsymbol{\nabla}_{\boldsymbol{\theta}}\Psi_{\boldsymbol{\theta}}(\boldsymbol{X})}{\Psi_{\boldsymbol{\theta}}(\boldsymbol{X})} = \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \Psi_{\boldsymbol{\theta}}(\boldsymbol{X}), \tag{3.48}$$

which allows us to write

$$\begin{aligned} \frac{\langle \Psi_{\boldsymbol{\theta}}|\boldsymbol{\nabla}_{\boldsymbol{\theta}}\Psi_{\boldsymbol{\theta}}\rangle}{\langle \Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle} &= \frac{\int d\boldsymbol{X}\langle \Psi_{\boldsymbol{\theta}}|\boldsymbol{X}\rangle\langle \boldsymbol{X}|\boldsymbol{\nabla}_{\boldsymbol{\theta}}\Psi_{\boldsymbol{\theta}}\rangle}{\int d\boldsymbol{X}\langle \Psi_{\boldsymbol{\theta}}|\boldsymbol{X}\rangle\langle \boldsymbol{X}|\Psi_{\boldsymbol{\theta}}\rangle} = \frac{\int d\boldsymbol{X}|\Psi_{\boldsymbol{\theta}}(\boldsymbol{X})|^2 \boldsymbol{O}_{\boldsymbol{\theta}}(\boldsymbol{X})}{\int d\boldsymbol{X}|\Psi_{\boldsymbol{\theta}}(\boldsymbol{X})|^2} \\ &\approx \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{O}_{\boldsymbol{\theta}}(\boldsymbol{X}_t) \equiv \langle \boldsymbol{O}_{\boldsymbol{\theta}}(\boldsymbol{X})\rangle, \end{aligned} \tag{3.49}$$

$$\begin{aligned} \frac{\langle \Psi_{\boldsymbol{\theta}}|\hat{H}|\boldsymbol{\nabla}_{\boldsymbol{\theta}}\Psi_{\boldsymbol{\theta}}\rangle}{\langle \Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle} &= \frac{\int d\boldsymbol{X}\langle \Psi_{\boldsymbol{\theta}}|\hat{H}|\boldsymbol{X}\rangle\langle \boldsymbol{X}|\boldsymbol{\nabla}_{\boldsymbol{\theta}}\Psi_{\boldsymbol{\theta}}\rangle}{\int d\boldsymbol{X}\langle \Psi_{\boldsymbol{\theta}}|\boldsymbol{X}\rangle\langle \boldsymbol{X}|\Psi_{\boldsymbol{\theta}}\rangle} = \frac{\int d\boldsymbol{X}|\Psi_{\boldsymbol{\theta}}(\boldsymbol{X})|^2 E_{\boldsymbol{\theta}}(\boldsymbol{X})\boldsymbol{O}_{\boldsymbol{\theta}}(\boldsymbol{X})}{\int d\boldsymbol{X}|\Psi_{\boldsymbol{\theta}}(\boldsymbol{X})|^2} \\ &\approx \frac{1}{T}\sum_{t=1}^{T} E_{\boldsymbol{\theta}}(\boldsymbol{X}_t)\boldsymbol{O}_{\boldsymbol{\theta}}(\boldsymbol{X}_t) \equiv \langle E_{\boldsymbol{\theta}}(\boldsymbol{X})\boldsymbol{O}_{\boldsymbol{\theta}}(\boldsymbol{X})\rangle, \end{aligned} \tag{3.50}$$

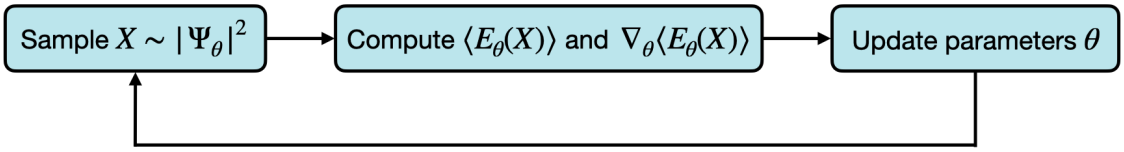

Figure 3.1: The cyclic workflow of the variational Monte Carlo algorithm. Starting with a random set of variational parameters, the cycle is repeated until the variational energy converges.

where we have once again assumed the wave function is real. Therefore, the gradient can be approximated as

$$\boldsymbol{\nabla_\theta} E[\Psi_{\boldsymbol{\theta}}] \approx \boldsymbol{\nabla_\theta}\langle E_{\boldsymbol{\theta}}(\boldsymbol{X})\rangle = 2\Big(\langle E_{\boldsymbol{\theta}}(\boldsymbol{X})\boldsymbol{O_\theta}(\boldsymbol{X})\rangle - \langle E_{\boldsymbol{\theta}}(\boldsymbol{X})\rangle\langle \boldsymbol{O_\theta}(\boldsymbol{X})\rangle\Big). \tag{3.51}$$

The variational Monte Carlo algorithm is a simple cyclic procedure; we sample configurations from the square of the wave function, estimate the variational energy and its gradient, and update the parameters. There are many ways we can complete the last step of this cycle, which we will cover in Sec. 3.2.2. More information about the implementation will be discussed in Chapter 5.

3.2.1 Trial Wave Functions

The choice of trial wave function, or ansatz, is the most crucial ingredient of a variational Monte Carlo calculation. If chosen incorrectly, the upperbound on the ground state energy provided by the variational principle is effectively meaningless. For this reason, traditional VMC calculations rely on carefully curating a trial wave function to capture the essential features and correlations of the system under study. This involves considering the symmetries, boundary conditions, and known properties of the system, as well as incorporating relevant physical insights and intuition. Consequently, these trial wave functions are intricately tailored to the specific Hamiltonians, making them ill-suited for

application to similar, albeit distinct, Hamiltonians without substantial structural modifications.

In this section, we will showcase a diverse range of conventional trial wave functions and analyze their suitability for different types of systems. Our objective is to identify the limitations of these methods while appreciating their inherent strengths. Through this exploration, we aim to develop a deeper understanding of the advantages offered by neural networks and their potential to overcome the limitations commonly associated with traditional strategies.

3.2.1.1 Kato's Cusp Condition

In the presence of singularities in the potential $V(\boldsymbol{X})$, Kato's cusp condition asserts that the local kinetic energy $K_{\boldsymbol{\theta}}(\boldsymbol{X})$ must precisely counterbalance the diverging potential such that the total local energy remains finite as any two particles approach each other. More formally, the cusp condition states

$$\lim_{r_{ij}\to 0} E_{\boldsymbol{\theta}}(\boldsymbol{X}) = \lim_{r_{ij}\to 0} \left(K_{\boldsymbol{\theta}}(\boldsymbol{X}) + V(\boldsymbol{X})\right) < \infty, \tag{3.52}$$

for all pairs i, j, where $r_{ij} = |\boldsymbol{r}_i - \boldsymbol{r}_j|$ is defined as the Euclidean distance between particles i and j, and the local kinetic energy is given by

$$K_{\boldsymbol{\theta}}(\boldsymbol{X}) = \sum_{i=1}^{N} -\frac{\hbar^2}{2m} \frac{\boldsymbol{\nabla}_i^2 \Psi_{\boldsymbol{\theta}}(\boldsymbol{X})}{\Psi_{\boldsymbol{\theta}}(\boldsymbol{X})}. \tag{3.53}$$

For strongly-correlated systems, it is especially important to design a trial wave function that upholds the cusp condition throughout training. Even a finite, but hard interaction potential may warrant a cusp condition to ensure stability throughout the optimization of

$\Psi_{\boldsymbol{\theta}}(\boldsymbol{X})$. A "hard" potential is often used to describe an interaction potential that exhibits abrupt changes in magnitude as the distance between particles varies. These potentials typically have a large scattering length while being short-range in nature.

3.2.1.2 Jastrow Factor

Enforcement of Kato's cusp condition usually comes in the form of a Jastrow factor, also known as a Jastrow wave function. For a system of identical particles, it can be any permutation-invariant function of the particles, as it preserves the symmetry of the overall wave function. The goal of the Jastrow factor is to incorporate the effects of particle-particle interactions into the trial wave function. The form of the Jastrow factor depends on the specific system and the character of the interactions being considered.

For a system of bosons, the ground state wave function must be positive-definite and symmetric with respect to particle exchange. Since we can ignore spin, the generic trial wave function for bosons can be written as

$$\Psi_{\boldsymbol{\theta}}(\boldsymbol{R}) = \Phi_0(\boldsymbol{R}) e^{J(\boldsymbol{R})}, \tag{3.54}$$

where the first term represents the ground state of the non-interacting problem and the second term is the symmetric Jastrow factor containing the inter-particle correlations. By writing

$$\Phi_0(\boldsymbol{R}) = \prod_{i=1}^{N} \xi_0(\boldsymbol{r}_i), \tag{3.55}$$

the non-interacting problem becomes simple to solve because it can be separated into N

identical and independent parts

$$\begin{aligned}\left(\hat{K} + V_1(\boldsymbol{R})\right)\Phi_0(\boldsymbol{R}) &= E_0^{\text{nonint}}\Phi_0(\boldsymbol{R}),\\ \left(-\frac{\hbar^2}{2m}\boldsymbol{\nabla}_i^2 + v_1(\boldsymbol{r}_i)\right)\xi_0(\boldsymbol{r}_i) &= \varepsilon_0\xi_0(\boldsymbol{r}_i),\\ E_0^{\text{nonint}} &= \sum_{i=1}^{N}\varepsilon_0 = N\varepsilon_0,\end{aligned} \tag{3.56}$$

which can often be solved analytically. Each of the N bosons occupy the lowest single-particle state with energy ε_0, so the non-interacting energy is simply their sum.

Examples of exactly-solvable non-interacting systems include the free system

$$\hat{H}^{\text{free}} = \hat{K}, \tag{3.57}$$

where the single-particle ground state is the trivial plane-wave orbital

$$\psi_0^{\text{free}}(\boldsymbol{r}_i) = 1, \tag{3.58}$$

with zero energy $\varepsilon_0^{\text{free}} = 0$, in which case, the trial wave function is just the Jastrow factor on its own. Another common example is the non-interacting system of isotropic harmonic oscillators

$$\hat{H}^{\text{ho}} = \hat{K} + \sum_{i=1}^{N}\frac{1}{2}m\omega^2\boldsymbol{r}_i^2, \tag{3.59}$$

where ω is the oscillation frequency. The single-particle ground state wave function is a Gaussian

$$\psi_0^{\text{ho}}(\boldsymbol{r}_i) = e^{-\frac{m\omega}{2\hbar}\boldsymbol{r}_i^2} \tag{3.60}$$

with the corresponding energy

$$\varepsilon_0^{\text{ho}} = \frac{d}{2}\hbar\omega. \tag{3.61}$$

Constructing the Jastrow factor is often much more difficult than the non-interacting ground state. Nonetheless, there are some exactly-solvable interactions that serve as great examples for demonstrating the general strategy. Consider the Calogero-Sutherland model of bosons in a one-dimensional harmonic oscillator trap interacting with an inverse-squared potential

$$\hat{H}^{\text{cs}} = \hat{H}^{\text{ho}} + V^{\text{cs}}(\boldsymbol{R}), \quad V^{\text{cs}}(\boldsymbol{R}) = \frac{\hbar^2}{m}\sum_{i<j}^{N}\frac{\beta(\beta-1)}{x_{ij}^2}, \tag{3.62}$$

where $x_{ij} = |x_i - x_j|$ is the distance between the pair i and j, and β is an interaction parameter. Say we look for an ansatz with the form

$$\Psi_{\boldsymbol{\theta}}^{\text{cs}}(\boldsymbol{R}) = \Phi_0^{\text{ho}}(\boldsymbol{R})e^{J(\boldsymbol{R})}, \tag{3.63}$$

where $\Phi_0^{\text{ho}}(\boldsymbol{R}) \equiv \prod_{i=1}^{N}\xi_0^{\text{ho}}$ is the ground state of the one-dimensional non-interacting harmonic oscillators. Then according to Kato's cusp condition, we need to find the function $J(\boldsymbol{R})$ that satisfies

$$\lim_{x_{ij}\to 0} E_L(\boldsymbol{R}) < \infty, \tag{3.64}$$

for all configurations $\boldsymbol{R} = (x_1, x_2, ..., x_N)$ and pairs i, j. It is straightforward to show that

$$\frac{1}{\Psi_{\boldsymbol{\theta}}}\frac{\partial^2}{\partial x_i^2}\Psi_{\boldsymbol{\theta}} = \frac{1}{\Phi_0^{\text{ho}}}\frac{\partial^2\Phi_0^{\text{ho}}}{\partial x_i^2} + \frac{2}{\Phi_0^{\text{ho}}}\frac{\partial\Phi_0^{\text{ho}}}{\partial x_i}\frac{\partial J(\boldsymbol{R})}{\partial x_i} + \frac{\partial^2 J(\boldsymbol{R})}{\partial x_i^2} + \left(\frac{\partial J(\boldsymbol{R})}{\partial x_i}\right)^2, \tag{3.65}$$

for any particular particle i. Since the ground state for the non-interacting system is $\Phi_0^{\text{ho}} =$

$\prod_{i=1}^{N} e^{-\frac{m\omega}{2\hbar}x_i}$ in one dimension, the local kinetic energy then evaluates to

$$K_L(\boldsymbol{R}) = K_L^{\text{ho}}(\boldsymbol{R}) + \frac{\hbar^2}{2m}\sum_{i=1}^{N}\left[\frac{m\omega}{\hbar}x_i\frac{\partial J(\boldsymbol{R})}{\partial x_i} - \frac{\partial^2 J(\boldsymbol{R})}{\partial x_i^2} - \left(\frac{\partial J(\boldsymbol{R})}{\partial x_i}\right)^2\right], \tag{3.66}$$

where $K_L^{\text{ho}}(\boldsymbol{R}) = \frac{1}{\Phi_0^{\text{ho}}}\hat{K}\Phi_0^{\text{ho}}$ is the local kinetic energy for the non-interacting harmonic oscillators. In order to cancel the divergences coming from the interaction potential in Eq. (3.62), the right-hand side must scale as $\sim \frac{1}{x_{ij}^2}$ as $x_{ij} \to \infty$ for all pairs. The simplest Jastrow correlator with the proper scaling behavior is

$$J(\boldsymbol{R}) = b\sum_{i<j}\log x_{ij}, \tag{3.67}$$

for some constant b. By matching the coefficients in Eq. (3.66), we find that $b = \beta$, and the resulting ansatz is the exact solution for the ground state

$$\Psi_0^{\text{cs}} = \Phi_0^{ho}\prod_{i<j}^{N} x_{ij}^{\beta}, \tag{3.68}$$

with the corresponding energy

$$E_0^{\text{cs}} = E_0^{ho} + \frac{1}{2}\beta N(N-1)\hbar\omega. \tag{3.69}$$

In this particular example, we were fortunate to uncover an exact solution, making the choice of a parameterized wave function an obvious one. Specifically, we can use the Jastrow factor as defined in Eq. (3.67), with a single variational parameter denoted as b. However, by the time we have found the optimal design for $J(\boldsymbol{R})$, we essentially solved the entire problem, rendering the application of variational Monte Carlo pointless except for

benchmarking purposes.

Although the cusp condition can be solved exactly in the aforementioned Calogero-Sutherland model, this is not the case for the vast majority of interesting systems. Furthermore, determining the appropriate form of the Jastrow factor necessitated a specific choice for the interaction potential. This means that the knowledge we gained throughout this process is not immediately applicable for other problems. We will see that when utilizing neural networks as trial wave functions, it is possible to forgo manually enforcing the cusp condition, as the neural networks can autonomously discover it, allowing a single ansatz to be reused for a large class of problems.

3.2.1.3 Slater Determinant

For systems of fermions, the situation becomes more complicated because the ground state wave function must be antisymmetric with respect to particle exchange, and we have to be mindful of spins. We can still use a symmetric Jastrow factor to capture the correlations in our trial wave function, but we need to replace the symmetric non-interacting ground state in Eq. (3.54) with an antisymmetric one. Hence, we reintroduce the spins and take our ansatz to be

$$\Psi_{\boldsymbol{\theta}}(\boldsymbol{X}) = \Phi_0(\boldsymbol{X})e^{J(\boldsymbol{X})}, \tag{3.70}$$

where $\Phi_0(\boldsymbol{X})$ is a Slater determinant involving the N single-particle spin-orbitals evaluated for each of the N particles

$$\Phi_0(\boldsymbol{X}) = \det\Big[\varphi_\alpha(\boldsymbol{x}_i)\Big] = \det\begin{bmatrix} \varphi_1(\boldsymbol{x}_1) & \varphi_1(\boldsymbol{x}_2) & \cdots & \varphi_1(\boldsymbol{x}_N) \\ \varphi_2(\boldsymbol{x}_1) & \varphi_2(\boldsymbol{x}_2) & \cdots & \varphi_2(\boldsymbol{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_N(\boldsymbol{x}_1) & \varphi_N(\boldsymbol{x}_2) & \cdots & \varphi_N(\boldsymbol{x}_N) \end{bmatrix} \tag{3.71}$$

The quantum numbers of the single-particle states as labeled generically by α, while the particles are labeled with the index i. As with bosons, we obtain the single-particle states by solving the non-interacting problem, but because of the Pauli exclusion principle, we must find the spectrum rather than just the ground state. The fermions are filled in a way that minimizes the total energy of the system, starting from the lowest energy state and moving upwards. This filling continues until the highest-occupied state, often referred to as the Fermi level, is reached. The Fermi level represents the boundary between filled and unfilled states in a system of fermions, and its corresponding single-particle energy is denoted as ε_F.

As another simple example, let us consider the circular quantum-dots system,

$$\hat{H}^{\mathrm{qd}} = \hat{H}^{\mathrm{ho}} + V^{\mathrm{qd}}(\boldsymbol{R}), \quad V^{\mathrm{qd}}(\boldsymbol{R}) = \frac{e^2}{4\pi\varepsilon_0}\sum_{i<j}\frac{1}{r_{ij}}, \tag{3.72}$$

a two-dimensional system of electrons trapped in a harmonic oscillator well. For convenience, we set $e = 4\pi\varepsilon_0 = 1$ and assume the system is unpolarized, meaning there is an equal number of spin-up and spin-down electrons, $N_\uparrow = N_\downarrow = N/2$. Then in two-dimensions, the single-

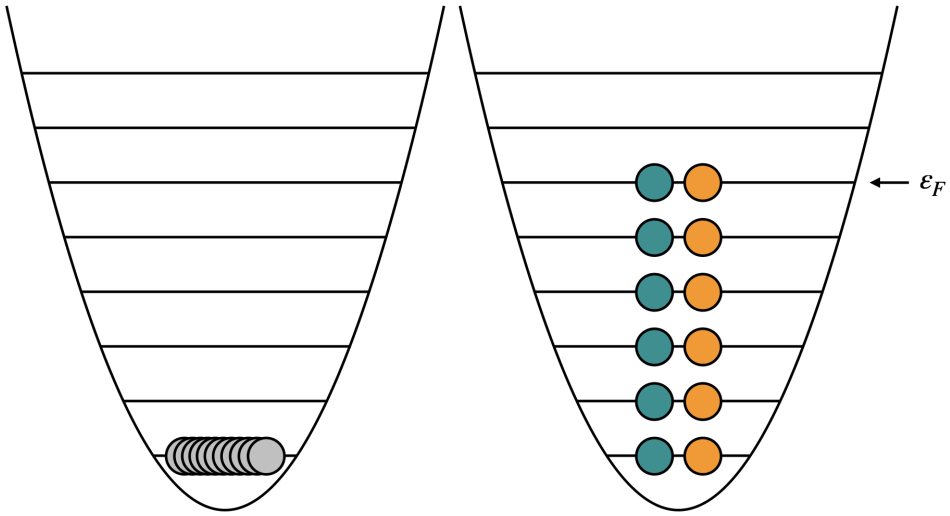

Figure 3.2: The occupancy of single-particle states for bosons (left) and fermions (right) at zero temperature. The two different colors (blue/orange) of fermions represent the two possible spin states (up/down). While bosons can occupy the same single-particle state, fermions are restricted to distinct states due to the Pauli exclusion principle. The energy of the highest occupied single-particle state is commonly referred to as the Fermi energy ε_F.

particle wave functions for the non-interacting problem are

$$\varphi_\alpha(\boldsymbol{r}_i) = e^{-\frac{m\omega}{2\hbar}\boldsymbol{r}_i^2} H_{m_\alpha}\left(\sqrt{\frac{m\omega}{\hbar}}x_i\right) H_{n_\alpha}\left(\sqrt{\frac{m\omega}{\hbar}}y_i\right) \langle s_i|\sigma_\alpha\rangle, \tag{3.73}$$

where $i = (\boldsymbol{r}_i, s_i) = (x_i, y_i, s_i)$ labels the single-particle degrees of freedom, $\alpha = (m_\alpha, n_\alpha, \sigma_\alpha)$ labels the single-particle states with $\sigma_\alpha \in \{\uparrow, \downarrow\}$, and $H_n(x)$ are the Hermite polynomials of degree n. By inserting the above expression into Eq. (3.71), we can automatically constrain the antisymmetry of the wave function.

3.2.1.4 Backflow Transformations

The Slater determinant discussed in the preceding section relies on single-particle orbitals derived from the non-interacting problem. Consequently, the nodal structure of the overall wave function remains fixed during the training process, as the Slater determinant contains

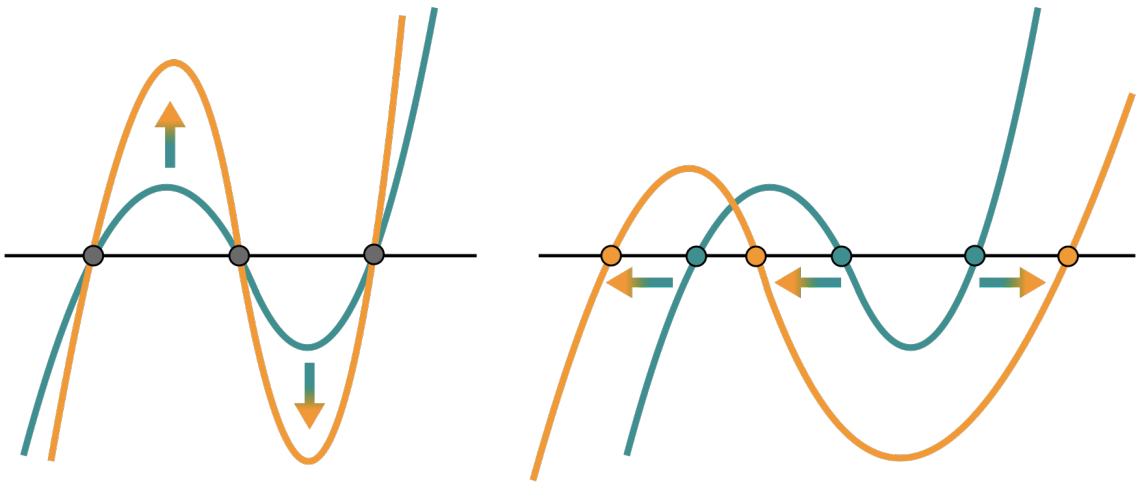

Figure 3.3: The effects of a symmetric Jastrow factor (left) and a backflow transformation (right) on the nodes of a fermionic wave function. While both the Jastrow factor and the backflow transformation maintain the antisymmetry of the overall wave function and have the ability to modify the wave function's amplitude, only the backflow transformation is capable of shifting the positions of the nodes.

no variational parameters. To improve the nodal structure of the Slater determinant, we can incorporate backflow transformations, which aim to capture the influence of all surrounding particles on the state of an individual particle. While Jastrow factors technically encode some backflow correlations as well, we only utilize positive definite Jastrows that are incapable of changing the nodal structure. For the sake of clarity, we will specifically use the term "backflow transformation" for the technique that facilitates alterations to the nodal structure of the Slater determinant.

A backflow transformation typically alters the single-particle positions as

$$\boldsymbol{r}_i \mapsto \boldsymbol{r}_i + \sum_{j \neq i}^{N} \eta(r_{ij})(\boldsymbol{r}_i - \boldsymbol{r}_j), \tag{3.74}$$

where $\eta(r)$ is the backflow correlation function. Then the transformed coordinates, along with the untransformed spin, serve as input to the non-interacting spin-orbitals $\varphi_\alpha(\boldsymbol{x}_i)$. Determining the functional form of $\eta(r)$, much like the Jastrow factor, necessitates the imposition of known physical insights about the system. However, accomplishing this task manually can be quite challenging in general.

Alternate forms for the backflow transformation can be chosen, as long as the transformation is permutation equivariant, i.e. for a given particle i, the transformation is invariant under any permutation of all the other particles $j \neq i$. With this constraint, the antisymmetry of the Slater determinant is preserved. We write this arbitrary transformation as

$$\boldsymbol{x}_i \mapsto \boldsymbol{\eta}(\boldsymbol{x}_i, \{\boldsymbol{x}_{j\neq i}\}), \tag{3.75}$$

which suggests a means to potentially incorporate the spins of all particles into the backflow transformations, in addition to the spatial degrees of freedom.

3.2.1.5 Bardeen-Cooper-Schrieffer Wave Function

In fermionic systems with strong pairing correlations, it is often insufficient to construct an antisymmetric ansatz from a Slater determinant of single-particle states. Even with the utilization of multiple Slater determinants or the incorporation of backflow transformations, the antisymmetric component of the wave function remains constrained within the single-particle mean-field framework. Thus, we graduate to the number-projected Bardeen-Cooper-Schrieffer (BCS) wave function, an antisymmetrized product of singlet-pair orbitals. This ansatz is suitable for unpolarized systems of fermions with an attractive interaction sufficiently strong enough in the s-wave channel to facilitate

the formation of Cooper pairs. It can be written as the determinant of an $N/2 \times N/2$ matrix,

$$\Phi_0(\boldsymbol{X}) = \det\left[\xi(\boldsymbol{r}_i^{\uparrow}, \boldsymbol{r}_j^{\downarrow})\right] = \det\begin{bmatrix} \xi(\boldsymbol{r}_1^{\uparrow}, \boldsymbol{r}_1^{\downarrow}) & \xi(\boldsymbol{r}_1^{\uparrow}, \boldsymbol{r}_2^{\downarrow}) & \cdots & \xi(\boldsymbol{r}_1^{\uparrow}, \boldsymbol{r}_{N/2}^{\downarrow}) \\ \xi(\boldsymbol{r}_2^{\uparrow}, \boldsymbol{r}_1^{\downarrow}) & \xi(\boldsymbol{r}_2^{\uparrow}, \boldsymbol{r}_2^{\downarrow}) & \cdots & \xi(\boldsymbol{r}_2^{\uparrow}, \boldsymbol{r}_{N/2}^{\downarrow}) \\ \vdots & \vdots & \ddots & \vdots \\ \xi(\boldsymbol{r}_{N/2}^{\uparrow}, \boldsymbol{r}_1^{\downarrow}) & \xi(\boldsymbol{r}_{N/2}^{\uparrow}, \boldsymbol{r}_2^{\downarrow}) & \cdots & \xi(\boldsymbol{r}_{N/2}^{\uparrow}, \boldsymbol{r}_{N/2}^{\downarrow}) \end{bmatrix}, \quad (3.76)$$

where $\xi(\boldsymbol{r}_i^{\uparrow}, \boldsymbol{r}_j^{\downarrow})$ is the unique singlet-pairing orbital evaluated for each combination of spin-up and spin-down fermions. It has been shown that the above BCS wave function can be expanded to include single-particle orbitals $\varphi_\alpha(\boldsymbol{r})$ and remain antisymmetric. Define $P = \min\{N_\uparrow, N_\downarrow\}$ as the number of pairs and $U = |N_\uparrow - N_\downarrow|$ as the number of unpaired particles. Then the augmented BCS wave function becomes the determinant of a $(P+U) \times (P+U)$ matrix,

$$\Phi_0(\boldsymbol{X}) = \det\begin{bmatrix} \xi(\boldsymbol{r}_1^{\uparrow}, \boldsymbol{r}_1^{\downarrow}) & \xi(\boldsymbol{r}_1^{\uparrow}, \boldsymbol{r}_2^{\downarrow}) & \cdots & \xi(\boldsymbol{r}_1^{\uparrow}, \boldsymbol{r}_P^{\downarrow}) & \varphi_1(\boldsymbol{r}_1^{\uparrow}) & \varphi_2(\boldsymbol{r}_1^{\uparrow}) & \cdots & \varphi_U(\boldsymbol{r}_1^{\uparrow}) \\ \xi(\boldsymbol{r}_2^{\uparrow}, \boldsymbol{r}_1^{\downarrow}) & \xi(\boldsymbol{r}_2^{\uparrow}, \boldsymbol{r}_2^{\downarrow}) & \cdots & \xi(\boldsymbol{r}_2^{\uparrow}, \boldsymbol{r}_P^{\downarrow}) & \varphi_1(\boldsymbol{r}_2^{\uparrow}) & \varphi_2(\boldsymbol{r}_2^{\uparrow}) & \cdots & \varphi_U(\boldsymbol{r}_2^{\uparrow}) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \xi(\boldsymbol{r}_P^{\uparrow}, \boldsymbol{r}_1^{\downarrow}) & \xi(\boldsymbol{r}_P^{\uparrow}, \boldsymbol{r}_2^{\downarrow}) & \cdots & \xi(\boldsymbol{r}_P^{\uparrow}, \boldsymbol{r}_P^{\downarrow}) & \varphi_1(\boldsymbol{r}_P^{\uparrow}) & \varphi_2(\boldsymbol{r}_P^{\uparrow}) & \cdots & \varphi_U(\boldsymbol{r}_P^{\uparrow}) \\ \varphi_1(\boldsymbol{r}_1^{\downarrow}) & \varphi_1(\boldsymbol{r}_2^{\downarrow}) & \cdots & \varphi_1(\boldsymbol{r}_P^{\downarrow}) & 0 & 0 & \cdots & 0 \\ \varphi_2(\boldsymbol{r}_1^{\downarrow}) & \varphi_2(\boldsymbol{r}_2^{\downarrow}) & \cdots & \varphi_2(\boldsymbol{r}_P^{\downarrow}) & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \varphi_U(\boldsymbol{r}_1^{\downarrow}) & \varphi_U(\boldsymbol{r}_2^{\downarrow}) & \cdots & \varphi_U(\boldsymbol{r}_P^{\downarrow}) & 0 & 0 & \cdots & 0 \end{bmatrix}. \quad (3.77)$$

The BCS wave function provides a more accurate description of the system's collective behavior in the presence of strong pairing correlations, such as in fermionic superfluids. However, it is limited in its ability to capture triplet correlations and is not applicable to Hamiltonians that involve spin exchange.

3.2.1.6 Pfaffian Wave Function

To overcome some the limitations of the BCS wave function, alternative wave functions based on the Pfaffian have been employed extensively in electronic structure as well as nuclear applications. Similar to the determinant, the Pfaffian of a matrix is a scalar value calculated by taking sums of products of permuted matrix elements. It is defined only for even-dimensional, skew-symmetric matrices with the explicit formula

$$\mathrm{pf}[A] = \frac{1}{2^n n!} \sum_{\hat{P} \in S_{2n}} \sigma(P) \prod_{i=1}^{n} a_{p(2i-1),p(2i)}, \tag{3.78}$$

where we have assumed A is a $2n \times 2n$ matrix. The Pfaffian allows us to write the most generic antisymmetric wave function constructed from pairing orbitals rather than single-particle ones. Assuming N is even, the Pfaffian wave function takes the form

$$\Phi_0(\boldsymbol{X}) = \mathrm{pf}\left[\phi(\boldsymbol{x}_i, \boldsymbol{x}_j)\right] = \mathrm{pf}\begin{bmatrix} 0 & \phi(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & \phi(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \phi(\boldsymbol{x}_2, \boldsymbol{x}_1) & 0 & \cdots & \phi(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\boldsymbol{x}_N, \boldsymbol{x}_1) & \phi(\boldsymbol{x}_N, \boldsymbol{x}_2) & \cdots & 0 \end{bmatrix}, \tag{3.79}$$

where $\phi(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is an *antisymmetric* pairing spin-orbital, $\phi(\boldsymbol{x}_i, \boldsymbol{x}_j) = -\phi(\boldsymbol{x}_j, \boldsymbol{x}_i)$. If we expand the matrix above to include unpaired single-particle spin-orbitals, mirroring the

approach we used for the BCS wave function, the wave function becomes the Pfaffian of an $(N+U)\times(N+U)$ matrix

$$\Phi_0(\boldsymbol{X}) = \mathrm{pf}\begin{bmatrix} \boldsymbol{\phi} & \boldsymbol{\varphi} \\ -\boldsymbol{\varphi}^T & \mathbf{0} \end{bmatrix}, \tag{3.80}$$

where $\boldsymbol{\phi}$ is the same $N\times N$ matrix displayed in Eq. (3.79), $\boldsymbol{\varphi}$ is an $N\times U$ matrix constructed from the occupied single-particle orbitals

$$\boldsymbol{\varphi} \equiv \begin{bmatrix} \varphi_1(\boldsymbol{x}_1) & \varphi_2(\boldsymbol{x}_1) & \cdots & \varphi_U(\boldsymbol{x}_1) \\ \varphi_1(\boldsymbol{x}_2) & \varphi_2(\boldsymbol{x}_2) & \cdots & \varphi_U(\boldsymbol{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_1(\boldsymbol{x}_N) & \varphi_2(\boldsymbol{x}_N) & \cdots & \varphi_U(\boldsymbol{x}_N) \end{bmatrix}, \tag{3.81}$$

and U is the number of unpaired particles. The pairing spin-orbital is commonly decomposed into explicit singlet and triplet contributions in order to write down the spatial dependence more easily

$$\begin{aligned} \phi(\boldsymbol{x}_i,\boldsymbol{x}_j) = {} & \xi(\boldsymbol{r}_i,\boldsymbol{r}_j)\langle s_i,s_j|\left(|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle\right)/\sqrt{2} \\ & + \chi^{\uparrow\uparrow}(\boldsymbol{r}_i,\boldsymbol{r}_j)\langle s_i,s_j|\uparrow\uparrow\rangle \\ & + \chi^{\uparrow\downarrow}(\boldsymbol{r}_i,\boldsymbol{r}_j)\langle s_i,s_j|\left(|\uparrow\downarrow\rangle + |\downarrow\uparrow\rangle\right)/\sqrt{2} \\ & + \chi^{\downarrow\downarrow}(\boldsymbol{r}_i,\boldsymbol{r}_j)\langle s_i,s_j|\downarrow\downarrow\rangle. \end{aligned} \tag{3.82}$$

The singlet pairing orbital $\xi(\boldsymbol{r}_i,\boldsymbol{r}_j)$ is the same as $\xi(\boldsymbol{r}_i^{\uparrow},\boldsymbol{r}_j^{\downarrow})$ in the BCS wave function, and must be even with respect to the spatial coordinates. The newly introduced triplet pairing orbitals $\chi^{\uparrow\uparrow}(\boldsymbol{r}_i,\boldsymbol{r}_j)$, $\chi^{\uparrow\downarrow}(\boldsymbol{r}_i,\boldsymbol{r}_j)$, $\chi^{\downarrow\downarrow}(\boldsymbol{r}_i,\boldsymbol{r}_j)$ must all be odd.

When the Hamiltonian does not exchange spin, in condensed matter problems for

instance, we can further decompose the $\boldsymbol{\phi}$ and $\boldsymbol{\varphi}$ matrices into separate blocks corresponding to the singlet and triplet contributions, and the spin-up and spin-down unpaired contributions, respectively. This singlet-triplet-unpaired (STU) Pfaffian ansatz, developed in Ref. [6], is written as

$$\Phi_0(\boldsymbol{X}) = \mathrm{pf} \begin{bmatrix} \boldsymbol{\chi}^{\uparrow\uparrow} & \boldsymbol{\xi}^{\uparrow\downarrow} & \boldsymbol{\varphi}^{\uparrow} \\ -\boldsymbol{\xi}^{\uparrow\downarrow T} & \boldsymbol{\chi}^{\downarrow\downarrow} & \boldsymbol{\varphi}^{\downarrow} \\ -\boldsymbol{\varphi}^{\uparrow T} & -\boldsymbol{\varphi}^{\downarrow T} & \mathbf{0} \end{bmatrix}, \tag{3.83}$$

where we have omitted the $\chi^{\uparrow\downarrow}(\boldsymbol{r}_i, \boldsymbol{r}_j)$ term in Eq. (3.82). The singlet block $\boldsymbol{\xi}^{\uparrow\downarrow}$ is an $N_\uparrow \times N_\downarrow$ matrix constructed from applying the singlet pairing orbital $\xi(\boldsymbol{r}_i^\uparrow, \boldsymbol{r}_j^\downarrow)$ to all combinations of the spin-up and spin-down particles. The triplet blocks $\boldsymbol{\chi}^{\uparrow\uparrow}$ and $\boldsymbol{\chi}^{\downarrow\downarrow}$ have sizes $N_\uparrow \times N_\uparrow$ and $N_\downarrow \times N_\downarrow$, respectively. They are both skew-symmetric and constructed by applying the triplet pairing orbitals $\chi^{\uparrow\uparrow}(\boldsymbol{r}_i^\uparrow, \boldsymbol{r}_j^\uparrow)$ and $\chi^{\downarrow\downarrow}(\boldsymbol{r}_i^\downarrow, \boldsymbol{r}_j^\downarrow)$ to the appropriate same-spin pairs. Finally, the unpaired blocks $\boldsymbol{\varphi}^\uparrow$ and $\boldsymbol{\varphi}^\downarrow$ can be viewed as a reorganization of the rows in $\boldsymbol{\varphi}$ such that the spin-up particles and spin-down particles are placed into separate blocks.

The STU wave function is a convenient approach if the spins of the particles can be fixed during the Monte Carlo simulation. Otherwise, the more general Pfaffian wave function can be used to handle the exchange of spins. Either way, the pairing spin-orbital $\phi(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is decomposed according to Eq. (3.82) so that we can leverage our intuition in its design. In our investigation of neural-network quantum states, we will find that such a decomposition is not necessary. Instead, we can construct the most general antisymmetric pairing orbital that allows the spatial and spin degrees of freedom to influence one another arbitrarily.

3.2.2 Optimization

After selecting a specific parameterization of the trial wave function, we require methods by which the variational parameters can be changed in order to minimize the energy. These optimization methods operate iteratively, gradually refining the solution over time. The first-order optimization methods we will discuss, known as gradient descent methods, coincide with the very same techniques commonly employed in numerous machine learning problems. However, we will also introduce one additional second-order optimization method, known as Stochastic Reconfiguration, which is specifically tailored for the variational Monte Carlo method. It is akin to the Natural Gradient method in the context of machine learning, but as our application of neural networks will always involve minimizing the energy, as opposed to a general objective function, we opt to explore all optimization methods within the context of variational Monte Carlo rather than machine learning.

3.2.2.1 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is a simple method that utilizes a single estimate of the gradient $\boldsymbol{\nabla}_{\boldsymbol{\theta}}\langle E_{\boldsymbol{\theta}}(\boldsymbol{X})\rangle$ to update the parameters $\boldsymbol{\theta}$. At any given iteration of the optimization procedure, the parameters are transformed following the opposite direction of the gradient

$$\boldsymbol{\theta} \leftarrow\!\shortmid \boldsymbol{\theta} - \eta \boldsymbol{\nabla}_{\boldsymbol{\theta}}\langle E_{\boldsymbol{\theta}}(\boldsymbol{X})\rangle, \tag{3.84}$$

where η is a small learning rate and the gradient is defined in Eq. (3.51). The learning rate can be scheduled to decrease with the number of optimization steps, with typical values between 10^{-5} and 10^{-2}.

Since our measures of the gradient are approximate, our trajectory through the energy

landscape can be noisy. On one hand, this noise can sometimes prevent us from getting trapped in a local minimum, as there is always a chance of a perturbation that propels us out of it. On the other hand, the presence of this noise can make the overall training procedure slow and inaccurate, as we may find ourselves frequently oscillating back and forth in a particularly narrow or shallow valley.

3.2.2.2 Momentum

The simple SGD algorithm can be improved by including momentum, a moving average of the gradients. Momentum is implemented by first defining a vector $\boldsymbol{m}$ that stores the moving average of gradients, and initializing it to zero. Then parameter updates are performed as

$$\boldsymbol{m} \leftarrow\!\shortmid \beta \boldsymbol{m} + (1-\beta)\boldsymbol{\nabla}_{\boldsymbol{\theta}}\langle E_{\boldsymbol{\theta}}(\boldsymbol{X})\rangle, \tag{3.85}$$

$$\boldsymbol{\theta} \leftarrow\!\shortmid \boldsymbol{\theta} - \eta \boldsymbol{m}, \tag{3.86}$$

where η is the learning rate and β is a constant hyperparameter that controls the influence of the previous gradients on the current update, usually set around $\beta = 0.9$. By averaging over a history of gradients, the optimization trajectory becomes smoother as oscillations in opposite directions tend to cancel out. This smoothing effect helps to stabilize the parameter updates and prevent the algorithm from getting trapped in erratic behavior. Moreover, in situations where the gradients become small, such as in plateaus or saddle points, momentum helps the training process to persist in the "right" direction.

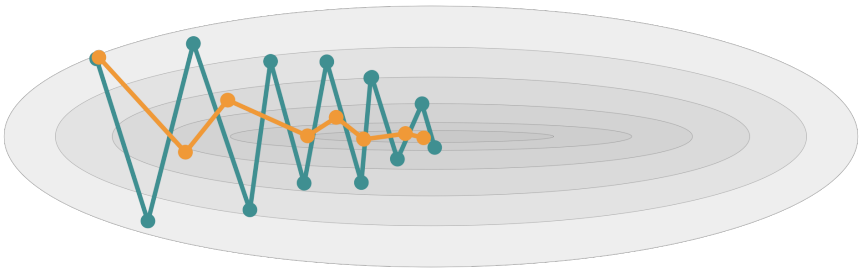

Figure 3.4: This cartoon depicts trajectories of stochastic gradient descent, with momentum (orange) and without (blue), near a narrow parameter space minimum. Momentum smooths the path by counteracting noise in opposing directions.

3.2.2.3 Root Mean Squared Propagation

Root Mean Squared Propagation (RMSprop) adaptively adjusts the learning rate for each parameter based on the magnitudes of the recent gradients. We begin by initializing a new vector $\boldsymbol{v} = \mathbf{0}$, and updating the parameters as

$$\boldsymbol{v} \leftarrow \beta \boldsymbol{v} + (1-\beta)\big(\boldsymbol{\nabla}_{\boldsymbol{\theta}} \langle E_{\boldsymbol{\theta}}(\boldsymbol{X}) \rangle\big)^2, \tag{3.87}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \frac{\eta}{\sqrt{\boldsymbol{v}} + \epsilon} \boldsymbol{\nabla}_{\boldsymbol{\theta}} \langle E_{\boldsymbol{\theta}}(\boldsymbol{X}) \rangle, \tag{3.88}$$

where the square in Eq. (3.87) and the square root in Eq. (3.88) are both element-wise operations. The hyperparameter β is typically set to 0.9. The hyperparameter ϵ provides numerical stability and is set around 10^{-8}. Unlike the previous methods, RMSprop is less sensitive to choices of the learning rate η. By normalizing the gradients in this way, RMSprop mitigates the problem of exploding gradients, as extreme fluctuations are alleviated, making the optimization process more stable. At the opposite extreme, RMSprop also prevents gradients from becoming too small, allowing training to continue effectively.

3.2.2.4 Adaptive Moment Estimation

Adaptive Moment Estimation (ADAM) combines the concepts of RMSprop and momentum, and it remains one of the most popular first-order optimization methods since its introduction in 2015[8]. In addition to computing the moving averages of gradients and the squares of gradients, the ADAM algorithm incorporates a learning rate correction to compensate for the bias caused by initializing these averages to zero. This correction is dependent on the optimization iteration t, so we begin by setting $t = 0$, $\boldsymbol{m} = \mathbf{0}$, and $\boldsymbol{v} = \mathbf{0}$. Then the parameters are updated as

$$\boldsymbol{m} \leftarrow \alpha \boldsymbol{m} + (1-\alpha)\boldsymbol{\nabla}_{\boldsymbol{\theta}} \langle E_{\boldsymbol{\theta}}(\boldsymbol{X}) \rangle, \tag{3.89}$$

$$\boldsymbol{v} \leftarrow \beta \boldsymbol{v} + (1-\beta)\big(\boldsymbol{\nabla}_{\boldsymbol{\theta}} \langle E_{\boldsymbol{\theta}}(\boldsymbol{X}) \rangle\big)^2, \tag{3.90}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \left(\frac{1-\beta^t}{1-\alpha^t} \right) \frac{\boldsymbol{m}}{\sqrt{\boldsymbol{v}} + \epsilon}, \tag{3.91}$$

$$t \leftarrow t + 1. \tag{3.92}$$

The hyperparameters are usually set to $\alpha = 0.9$, $\beta = 0.999$, and $\epsilon = 10^{-8}$. ADAM has gained popularity across diverse optimization problems, thanks to its efficient and robust performance.

3.2.2.5 Stochastic Reconfiguration

Stochastic Reconfiguration [9] (SR) is a second-order optimization method specific to the variational Monte Carlo (VMC) method. Instead of manipulating the gradients according to their history, the SR algorithm manipulates the gradients according to the *curvature* of the energy landscape. It can alternatively be viewed as stretching and squeezing the

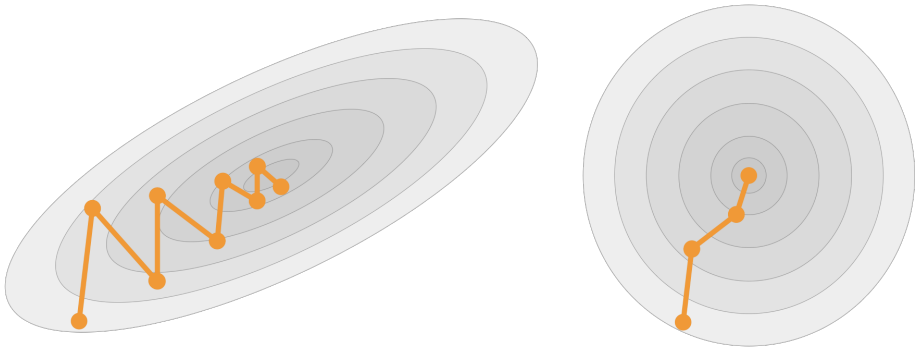

Figure 3.5: A visualization of the Stochastic Reconfiguration algorithm, a second-order optimization method that stretches and squeezes the energy landscape such that difficult minima (left) are more isotropic (right). This method typically reduces the required number of optimization steps by an order of magnitude compared to the simple stochastic gradient descent method.

landscape itself, making it smoother or more isotropic in certain areas. Stochastic Reconfiguration provides a more favorable terrain for finding the global minimum and improves the exploration of the parameter space.

The SR update is derived by determining the parameter change, denoted as $\delta\boldsymbol{\theta}$, that best reproduces a small step $\delta\tau$ in an imaginary-time evolution. The concept of imaginary-time propagation, which will be further elaborated on in Sec. 9.5, involves applying the operator $e^{-\delta\tau\hat{H}}$ to states non-orthogonal to the ground state, thereby driving them closer to the ground state. We can approximate the imaginary-time propagator to first order in $\delta\tau$ and apply it to our original state $|\Psi_{\boldsymbol{\theta}}\rangle$, resulting in a new state

$$|\Phi\rangle = e^{-\delta\tau\hat{H}}|\Psi_{\boldsymbol{\theta}}\rangle \approx (1-\delta\tau\hat{H})|\Psi_{\boldsymbol{\theta}}\rangle = |\Psi_{\boldsymbol{\theta}}\rangle - \delta\tau\hat{H}|\Psi_{\boldsymbol{\theta}}\rangle. \tag{3.93}$$

Our goal is to match this state to one obtained by changing the parameters $\boldsymbol{\theta}$. To do this, we expand around our original state to first order in $\delta\boldsymbol{\theta}$,

$$|\Psi_{\boldsymbol{\theta}+\delta\boldsymbol{\theta}}\rangle \approx |\Psi_{\boldsymbol{\theta}}\rangle + (\delta\boldsymbol{\theta})\cdot\boldsymbol{\nabla}_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle. \tag{3.94}$$

For $|\Phi\rangle$ and $|\Psi_{\boldsymbol{\theta}+\delta\boldsymbol{\theta}}\rangle$ to coincide, we need their projections onto the original state be the same, keeping in mind it may not be normalized,

$$\frac{\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}+\delta\boldsymbol{\theta}}\rangle}{\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle} = \frac{\langle\Psi_{\boldsymbol{\theta}}|\Phi\rangle}{\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle}, \tag{3.95}$$

$$\frac{\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle}{\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle} + (\delta\boldsymbol{\theta})\cdot\frac{\langle\Psi_{\boldsymbol{\theta}}|\boldsymbol{\nabla}_{\boldsymbol{\theta}}\Psi_{\boldsymbol{\theta}}\rangle}{\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle} = \frac{\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle}{\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle} - \delta\tau\frac{\langle\Psi_{\boldsymbol{\theta}}|\hat{H}|\Psi_{\boldsymbol{\theta}}\rangle}{\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle}, \tag{3.96}$$

Similarly, the gradient with respect to the parameters must also coincide,

$$\frac{\langle\Psi_{\boldsymbol{\theta}}|\boldsymbol{\nabla}_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}+\delta\boldsymbol{\theta}}\rangle}{\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle} = \frac{\langle\Psi_{\boldsymbol{\theta}}|\boldsymbol{\nabla}_{\boldsymbol{\theta}}|\Phi\rangle}{\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle}, \tag{3.97}$$

$$\frac{\langle\Psi_{\boldsymbol{\theta}}|\boldsymbol{\nabla}_{\boldsymbol{\theta}}\Psi_{\boldsymbol{\theta}}\rangle}{\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle} + (\delta\boldsymbol{\theta})\cdot\frac{\langle\boldsymbol{\nabla}_{\boldsymbol{\theta}}\Psi_{\boldsymbol{\theta}}|\boldsymbol{\nabla}_{\boldsymbol{\theta}}\Psi_{\boldsymbol{\theta}}\rangle}{\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle} = \frac{\langle\Psi_{\boldsymbol{\theta}}|\boldsymbol{\nabla}_{\boldsymbol{\theta}}\Psi_{\boldsymbol{\theta}}\rangle}{\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle} - \delta\tau\frac{\langle\boldsymbol{\nabla}_{\boldsymbol{\theta}}\Psi_{\boldsymbol{\theta}}|\hat{H}|\Psi_{\boldsymbol{\theta}}\rangle}{\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle}, \tag{3.98}$$

$$\tag{3.99}$$

Combining Eqs. (3.96) and (3.99) reveals the following update rule for the parameters:

$$\boldsymbol{\theta} \leftarrow\!\shortmid\ \eta S^{-1}\boldsymbol{\nabla}_{\boldsymbol{\theta}}\langle E_{\boldsymbol{\theta}}(\boldsymbol{X})\rangle, \tag{3.100}$$

where S is the quantum geometric tensor [10] with the following matrix elements

$$S_{ij} = \frac{\langle\partial_i\Psi_{\boldsymbol{\theta}}|\partial_j\Psi_{\boldsymbol{\theta}}\rangle}{\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle} - \frac{\langle\partial_i\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle\langle\Psi_{\boldsymbol{\theta}}|\partial_j\Psi_{\boldsymbol{\theta}}\rangle}{\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle} \tag{3.101}$$

and ∂_i denotes a derivative with respect to the ith variational parameter θ_i.

3.3 Diffusion Monte Carlo

Diffusion Monte Carlo (DMC) is a virtually exact stochastic projector method based on solving the imaginary-time many-body Schrödinger equation. By transforming $t \mapsto -i\tau\hbar$, the Schrödinger equation becomes

$$-\frac{\partial}{\partial\tau}\Psi(\boldsymbol{X},\tau) = \hat{H}\Psi(\boldsymbol{X},\tau), \tag{3.102}$$

and the originally oscillating stationary-state solutions are now convergent in the $\tau \to \infty$ limit

$$\Psi_\alpha(\boldsymbol{X},\tau) = \psi_\alpha(\boldsymbol{X})e^{-E_\alpha\tau}, \tag{3.103}$$

where

$$\hat{H}\psi_\alpha(\boldsymbol{X}) = E_\alpha\psi_\alpha(\boldsymbol{X}). \tag{3.104}$$

Let us assume these energy eigenstates are ordered such that $E_0 \leq E_1 \leq E_2 \leq \cdots$. By expanding the general wave function in terms of the transformed stationary states,

$$\Psi(\boldsymbol{X},\tau) = \sum_{\alpha=0}^{\infty} c_\alpha\Psi_\alpha(\boldsymbol{X},\tau) = \sum_{\alpha=0}^{\infty} c_\alpha\psi_\alpha(\boldsymbol{X})e^{-E_\alpha\tau}, \tag{3.105}$$

we find that any state that is not orthogonal to the ground state, $c_0 \neq 0$, will evolve to the ground state

$$\lim_{\tau\to\infty}\Psi(\boldsymbol{X},\tau) = c_0\psi_0(\boldsymbol{X})e^{-E_0\tau}, \tag{3.106}$$

in the large τ limit. By introducing a constant offset energy E_T to Eq. (3.102), where E_T is as close to the ground state energy E_0 as possible, the large τ limit can be kept finite, while

leaving the eigenstates themselves unchanged. Then by ignoring spin momentarily, $\boldsymbol{X} \to \boldsymbol{R}$, the shifted imaginary-time Schrödinger equation then reads

$$\frac{\partial}{\partial \tau}\Psi(\boldsymbol{R},\tau) = \left(D\boldsymbol{\nabla}\cdot\boldsymbol{\nabla} - (V(\boldsymbol{R}) - E_T)\right)\Psi(\boldsymbol{R},\tau), \tag{3.107}$$

where we have again used the notation $D = \frac{\hbar^2}{2m}$ and $\boldsymbol{\nabla} \equiv (\boldsymbol{\nabla}_1, \boldsymbol{\nabla}2, ..., \boldsymbol{\nabla}_N)$. At this point, the above equation should look familiar, as it has a similar form to the Fokker-Planck equation we introduced in Eq. (3.25). In fact, if we make the substitution $\Psi(\boldsymbol{R},\tau) = f(\boldsymbol{R},\tau)/\Psi_T(\boldsymbol{R})$, take all the necessary derivatives, and multiply by $\Psi_T(\boldsymbol{R})$, the equation we obtain for $f(\boldsymbol{R},\tau)$, has nearly identical form to the Fokker-Planck equation

$$\frac{\partial}{\partial \tau}f(\boldsymbol{R},\tau) = D\boldsymbol{\nabla}\cdot\left(\boldsymbol{\nabla} - \boldsymbol{F}(\boldsymbol{R})\right)f(\boldsymbol{R},\tau) - (E(\boldsymbol{R}) - E_T)f(\boldsymbol{R},\tau), \tag{3.108}$$

except for the last term, where

$$E(\boldsymbol{R}) = \frac{1}{\Psi_T(\boldsymbol{R})}\hat{H}\Psi_T(\boldsymbol{R}). \tag{3.109}$$

This modified form of the Schrödinger equation can be solved for $f(\boldsymbol{R},\tau)$ through standard Green's function methods, but as we already mentioned before at the end of Sec. 3.1.1, the Green's function solution to the Fokker-Planck equation coincides with the proposal density $Q(\boldsymbol{R}'|\boldsymbol{R},\Delta t)$ we inferred from the corresponding Langevin equation. Without the extra term in Eq. (3.108), the Green's function solution would be exactly the same as before, as introduced in Eq. (3.34),

$$G_D(\boldsymbol{R}'|\boldsymbol{R},\Delta\tau) = \frac{1}{(4\pi D\Delta\tau)^{Nd/2}}e^{-(\boldsymbol{R}'-\boldsymbol{R}-D\Delta\tau\boldsymbol{F}(\boldsymbol{R}))^2/4D\Delta\tau}, \tag{3.110}$$

for small $\Delta\tau$. The subscript D stands for drift. However, the inclusion of the last term modifies the Green's function, which can be approximated by

$$G_B(\boldsymbol{R}'|\boldsymbol{R}, \Delta\tau) = e^{\Delta\tau(E(\boldsymbol{X}')+E(\boldsymbol{X})-2E_T)/2}, \tag{3.111}$$

where B standard for branching. Thus the total Green's function is approximated as

$$G(\boldsymbol{R}'|\boldsymbol{R}, \Delta\tau) \approx G_D(\boldsymbol{R}'|\boldsymbol{R}, \Delta\tau)G_B(\boldsymbol{R}'|\boldsymbol{R}, \Delta\tau). \tag{3.112}$$

The factorization of the Green's function suggests a way we can evolve to the ground state in practice. First, we initialize an ensemble of MCMC walkers according to the probability distribution of a good estimation of the wave function $|\Psi_T(\boldsymbol{R})|^2$. Next, by using the importance sampling method we covered in Sec. 3.1.1, we move the walkers as usual. Then, to account for the extra branching term in the Green's function, we make copies of the walkers with a probability given by Eq. (3.111) . Proceeding in this way is the spatial realization of the imaginary-time propagation operator $e^{-\tau(\hat{H}-E_T)}$.

Unlike the variational Monte Carlo method, where walkers are sampled from a positive semi-definite distribution given by $|\Psi_{\boldsymbol{\theta}}(\boldsymbol{X})|^2$, a diffusion Monte Carlo calculation employs the walkers' distribution itself to represent the wave function. For systems of bosons, this poses no issue, but for fermions, it is difficult to maintain the antisymmetry of the wave function. Therefore, when we reintroduce spin into our formulation, we must add the extra step of killing the walkers if they attempt to cross a node. This solution is known as the fixed-node approximation. The nodes themselves are determined by a previous VMC calculation, and consequently, DMC calculations carry a residual dependence on the VMC calculation.

4 Machine Learning

Machine learning encompasses a wide range of computational models and algorithms designed to learn from data rather than rely on explicit programming. It is a subset of artificial intelligence, a rapidly evolving field that aims to endow computers with human-like cognition, such as the ability to interpret complicated data, adapt to new situations, and make decisions autonomously. These ambitious goals have driven the development of machine learning models and have transformed them into the highly versatile tools we witness today.

In this chapter, we introduce fundamental concepts in machine learning, in the hopes of building a broad perspective that extends beyond specific physics applications. We will explore different approaches to learning, providing context for the artificial neural networks we will cover afterwards. In particular, we place emphasis on the mathematical aspects of artificial neural networks, aiming to clarify their relevance and utility as neural-network quantum states.

4.1 The Curse of Dimensionality

When dealing with high-dimensional data, be it in machine learning or any data other analysis endeavor, a recurring challenge arises. It is known as the "curse of dimensionality," wherein the performance of algorithms decreases as the number of dimensions increases. The basis of the curse of dimensionality is geometric: given a fixed number of data points, the density of those points in space decreases exponentially with the number of dimensions. To counteract the sparsity of the data, is it common to focus considerable effort and resources on

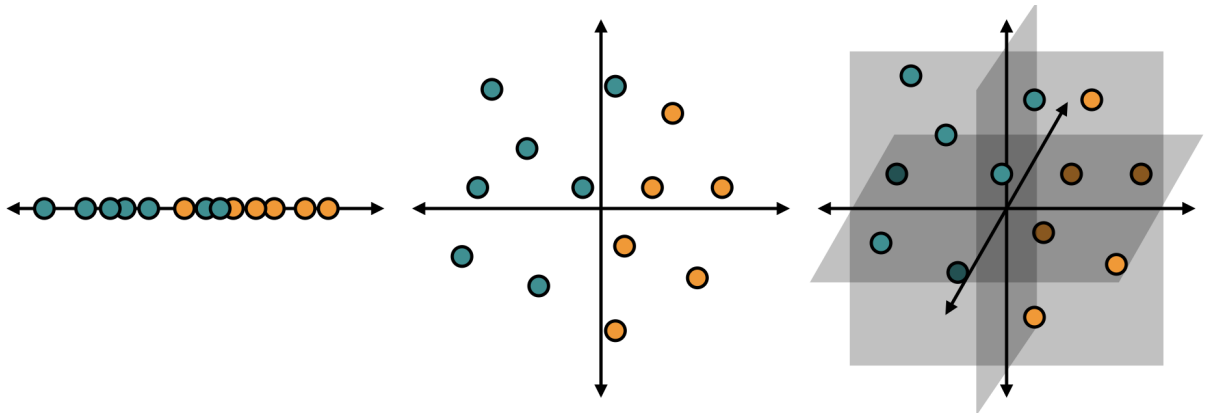

Figure 4.1: Cartoon of the curse of dimensionality; as the dimension of the data increases (left to right), the density of a fixed number of data points decreases exponentially.

finding a lower-dimensional representation of the data before applying any given algorithm.

Is it reasonable to assume such a lower-dimensional representation exists? The short answer is yes—pure geometric scaling only truly applies if our data set contains no information at all, i.e. it is completely random. Real-life data sets, ones that actually contain *information*, have structure. This is reminiscent of the exponential scaling of the Hilbert space dimension with respect to the size of the quantum system. As we have discussed in Sec. 2.2, the effective Hilbert space dimension decreases if we restrict ourselves only to states that accurately represent real, physical systems.

Despite the challenges posed by the curse of dimensionality, machine learning algorithms have proven exceptional in discovering lower-dimensional representations of data. Among them, artificial neural networks have emerged as unparalleled tools for handling large and intricate datasets, where the underlying structure is difficult to predict a priori. Consequently, neural networks, particularly deep ones, have established themselves as the dominant approach for tackling some of the most formidable problems across various domains. Notably, they have been instrumental in solving complex tasks such as protein folding, game strategizing, natural language processing, image and voice

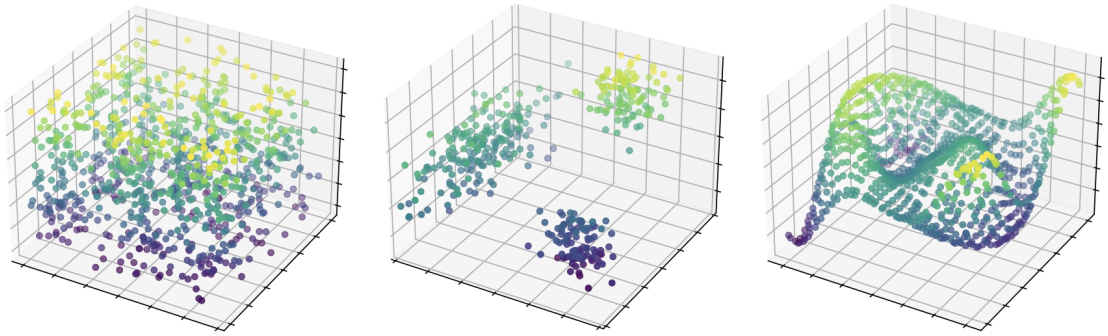

Figure 4.2: Comparison of a random data set (left) with no information, a data set with obvious clusters (middle), and a data set with an obvious lower-dimensional manifold (right). The latter two are examples of real-life data sets that contain information, i.e. they have structure.

recognition, and autonomous robotics.

4.2 Cost Functions

The goals of a machine learning problem are often narrow in scope, usually involving one or two tasks that can be expressed mathematically through a so-called "cost function". The terms "cost function" and "loss function" are often used interchangeably, but the cost function typically refers to the overall measure of error across the entire dataset, while the loss function calculates the error for individual data points. If the objective function needs to be maximized rather than minimized, it is commonly called a "reward function" in the context of machine learning.

While not all machine learning problems are explicitly framed as optimization problems, many involve minimizing or maximizing certain quantities. In some cases, this optimization can be achieved in a single step, leading to simple and highly efficient models, such as Kernel Ridge Regression and Gaussian Processes. However, more generally, optimization problems require iterative methods, as discussed in the Sec. 3.2.2. For the purposes of our discussion,

we will assume that all machine learning problems have an underlying cost function, even if it may not be obvious at first glance.

Determining the structure of a particular problem ultimately depends on the training data that is, or can be made, available. In the following sections, we will aim to give an intuitive understanding of machine learning and why exactly it has been proven to be so powerful across a wide variety of applications. When faced with a brand new problem, the reader should be able to zoom out, and quickly identify its core elements and interdependencies. Most importantly, the reader should be able to break a large, complicated problem into small, digestible components that are each suitable for a simple machine learning problem.

4.3 Supervised Learning

The goal of a supervised learning problem is to identify a mapping between two spaces, where only examples of the endpoints in each space are known. If we denote $\boldsymbol{x}$ as the data points in our input space, $\boldsymbol{y}$ as the data points in our output space, and $\hat{\boldsymbol{y}}$ as the predictions of our model in the output space, then the cost function of any supervised learning problem can be written as $C(\boldsymbol{y}(\boldsymbol{x}), \hat{\boldsymbol{y}}(\boldsymbol{x}))$, where C measures the discrepancy between $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$. The elements within the output space are often referred to as "labels." However, we will refrain from using this terminology to prevent any potential misconception that the output space must necessarily be discrete in nature.

Supervised learning problems can be separated into two major categories: regression and classification. The former involves predicting a continuous numerical value, while the latter involves predicting discrete values corresponding to distinct categories. Python packages like `scikit-learn` and `Keras` provide a wide range of pre-implemented supervised learning algorithms that are readily available for use "out-of-the-box," including

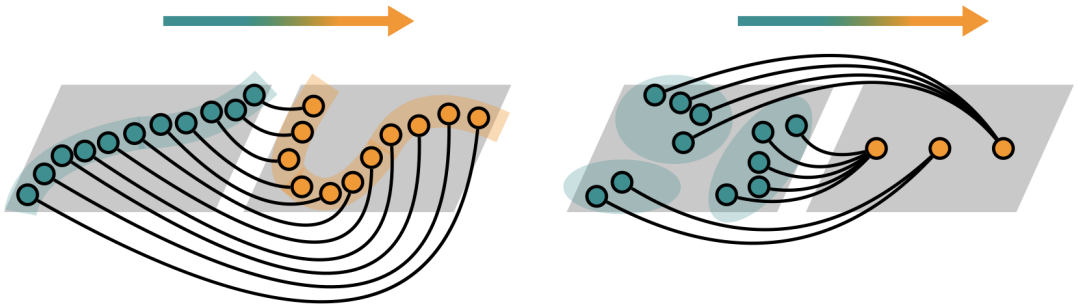

Figure 4.3: A visualization of the two main types of supervised learning: regression (left) and classification (right). The blue data points are the inputs, and the orange data points are the target outputs. The goal of a supervised machine learning problem is to learn and generalize the mapping (solid black lines).

decision trees, support vector machines, linear and logistic regression, and neural networks, to name a few.

Choosing an appropriate model depends strongly on the desired task, the complexity of the data, and the number of data points. If the number of data points is limited compared to the flexibility of a certain model, it is likely to overfit the data, leading to poor predictive power when new data is introduced. On the other hand, models that are too simple can fail at the task all together. Balancing the risk of overfitting and underfitting is a reflection of the bias-variance trade-off, the clash between a model's sensitivity to fluctuations in the input data and the model's underlying assumptions about the mapping.

To manage these challenges, supervised learning techniques commonly involve dividing a fixed dataset into three distinct parts: a training set, a validation set, and a test set. The training set is utilized to optimize the model, as its name implies. The validation set is used to estimate the bias and variance of the model throughout the training process, helping to determine the ideal hyperparameters. Finally, the model's performance is evaluated on the test set once training is completed.

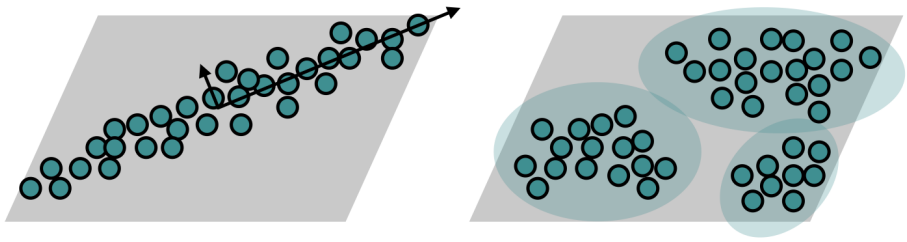

Figure 4.4: A depiction of principal component analysis (left) and cluster identification (right), two common types of unsupervised learning.

4.4 Unsupervised Learning

Unlike supervised learning, where the model learns a mapping from an input space to an output space, unsupervised learning models aim to learn the underlying structure of the input data set itself. Therefore, the associated cost functions solely depend on the inputs, $C(\boldsymbol{x})$. Unsupervised learning problems can be divided into three main categories: clustering, dimensionality reduction, and anomaly detection. Each of these categories can be used as a preprocessing step in conjunction with supervised or reinforcement learning problems to improve their overall performance.

In unsupervised clustering tasks, the model assumes the input data can be mapped to some latent space in which similar data points are physically closer together. Specifically, the assumed structure of the points in the n-dimensional latent space are (roughly) n-spheres, where n is at most the number of dimensions in the original input data. The key ingredient of such models is the quantification of "distance" between points. In simple problems, the input space itself can act as the latent space, as the data points may already exist in localized groups.

Clustering and dimensionality reduction are not necessarily distinct tasks, as both aim

to find some lower-dimensional representation of the data. However, the latter typically refers to problems in which the underlying structure of the dataset consists of hyperplanes rather than hyperspheres. Commonly used algorithms include Principal Component Analysis and Singular Value Decomposition, both of which seek to find the most relevant degrees of freedom in the data.

Dimensionality reduction problems may also include finding more abstract representations of the data. For instance, autoencoders are neural networks with an architecture that maps to a very small latent space before mapping back to the original input space. After training the autoencoder to reproduce each element of the data set, the latent space representation contains only the most essential and informative features of the data. This style of dimensionality reduction is particularly useful for file or image compression, as the relationships between data points can be highly non-intuitive.

Both clustering and dimensionality reduction go hand in hand with anomaly detection, because it focuses on identifying data points that deviate significantly from the expected behavior. Anomaly detection plays an important role in enhancing security in the digital age by identifying fraudulent behavior and detecting malware at a pace that surpasses human capabilities. Furthermore, anomaly detection finds practical utility in experimental sciences, where it aids in discerning genuine events from background noise. This capability enables efficient allocation of memory resources to statistically significant events, ensuring optimal utilization of computational resources.

4.5 Reinforcement Learning

Reinforcement learning is a specific branch of machine learning in which datasets rely on the model itself. If we let $\hat{\boldsymbol{y}}_t$ denote the state of the model at iteration t, and $\boldsymbol{x}_t(\hat{\boldsymbol{y}}_t)$ denote

the data collected based on that state, then the cost function of a reinforcement learning problem may be written schematically as

$$C(\boldsymbol{x}_t(\hat{\boldsymbol{y}}_t(\boldsymbol{x}_{t-1}(\hat{\boldsymbol{y}}_{t-1}(\cdots(\boldsymbol{x}_0(\hat{\boldsymbol{y}}_0))\cdots))))). \tag{4.1}$$

In other words, starting from some initial model $\hat{\boldsymbol{y}}_0$, the data $\boldsymbol{x}_0(\hat{\boldsymbol{y}}_0)$ are generated by letting the model attempt the task at hand. Its performance is evaluated and used to updated the model for the next iteration, continuing to collect new data based on its previous experiences. This notation for the cost function is simply meant to reflect the inherent cyclic nature of the problem; it is never explicitly written in this form in practice.

Reinforcement learning is commonly framed in terms of agent-based modeling, because the model learns by interacting with its environment and receiving positive or negative feedback, mirroring the way humans learn through trial and error in their everyday experiences (outside of the classroom). This learning approach is widely employed to train robots in game-playing and navigation tasks, which is impossible to accomplish through supervised learning methods as it would require an immense amount of example data.

Let us now rephrase variational Monte Carlo explicitly in the agent-based modeling framework. We begin by initializing the state of our agent, i.e. setting random parameters in our wave function. Then we allow our agent to interact with the environment, collecting data along the way. In VMC, the environment is the many-body Hilbert space and the data are the sampled configurations $\boldsymbol{X}$. Then we evaluate the performance of our agent, by calculating the average local energy based on the samples. This process continues until convergence is reached.

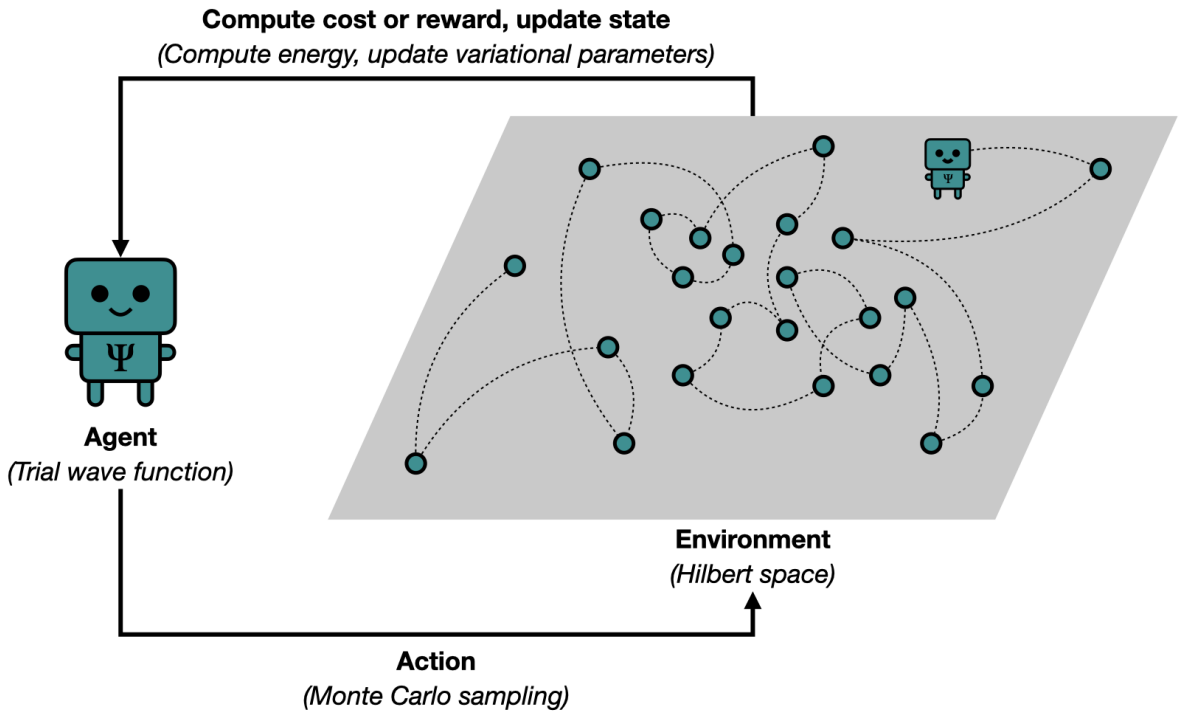

Figure 4.5: A depiction of the variational Monte Carlo algorithm in the context of agent-based modeling, a commonly employed reinforcement learning framework.

4.6 Transfer Learning

Transfer learning is a powerful technique that applies the knowledge gained from solving one problem to a more challenging and sometimes even unrelated problem. For example, researchers [11] have found that pre-training a convolutional neural network on ImageNet data provides immediate benefits for training the network on particle detection data. Interestingly, a fixed, pretrained network can be appended with additional trainable layers, while still achieving accelerated and stabilized training. This approach effectively reduces the number of trainable parameters while preserving the entire network's representation power. Consequently, this finding suggests that all image recognition models rely on a shared set of core capabilities.

When using neural-network quantum states for variational Monte Carlo, we will use transfer learning to handle particularly challenging interaction potentials. Pre-training the neural-network quantum states on softer potentials allows the training on harder potentials to proceed in a smooth, controlled manner. These potentials may contain singularities, which we renormalize according to some newly defined hyperparameter, or they may simply be hard by nature. We will discuss transfer learning in these applications on a case-by-case basis.

4.7 Artificial Neural Networks

We are now in a suitable position to discuss our primary focus: the mathematics of artificial neural networks (ANNs). This class of machine learning models is inspired by the structure of the human brain, consisting of interconnected nodes and nonlinear connections between them. While the term is often used interchangeably with feedforward neural networks, it is

important to note that the latter represents just one specific type within the broader ANN framework. We will exclusively refer to the broader class as artificial neural networks to ensure clarity.

In the following sections, we will formulate our artificial neural networks with the eventual goal of applying them as trial wave functions for variational Monte Carlo. The networks will be trained using the same optimization algorithms we use for traditional variational Monte Carlo calculations, discussed in Sec. 3.2.2. Consequently, our discussion may prioritize certain aspects that differ from those typically encountered in standard machine learning applications. Nonetheless, we will strive to maintain a level of generality throughout our discourse, ensuring that the concepts and techniques discussed herein can be applied to a wide range of problems.

4.7.1 Boltzmann Machines

Boltzmann machines are generative stochastic artificial neural networks that are primarily used for unsupervised learning tasks. They have two types of nodes: visible nodes, which represent the inputs to the network, and hidden nodes, which serve as latent variables capturing higher-level features or patterns in the data. In a general Boltzmann machine, all the nodes are fully connected, regardless of type. Additionally, the connections are undirected, unlike other well-known models such as feedforward neural networks. Unconstrained Boltzmann machines have limited practical utility for machine learning and inference. However, their optimization efficiency can be greatly enhanced by appropriately restraining the connectivity. This improvement allows for better applicability in solving real-world problems.

Restricted Boltzmann machines (RBMs) are a variant of Boltzmann machines that

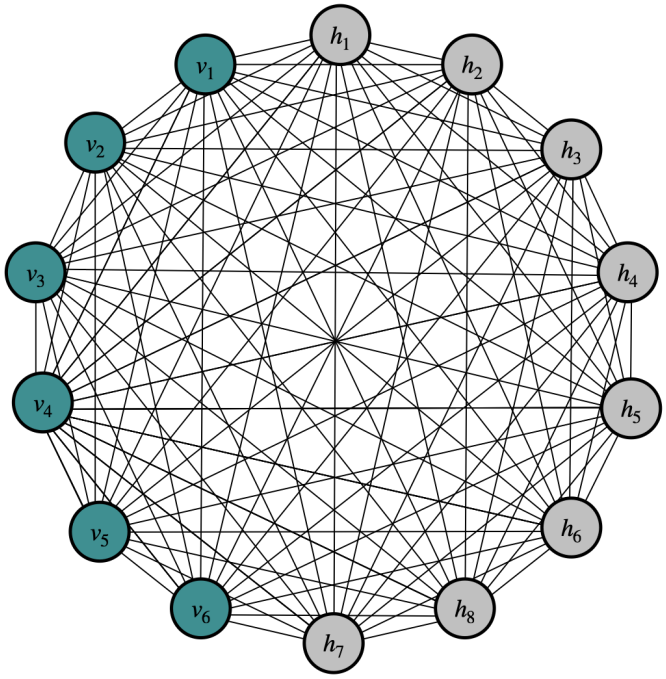

Figure 4.6: A general Boltzmann machine in which all visible (blue circles) and hidden (gray circles) nodes are fully connected (solid black lines).

organize the visible and hidden nodes into two parallel layers. The layers are joined by undirected connections, but there are no connections between individual nodes in a given layer, hence they are "restricted". This architectural arrangement improves training efficiency by leveraging highly optimized linear algebra operations.

The objective of an RBM is to reveal the underlying probability distribution of the input data, enabling applications such as dimensionality reduction and the generation of samples resembling the data. It is precisely their ability to model probability distributions that makes them a natural choice for implementing them as neural-network quantum states. Furthermore, RBMs can be stacked to construct deep belief networks, which excel in capturing hierarchical representations of complex datasets. These networks have exhibited remarkable success in addressing challenges in image and speech recognition domains, but training deep belief networks can present greater challenges compared to

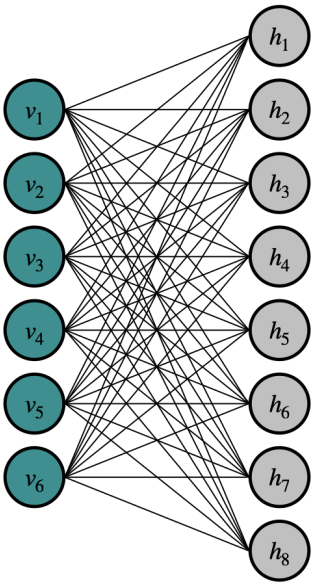

Figure 4.7: A standard restricted Boltzmann machine, where the hidden nodes (gray circles) are binary and the visible nodes (blue circles) can be either binary or Gaussian. There are no connections (solid black lines) between nodes within the same layer.

deep feedforward neural networks, which will be introduced in Sec. 4.7.2.

The most common type of RBM is a binary-binary RBM, in which all the visible and hidden nodes are discrete and constrained to the values of 0 or 1. Since we will exclusively work with real-valued inputs, our visible nodes will be taken to be Gaussian units, each with their own mean and variance. The hidden units will remain binary. The resulting formulation is called a Gaussian-binary restricted Boltzmann machine, which we will delve into shortly. In the subsequent sections, we will develop different variations of continuous restricted Boltzmann machines.

4.7.1.1 Gaussian-Binary Restricted Boltzmann Machines

We will begin by presenting the most common type of Gaussian-binary restricted Boltzmann machine (GB-RBM). The first step is to define an energy-like quantity $\mathcal{E}(\boldsymbol{v}, \boldsymbol{h})$ that describes

the interplay between the visible nodes $\boldsymbol{v} \in \mathbb{R}^V$ and hidden nodes $\boldsymbol{h} \in \mathbb{R}^H$ within the network. This quantity should not be conflated with the physical energy of our quantum system, which is why it has been assigned a distinct name $\mathcal{E}$ instead of E. The energy of the GB-RBM is defined as

$$\mathcal{E}(\boldsymbol{v}, \boldsymbol{h}) = \sum_{i=1}^{V} \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_{j=1}^{H} b_j h_j - \sum_{i=1}^{V} \sum_{j=1}^{H} \frac{v_i}{\sigma_i^2} W_{ij} h_j, \tag{4.2}$$

where a_i is the bias or mean of the ith visible node, σ_i^2 is the variance of the ith visible node, b_j is the bias for the jth hidden node, and W_{ij} is the weight between the ith visible node and jth hidden node. In many machine learning applications, it is sufficient to simply treat the variances as a constant hyperparameter $\sigma_i^2 = \sigma^2$. However, training the variances individually will be important in our application, so we reparameterize them as

$$\frac{1}{\sigma_i^2} = \exp(s_i), \tag{4.3}$$

which improves their training and ensures they are always positive. The trainable parameters are collectively denoted as $\boldsymbol{\theta} = (\boldsymbol{a}, \boldsymbol{s}, \boldsymbol{b}, W)$, where $\boldsymbol{a} \in \mathbb{R}^V$, $\boldsymbol{s} \in \mathbb{R}^V$, $\boldsymbol{b} \in \mathbb{R}^H$, and $W \in \mathbb{R}^{V \times H}$.

By taking the Boltzmann distribution, we can define a corresponding joint probability distribution

$$\mathcal{P}(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{\mathcal{Z}} e^{-\mathcal{E}(\boldsymbol{v}, \boldsymbol{h})}, \tag{4.4}$$

where the normalization constant is

$$\mathcal{Z} = \sum_{\boldsymbol{h}} \int d\boldsymbol{v} e^{-\mathcal{E}(\boldsymbol{v}, \boldsymbol{h})}. \tag{4.5}$$

This particular step justifies the inclusion of "Boltzmann" in the name of the model. We are specifically interested in the marginal probability distribution of the visible nodes, so we sum over the possible values of the hidden nodes

$$\begin{aligned}
\mathcal{P}(\boldsymbol{v}) &= \sum_{\boldsymbol{h}} \mathcal{P}(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{\mathcal{Z}} \sum_{\boldsymbol{h}} e^{-\mathcal{E}(\boldsymbol{v}, \boldsymbol{h})} \\
&= \frac{1}{\mathcal{Z}} \sum_{\boldsymbol{h}} \exp\left(-\sum_{i=1}^{V} \frac{(v_i - a_i)^2}{2\sigma_i^2} + \sum_{j=1}^{H} b_j h_j + \sum_{i=1}^{V} \sum_{j=1}^{H} \frac{v_i}{\sigma_i^2} W_{ij} h_j \right) \\
&= \frac{1}{\mathcal{Z}} \exp\left(-\sum_{i=1}^{V} \frac{(v_i - a_i)^2}{2\sigma_i^2} \right) \prod_{j=1}^{H} \sum_{h_j=0}^{1} \exp\left(b_j h_j + \sum_{i=1}^{V} \frac{v_i}{\sigma_i^2} W_{ij} h_j \right) \\
&= \frac{1}{\mathcal{Z}} \exp\left(-\sum_{i=1}^{V} \frac{(v_i - a_i)^2}{2\sigma_i^2} \right) \prod_{j=1}^{H} \left(1 + \exp\left(b_j + \sum_{i=1}^{V} \frac{v_i}{\sigma_i^2} W_{ij} \right) \right) \\
&= \frac{1}{\mathcal{Z}} \exp\left(-\sum_{i=1}^{V} \frac{(v_i - a_i)^2}{2\sigma_i^2} \right) \prod_{j=1}^{H} \Big(1 + \exp(z_j(\boldsymbol{v})) \Big),
\end{aligned} \tag{4.6}$$

where we have defined a new vector $\boldsymbol{z}(\boldsymbol{v}) \in \mathbb{R}^H$ with the elements

$$z_j(\boldsymbol{v}) = b_j + \sum_{i=1}^{V} \frac{v_i}{\sigma_i^2} W_{ij}, \tag{4.7}$$

for convenience. One of the notable strengths of a GB-RBM lies in its foundation on intuitive physical laws. Unlike various other neural network models, the GB-RBM avoids being labeled as a "black-box," as it is fairly straightforward to understand how changes in the trainable parameters $\boldsymbol{\theta}$ causes changes in the learned distributions. For example, we can interpret the marginal probability distribution of the visible nodes in Eq. (4.6) as the product of a non-interacting Gaussian part and a nonlinear interaction term.

We can similarly derive the marginal probability distribution of the hidden nodes by

integrating over the visible nodes

$$\begin{aligned}\mathcal{P}(\boldsymbol{h}) &= \int d\boldsymbol{v}\mathcal{P}(\boldsymbol{v},\boldsymbol{h}) = \frac{1}{\mathcal{Z}}\int d\boldsymbol{v}e^{-\mathcal{E}(\boldsymbol{v},\boldsymbol{h})} \\ &= \frac{1}{\mathcal{Z}}\int d\boldsymbol{v}\exp\left(-\sum_{i=1}^{V}\frac{(v_i-a_i)^2}{2\sigma_i^2}+\sum_{j=1}^{H}b_jh_j+\sum_{i=1}^{V}\sum_{j=1}^{H}\frac{v_i}{\sigma_i^2}W_{ij}h_j\right) \\ &= \frac{1}{\mathcal{Z}}\exp\left(\sum_{j=1}^{H}b_jh_j\right)\prod_{i=1}^{V}\int dv_i\exp\left(-\frac{(v_i-a_i)^2}{2\sigma_i^2}+\sum_{j=1}^{H}\frac{v_i}{\sigma_i^2}W_{ij}h_j\right).\end{aligned} \tag{4.8}$$

Defining another vector $\boldsymbol{c}(\boldsymbol{h})\in\mathbb{R}^V$ with the elements

$$c_i(\boldsymbol{h}) = a_i + \sum_{j}^{H} W_{ij}h_j, \tag{4.9}$$

and completing the square on the right-hand side of Eq. (4.8)

$$\begin{aligned}-\frac{(v_i-a_i)^2}{2\sigma_i^2}+\sum_{j=1}^{H}\frac{v_i}{\sigma_i^2}W_{ij}h_j &= -\frac{1}{2\sigma_i^2}\left((v_i-a_i)^2+\sum_{j=1}^{H}v_iW_{ij}h_j\right) \\ &= -\frac{1}{2\sigma_i^2}\left(v_i^2-2\left(a_i+\sum_{j=1}^{H}W_{ij}h_j\right)v_i+a_i^2\right) \\ &= -\frac{1}{2\sigma_i^2}\left(v_i^2-2c_i(\boldsymbol{h})v_i+a_i^2\right) \\ &= -\frac{1}{2\sigma_i^2}\left((v_i-c_i(\boldsymbol{h}))^2+a_i^2-c_i(\boldsymbol{h})^2\right)\end{aligned} \tag{4.10}$$

yields

$$\begin{aligned}\mathcal{P}(\boldsymbol{h}) &= \frac{1}{\mathcal{Z}}\exp\left(\sum_{j=1}^{H}b_jh_j\right)\prod_{i=1}^{V}\int dv_i\exp\left(-\frac{(v_i-c_i(\boldsymbol{h}))^2+a_i^2-c_i(\boldsymbol{h})^2}{2\sigma_i^2}\right) \\ &= \frac{1}{\mathcal{Z}}\exp\left(\sum_{j=1}^{H}b_jh_j\right)\prod_{i=1}^{V}\exp\left(\frac{c_i(\boldsymbol{h})^2-a_i^2}{2\sigma_i^2}\right)\sqrt{2\pi}\sigma_i.\end{aligned} \tag{4.11}$$

The marginal distribution of the hidden nodes is yet again another product of non-interacting and interacting parts. From here, we can obtain the conditional probabilities of the visible nodes $\mathcal{P}(\boldsymbol{v}|\boldsymbol{h})$ and hidden nodes $\mathcal{P}(\boldsymbol{h}|\boldsymbol{v})$ by dividing the joint probability distribution $\mathcal{P}(\boldsymbol{v},\boldsymbol{h})$ by $\mathcal{P}(\boldsymbol{h})$ and $\mathcal{P}(\boldsymbol{v})$, respectively. For the former, we find a product of normalized Gaussian distributions,

$$\begin{aligned}
\mathcal{P}(\boldsymbol{v}|\boldsymbol{h}) &= \frac{\mathcal{P}(\boldsymbol{v},\boldsymbol{h})}{\mathcal{P}(\boldsymbol{h})} \\
&= \frac{\exp\left(-\sum_{i=1}^{V}\frac{(v_i-a_i)^2}{2\sigma_i^2}+\sum_{i=1}^{V}\sum_{j=1}^{H}\frac{v_i}{\sigma_i^2}W_{ij}h_j\right)}{\prod_{i=1}^{V}\exp\left(\frac{c_i(\boldsymbol{h})^2-a_i^2}{2\sigma_i^2}\right)\sqrt{2\pi}\sigma_i} \\
&= \prod_{i=1}^{V}\frac{1}{\sqrt{2\pi}\sigma_i}\exp\left(-\frac{(v_i-c_i(\boldsymbol{h}))^2+a_i^2-c_i(\boldsymbol{h})^2}{2\sigma_i^2}-\frac{c_i(\boldsymbol{h})^2-a_i^2}{2\sigma_i^2}\right) \\
&= \prod_{i=1}^{V}\frac{1}{\sqrt{2\pi}\sigma_i}\exp\left(-\frac{(v_i-c_i(\boldsymbol{h}))^2}{2\sigma_i^2}\right) \\
&= \prod_{i=1}^{V}\mathcal{N}(v_i;c_i(\boldsymbol{h}),\sigma_i^2),
\end{aligned} \tag{4.12}$$

with their individual means $c_i(\boldsymbol{h})$ and variances of σ_i^2. For the latter we have

$$\mathcal{P}(\boldsymbol{h}|\boldsymbol{v}) = \frac{\mathcal{P}(\boldsymbol{v},\boldsymbol{h})}{\mathcal{P}(\boldsymbol{v})} = \frac{\exp\left(\sum_{j=1}^{H}b_jh_j+\sum_{i=1}^{V}\sum_{j=1}^{H}\frac{v_i}{\sigma_i^2}W_{ij}h_j\right)}{\prod_{j=1}^{H}\left(1+e^{z_j(\boldsymbol{v})}\right)} = \prod_{j=1}^{H}\frac{e^{z_j(\boldsymbol{v})h_j}}{1+e^{z_j(\boldsymbol{v})}}, \tag{4.13}$$

which implies that the conditional probability of the jth hidden node being active is given by

$$\mathcal{P}(h_j=1|\boldsymbol{v}) = \frac{e^{z_j(\boldsymbol{v})}}{1+e^{z_j(\boldsymbol{v})}} = \frac{1}{1+e^{-z_j(\boldsymbol{v})}} = \sigma(z_j(\boldsymbol{v})), \tag{4.14}$$

where

$$\sigma(x) \equiv \frac{1}{1+e^{-x}} \tag{4.15}$$

is the sigmoid function. In a sense, the binary nodes act as mediators for the interaction, switching between "on" and "off" states with a probability determined by the configuration of the visible nodes.

In standard unsupervised learning problems, the parameters $\boldsymbol{\theta}$ of a Restricted Boltzmann machine are trained by maximizing the log-likelihood $\log P(\boldsymbol{v})$. The gradients of the log-likelihood with respect to the parameters can be approximated using the Contrastive Divergence method and Gibbs sampling, increasing the efficiency of the training process significantly. However, in our application, the parameters will be trained by minimizing the expectation value of the physical energy of our quantum system, a major departure from traditional machine learning problems. Nonetheless, it will be useful to compute the gradients of the log-likelihood.

In terms of the parameters $\boldsymbol{\theta} = (\boldsymbol{a}, \boldsymbol{s}, \boldsymbol{b}, W)$, the log-likelihood reads

$$\log \mathcal{P}(\boldsymbol{v}) = -\frac{1}{2}\sum_{i=1}^{V} \exp(s_i)(v_i - a_i)^2 + \sum_{j=1}^{H} f(z_j(\boldsymbol{v})) - \log \mathcal{Z}, \tag{4.16}$$

where $z_j(\boldsymbol{v})$ was defined in Eq. (4.7) and is dependent on $\boldsymbol{s}, \boldsymbol{b}$, and W, while $f(x)$ is the "softplus" function

$$f(x) = \log\left(1 + \exp(x)\right). \tag{4.17}$$

It will become clear why we write the log-likelihood in terms of $f(x)$ when we develop generalizations of the GB-RBM. The derivatives of the log-likelihood with respect to the

parameters are then given by

$$\frac{\partial}{\partial a_i}\log\mathcal{P}(\boldsymbol{v}) = \exp(s_i)(v_i - a_i), \tag{4.18}$$

$$\frac{\partial}{\partial s_i}\log\mathcal{P}(\boldsymbol{v}) = -\frac{1}{2}\exp(s_i)(v_i - a_i)^2 + \exp(s_i)v_i\sum_{j=1}^{H} W_{ij}f'(z_j(\boldsymbol{v})), \tag{4.19}$$

$$\frac{\partial}{\partial b_j}\log\mathcal{P}(\boldsymbol{v}) = f'(z_j(\boldsymbol{v})), \tag{4.20}$$

$$\frac{\partial}{\partial W_{ij}}\log\mathcal{P}(\boldsymbol{v}) = \exp(s_i)v_i f'(z_j(\boldsymbol{v})), \tag{4.21}$$

where $f'(x)$ is the same as the sigmoid function $\sigma(x)$ defined in Eq. (4.15).

4.7.1.2 Multivariate Gaussian-Binary Restricted Boltzmann Machines

We will now move beyond the standard formulation of the Gaussian-Binary restricted Boltzmann machine, and introduce correlations among the Gaussian visible nodes. The resulting network is a Multivariate Gaussian-Binary restricted Boltzmann machine (mGB-RBM), which is halfway between the standard GB-RBM and a general RBM. We will still label this network as "restricted" because there are no connections between the hidden nodes. The energy of the mGB-RBM can be written as

$$\mathcal{E}(\boldsymbol{v},\boldsymbol{h}) = \frac{1}{2}(\boldsymbol{v}-\boldsymbol{a})^T\Sigma^{-1}(\boldsymbol{v}-\boldsymbol{a}) - \boldsymbol{b}^T\boldsymbol{h} - \boldsymbol{v}^T\Sigma^{-1}W\boldsymbol{h}, \tag{4.22}$$

where we have used the covariance matrix

$$\Sigma = \mathrm{Cov}(\boldsymbol{v}, \boldsymbol{v}) = \begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2 & \cdots & \sigma_1\sigma_V \\ \sigma_2\sigma_1 & \sigma_2^2 & \cdots & \sigma_2\sigma_V \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_V\sigma_1 & \sigma_V\sigma_2 & \cdots & \sigma_V^2 \end{bmatrix} \in \mathbb{R}^{V\times V}, \tag{4.23}$$

in addition to the usual biases $\boldsymbol{a} \in \mathbb{R}^V$, $\boldsymbol{b} \in \mathbb{R}^H$ and weights $W \in \mathbb{R}^{V\times H}$. Following the previous reparameterizing of the variances in Eq. (4.3), we write the diagonal elements of the inverse covariance matrix as

$$\Sigma_{ii}^{-1} = \exp(S_{ii}), \tag{4.24}$$

and the offdiagonal elements as,

$$\Sigma_{ij}^{-1} = S_{ij}, \text{ for } i \neq j, \tag{4.25}$$

for some matrix $S \in \mathbb{R}^{V\times V}$. This choice is intended to avoid explicitly computing inverses of matrices and to guarantee the invertibility of the inverse covariance matrix Σ^{-1}. However, there is still a small risk that the variances corresponding to this reparameterization become negative, so we bias against this possibility by randomly initializing the diagonal elements S_{ii} to small positive values, and the off diagonal elements $S_{ij},\ j \neq i$, to small negative values. This initialization results in a covariance matrix Σ close to a diagonal matrix with positive covariances. The covariance matrix (and its inverse) are symmetric, so only the upper triangular part of the matrix S, denoted as $\mathrm{triu}(S)$, is required. Then the trainable parameters are collectively represented as $\boldsymbol{\theta} = (\boldsymbol{a}, \mathrm{triu}(S), \boldsymbol{b}, W)$.

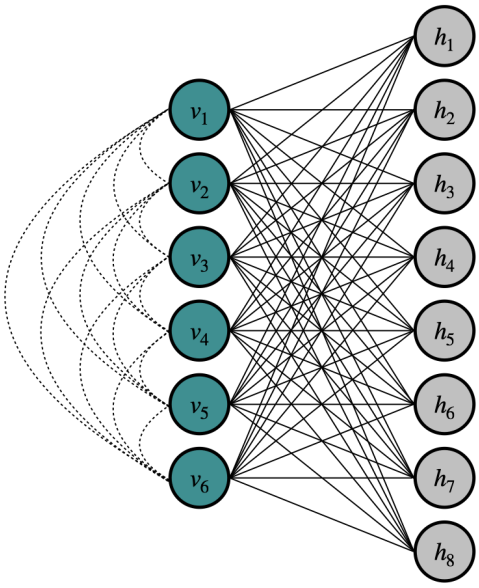

Figure 4.8: A multivariate Gaussian-binary restricted Boltzmann machine, a step between a standard Gaussian-binary restricted Boltzmann machine and a general Boltzmann machine. In addition to the connections (solid black lines) between the visible nodes (blue circles) and hidden nodes (gray circles), there are connections (dotted black lines) among the visible nodes.

The calculations of the joint, marginal, and conditional probability distributions will follow the same process as before. Having previously discussed the process extensively, we will simply list the final results. The joint probability distribution corresponding to the energy in Eq. (4.22) is written as

$$\mathcal{P}(\boldsymbol{v},\boldsymbol{h}) = \exp\left(-\frac{1}{2}(\boldsymbol{v}-\boldsymbol{a})^T\Sigma^{-1}(\boldsymbol{v}-\boldsymbol{a}) + \boldsymbol{b}^T\boldsymbol{h} + \boldsymbol{v}^T\Sigma^{-1}W\boldsymbol{h}\right). \tag{4.26}$$

Summing over the hidden nodes yields the marginal probability distribution of the visible nodes

$$\mathcal{P}(\boldsymbol{v}) = \frac{1}{\mathcal{Z}}\exp\left(-\frac{1}{2}(\boldsymbol{v}-\boldsymbol{a})^T\Sigma^{-1}(\boldsymbol{v}-\boldsymbol{a})\right)\prod_{j=1}^{H}\left(1+\exp\left(z_j(\boldsymbol{v})\right)\right), \tag{4.27}$$

where

$$\boldsymbol{z}(\boldsymbol{v}) = \boldsymbol{b} + W^T\Sigma^{-1}\boldsymbol{v} \in \mathbb{R}^H. \tag{4.28}$$

Unlike the standard GB-RBM, the distribution above does not contain a non-interacting part, as the Gaussian nodes are not independent (unless Σ is diagonal). Integrating the joint distribution over the visible nodes gives the marginal distribution over the hidden nodes

$$\mathcal{P}(\boldsymbol{h}) = \frac{1}{\mathcal{Z}}\exp\left(\boldsymbol{b}^T\boldsymbol{h} + \frac{1}{2}\boldsymbol{c}^T\Sigma\boldsymbol{c} - \frac{1}{2}\boldsymbol{a}^T\Sigma\boldsymbol{a}\right)\sqrt{(2\pi)^V|\Sigma|}, \tag{4.29}$$

where

$$\boldsymbol{c}(\boldsymbol{h}) = \boldsymbol{a} + W\boldsymbol{h}. \tag{4.30}$$

The conditional probability distribution of the visible nodes is no longer a product of independent Gaussians, but a V-dimensional multivariate Gaussian

$$\mathcal{P}(\boldsymbol{v}|\boldsymbol{h}) = \frac{\mathcal{P}(\boldsymbol{v},\boldsymbol{h})}{\mathcal{P}(\boldsymbol{h})} = \mathcal{N}(\boldsymbol{v};\boldsymbol{c}(\boldsymbol{h}),\Sigma). \tag{4.31}$$

Meanwhile, the conditional probability distribution of the hidden nodes is the same

$$\mathcal{P}(\boldsymbol{h}|\boldsymbol{v}) = \frac{\mathcal{P}(\boldsymbol{v},\boldsymbol{h})}{\mathcal{P}(\boldsymbol{v})} = \prod_{j=1}^{H} \frac{e^{z_j(\boldsymbol{v})h_j}}{1+e^{z_j(\boldsymbol{v})}}, \quad \mathcal{P}(h_j=1|\boldsymbol{v}) = \sigma(z_j(\boldsymbol{v})), \tag{4.32}$$

as long as we use our new definition of $\boldsymbol{z}(\boldsymbol{v})$ from Eq. (4.28). Finally, the log-likelihood is given by

$$\log \mathcal{P}(\boldsymbol{v}) = -\frac{1}{2}(\boldsymbol{v}-\boldsymbol{a})^T\Sigma^{-1}(\boldsymbol{v}-\boldsymbol{a}) + \sum_{j=1}^{H} f(z_j(\boldsymbol{v})) - \log \mathcal{Z}, \tag{4.33}$$

and its derivatives with respect to the parameters are

$$\frac{\partial}{\partial \boldsymbol{a}} \log \mathcal{P}(\boldsymbol{v}) = \Sigma^{-1}(\boldsymbol{v}-\boldsymbol{a}), \tag{4.34}$$

$$\frac{\partial}{\partial S} \log \mathcal{P}(\boldsymbol{v}) = \Big(-\frac{1}{2}(\boldsymbol{v}-\boldsymbol{a})(\boldsymbol{v}-\boldsymbol{a})^T + W\boldsymbol{f}'(\boldsymbol{z}(\boldsymbol{v}))\boldsymbol{v}^T \Big) \circ \frac{\partial \Sigma^{-1}}{\partial S}, \tag{4.35}$$

$$\frac{\partial}{\partial \boldsymbol{b}} \log \mathcal{P}(\boldsymbol{v}) = \boldsymbol{f}'(\boldsymbol{z}(\boldsymbol{v})), \tag{4.36}$$

$$\frac{\partial}{\partial W} \log \mathcal{P}(\boldsymbol{v}) = \boldsymbol{v}\Sigma^{-1}\left(\boldsymbol{f}'(\boldsymbol{z}(\boldsymbol{v}))\right)^T, \tag{4.37}$$

where $\boldsymbol{f}'(\boldsymbol{z}(\boldsymbol{v}))$ denotes the element-wise application of the derivative of the softplus function (Eq. (4.17)), also known as the sigmoid function, and $\circ$ denotes element-wise multiplication. In addition, $\frac{\partial \Sigma^{-1}}{\partial S}$ is a matrix filled with ones except for the diagonal, which is filled with $\exp(S_{ii})$.

There are clearly many similarities between the standard GB-RBM and the

multivariate GB-RBM. However, the inclusion of covariances in the latter enhances its flexibility compared to the former, and makes computing and writing down the various probabilities simpler. Additionally, the multivariate GB-RBM offers the advantage of performing most computations through matrix-matrix multiplication or matrix-vector multiplication, enabling more efficient implementations.

Before moving onto another step in generalization, let us consider one slight variation of the Multivariate Gaussian-Binary restricted Boltzmann machine, in which the binary nodes are allowed to take the values of -1 and 1, instead of 0 and 1. We can take the same energy function as Eq. (4.22), so that our joint probability distribution $\mathcal{P}(\boldsymbol{v}, \boldsymbol{h})$ is also the same. However, when we sum over the hidden nodes to compute the marginal distribution of the visible nodes, we obtain

$$\begin{aligned}\mathcal{P}(\boldsymbol{v}) &= \frac{1}{\mathcal{Z}} \exp\left(-\frac{1}{2}(\boldsymbol{v}-\boldsymbol{a})^T\Sigma^{-1}(\boldsymbol{v}-\boldsymbol{a})\right) \prod_{j=1}^{H}\Big(\exp(z_j(\boldsymbol{v})) + \exp(-z_j(\boldsymbol{v}))\Big) \\ &= \frac{1}{\mathcal{Z}} \exp\left(-\frac{1}{2}(\boldsymbol{v}-\boldsymbol{a})^T\Sigma^{-1}(\boldsymbol{v}-\boldsymbol{a})\right) 2^H \prod_{j=1}^{H}\cosh\big(z_j(\boldsymbol{v})\big)\,,\end{aligned} \tag{4.38}$$

instead of Eq. (4.27). Then the conditional probability distribution for the hidden nodes becomes

$$\begin{aligned}\mathcal{P}(\boldsymbol{h}|\boldsymbol{v}) &= \frac{\mathcal{P}(\boldsymbol{v},\boldsymbol{h})}{\mathcal{P}(\boldsymbol{v})} = \frac{1}{2^H}\prod_{j=1}^{H}\frac{\exp(z_j(\boldsymbol{v})h_j)}{\cosh(z_j(\boldsymbol{v}))}, \\ \mathcal{P}(h_j=1|\boldsymbol{v}) &= \frac{1}{2}\Big(\tanh(z_j(\boldsymbol{v}))+1\Big).\end{aligned} \tag{4.39}$$

In Figure 4.9, we compare the form of $P(h_j = 1|\boldsymbol{v})$ given above, with the one obtained in Eq. (4.32). When the hidden nodes were allowed the take the values of 0 or 1, we found that $P(h_j = 1|\boldsymbol{v})$ was given by the sigmoid function. When we instead allowed the hidden nodes

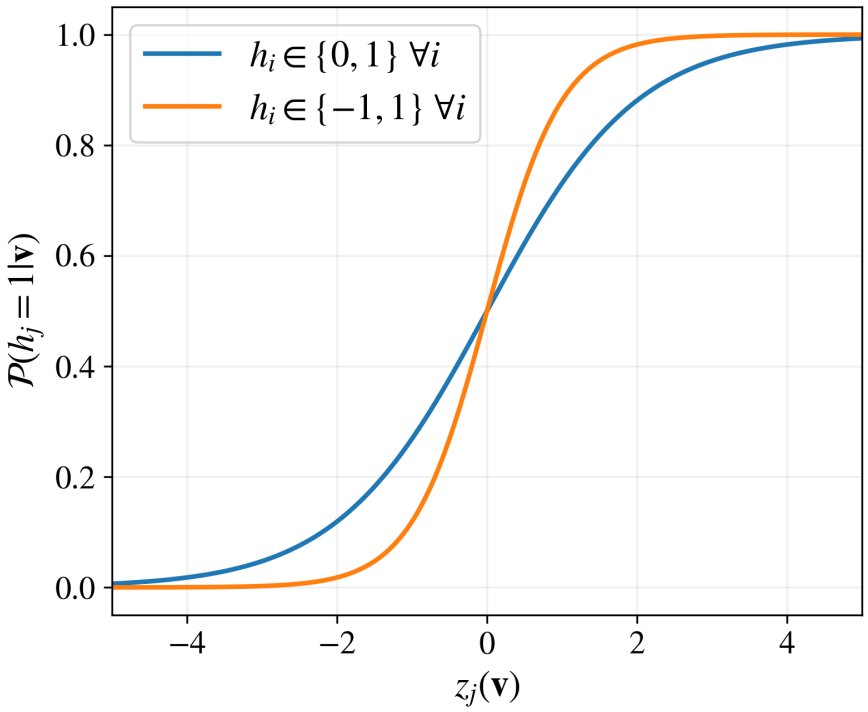

Figure 4.9: The conditional probability of a single binary hidden node h_j activating, given the state of the visible nodes $\boldsymbol{v}$, for a multivariate Gaussian-binary restricted Boltzmann machine. The blue line represents the probability if the hidden nodes are allowed to take values of 0 and 1 (Eq. (4.32)), while the orange line represents the probability of the hidden nodes are allowed to take values of -1 and 1 (Eq. (4.39)). The x-axis is a transformation of the visible nodes given by Eq. (4.28).

to take the values of -1 or 1, we obtained the hyperbolic tangent function, normalized to be between 0 and 1. The results are surprisingly intuitive.

To finish off our calculations, we write down the new log-likelihood

$$\log \mathcal{P}(\boldsymbol{v}) = -\frac{1}{2}(\boldsymbol{v} - \boldsymbol{a})^T \exp(S)(\boldsymbol{v} - \boldsymbol{a}) + \sum_{j=1}^{H} f(z_j(\boldsymbol{v})) - \log \mathcal{Z}', \tag{4.40}$$

where $\mathcal{Z}'$ contains the extra factor of 2^H from Eq. (4.38), and $f(x)$ from Eq. (4.17) is replaced

by

$$f(x) = \log(\cosh(x)), \tag{4.41}$$

which we call the "log-cosh" function[1]. With this redefinition of $f(x)$, the derivatives of the log-likelihood take the same form as before, in Eq. (4.37).

4.7.1.3 Multivariate Gaussian-Uniform Restricted Boltzmann Machines

Another variation of the continuous RBM involves allowing the hidden nodes to be continuous rather than discrete. We will name this variation the multivariate Gaussian-Uniform Restricted Boltzmann Machine (mGU-RBM), as we let the hidden nodes take any between 0 and 1. Using the same form of the energy as Eq. (4.22), the marginal distribution of the visible nodes becomes

$$\begin{aligned}
\mathcal{P}(\boldsymbol{v}) &= \int_0^1 d\boldsymbol{h} e^{-\mathcal{E}(\boldsymbol{v},\boldsymbol{h})} \\
&= \frac{1}{\mathcal{Z}} \exp\left(-\frac{1}{2}(\boldsymbol{v}-\boldsymbol{a})^T \Sigma^{-1} (\boldsymbol{v}-\boldsymbol{a})\right) \prod_{j=1}^{H} \int_0^1 dh_j \exp(z_j(\boldsymbol{v}) h_j) \\
&= \frac{1}{\mathcal{Z}} \exp\left(-\frac{1}{2}(\boldsymbol{v}-\boldsymbol{a})^T \Sigma^{-1} (\boldsymbol{v}-\boldsymbol{a})\right) \prod_{j=1}^{H} \frac{\exp(z_j(\boldsymbol{v})) - 1}{z_j(\boldsymbol{v})},
\end{aligned} \tag{4.42}$$

[1] Just as the softplus function, Eq. (4.17), is a smoothly differentiable alternative to the Rectified Linear Unit (ReLU), a non-continuously differentiable activation function commonly used for feedforward neural networks, the log-cosh function, Eq. (4.41), is a smoothly differentiable alternative to the mean-absolute-error (MAE), a non-continuously differentiable cost function that tends to prevent overfitting.

while the marginal distribution of the hidden nodes remains the same as before. Then for the conditional probabilities for the hidden units, we obtain

$$\begin{aligned} \mathcal{P}(\boldsymbol{h}|\boldsymbol{v}) &= \frac{\mathcal{P}(\boldsymbol{v},\boldsymbol{h})}{\mathcal{P}(\boldsymbol{v})} = \prod_{j=1}^{H} \frac{z_j(\boldsymbol{v}) \exp(z_j(\boldsymbol{v}) h_j)}{\exp(z_j(\boldsymbol{v})) - 1}, \\ \mathcal{P}(h_j = 1|\boldsymbol{v}) &= \frac{z_j(\boldsymbol{v})}{1 - \exp(-z_j(\boldsymbol{v}))}. \end{aligned} \tag{4.43}$$

The derivatives of the log-likelihood can remain in the same form as Eq. (4.37) by redefining $f(x)$ as

$$f(x) = \log \frac{\exp(x) - 1}{x}, \tag{4.44}$$

which we will name the "leaky-softplus" function. Even though the limit of this function exists mathematically as $|x| \to 0$, we may encounter instabilities near $x = 0$ when implemented computationally. Therefore, when $|x|$ is small, we replace $f(x)$ and $f'(x)$ with their Taylor expansions up to first order in x.

Now for the final variation of the mGU-RBM, we allow the hidden nodes to take any value between -1 and 1,

$$\begin{aligned} \mathcal{P}(\boldsymbol{v}) &= \int_0^1 d\boldsymbol{h} e^{-\mathcal{E}(\boldsymbol{v},\boldsymbol{h})} \\ &= \frac{1}{\mathcal{Z}} \exp\left(-\frac{1}{2}(\boldsymbol{v}-\boldsymbol{a})^T \Sigma^{-1} (\boldsymbol{v}-\boldsymbol{a})\right) \prod_{j=1}^{H} \int_{-1}^{1} dh_j \exp(z_j(\boldsymbol{v}) h_j) \\ &= \frac{1}{\mathcal{Z}} \exp\left(-\frac{1}{2}(\boldsymbol{v}-\boldsymbol{a})^T \Sigma^{-1} (\boldsymbol{v}-\boldsymbol{a})\right) \prod_{j=1}^{H} \frac{\exp(z_j(\boldsymbol{v})) - \exp(-z_j(\boldsymbol{v}))}{z_j(\boldsymbol{v})} \\ &= \frac{1}{\mathcal{Z}} \exp\left(-\frac{1}{2}(\boldsymbol{v}-\boldsymbol{a})^T \Sigma^{-1} (\boldsymbol{v}-\boldsymbol{a})\right) 2^H \prod_{j=1}^{H} \frac{\sinh(z_j(\boldsymbol{v}))}{z_j(\boldsymbol{v})}. \end{aligned} \tag{4.45}$$

The conditional probabilities for the hidden units are given by

$$\begin{aligned}\mathcal{P}(\boldsymbol{h}|\boldsymbol{v}) &= \frac{\mathcal{P}(\boldsymbol{v},\boldsymbol{h})}{\mathcal{P}(\boldsymbol{v})} = \frac{1}{2^H}\prod_{j=1}^{H}\frac{z_j(\boldsymbol{v}))\exp(z_j(\boldsymbol{v})h_j)}{\sinh(z_j(\boldsymbol{v}))},\\ \mathcal{P}(h_j=1|\boldsymbol{v}) &= \frac{1}{2}z_j(\boldsymbol{v})\Big(\coth(z_j(\boldsymbol{v}))+1\Big),\end{aligned} \tag{4.46}$$

where $\mathcal{P}(h_j = 1|\boldsymbol{v})$ for the two variations of the mGU-RBM will be plotted alongside the softplus function in Fig. 4.10. Notice that both variations of the mGU-RBM involve a conditional probability that is unbounded from above, as opposed to the mGB-RBMs, which produce a conditional probability that is bounded between 0 and 1. To interpret the conditional probabilities we obtained for the mGU-RBMs as true probabilities, let us assume all values are capped at 1.

Again, the derivatives of the log-likelihood can remain in the same form as Eq. (4.37) by redefining $f(x)$ as

$$f(x) = \log\frac{\sinh(x)}{x}, \tag{4.47}$$

which we will name the "log-sinhc" function, inspired by the "sine cardinal" function defined as $\mathrm{sinc}(x) \equiv \sin(x)/x$. As before, we handle the singularity at $x = 0$ by replacing $f(x)$ and $f'(x)$ by their first-order Taylor expansions.

To end our discussion on the various continuous restricted Boltzmann machines, let us compare the different definitions of $f(x)$ that appear in the log-likelihood. In Fig. 4.11, we plot the softplus, log-cosh, leaky-softplus, and log-sinhc functions, defined in Eqs. (4.17), (4.41), (4.44), and (4.47). The leaky-softplus function is the only function that permits negative values, hence the label "leaky". Both softplus and log-cosh scale as x for large, positive x, while leaky-softplus and log-sinhc scale as $x - \log(x)$. Finally, log-cosh

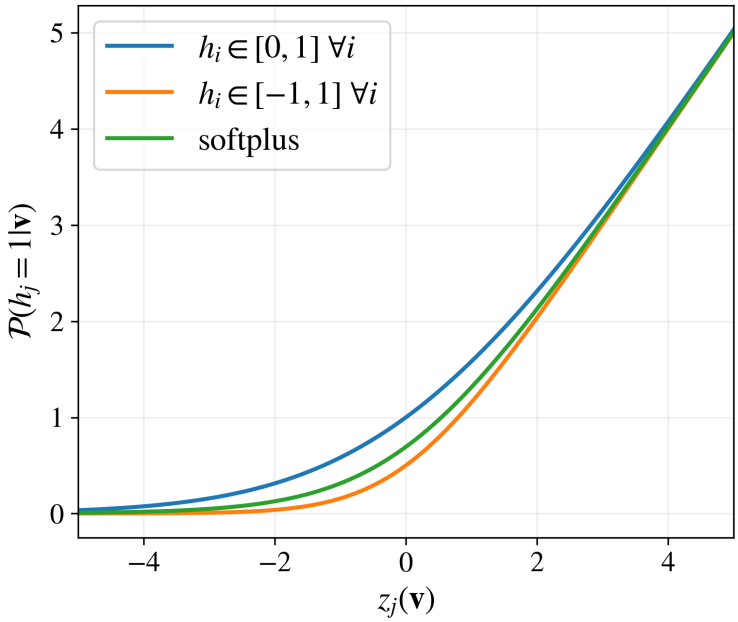

Figure 4.10: The conditional probability of a single uniform node h_j activating, given the state of the visible nodes $\boldsymbol{v}$, for a multivariate Gaussian-uniform restricted Boltzmann machine. The blue line represents the probability if the uniform nodes are allowed to take values between 0 and 1 (Eq. (4.43)), while the orange line represents the probability of the hidden nodes are allowed to take values of -1 and 1 (Eq. (4.46)). The x-axis is a transformation of the visible nodes given by Eq. (4.28). We also plot the softplus function (Eq. (4.17)) for comparison, as its exhibits similar character. Probability values greater than 1 are assumed to mean the hidden node h_j is always activated.

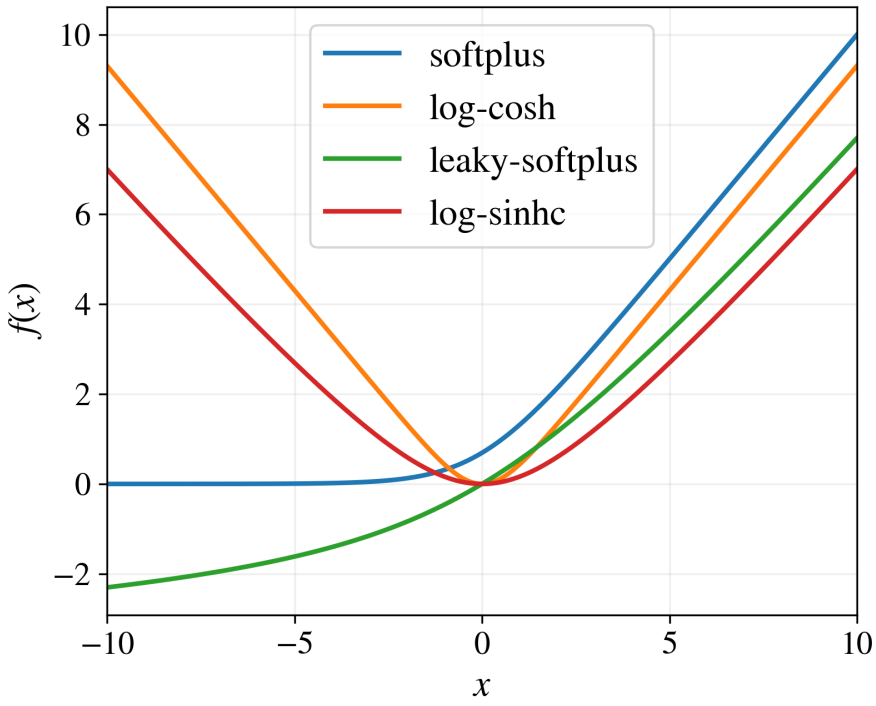

Figure 4.11: Comparison of the four functions we called $f(x)$ during our derivations of the multivariate Gaussian-binary and Gaussian-uniform restricted Boltzmann machines (Eqs. (4.17), (4.41), (4.44), (4.47)). These functions appear in the log-likelihoods $\log P(\boldsymbol{v})$.

and log-sinhc are both even functions, just like the interval for which the hidden nodes were defined during their construction.

4.7.2 Feedforward Neural Networks

Feedforward neural networks (FNNs) are widely recognized as one of the most prominent forms of artificial neural networks, primarily due to their training simplicity, scalability, and generality. As a result, they have been used to solve a diverse set of complex problems within all three branches of machine learning. However, in contrast to more interpretable models like restricted Boltzmann machines, feedforward neural networks are often characterized as black boxes, as their inner workings are difficult to interpret. Feedforward neural networks

are so successful, in fact, that it is recommended to use them only when low-bias models are required and a large data set is available. Otherwise, they can be prone to overfitting, especially in supervised learning problems.

A deep feedforward neural network is constructed by alternating compositions of affine transformations and simple, nonlinear transformations. It consists of an input layer, at least one hidden layer, and an explicit output layer, all of which are densely-connected to the adjacent layers only. As the name implies, the information flows only in one direction, starting from the input layer, and ending at the output layer. For an FNN with L layers (including the output layer, but excluding the input layer), we construct the nodes as

$$\boldsymbol{h}^{(0)} = \boldsymbol{v}, \tag{4.48}$$

$$\boldsymbol{h}^{(\ell)} = \boldsymbol{f}_\ell \left(W^{(\ell)} \boldsymbol{h}^{(\ell-1)} + \boldsymbol{b}^{(\ell)} \right), \quad \text{for } \ell = 1, ..., L, \tag{4.49}$$

where $\boldsymbol{v}$ are the visible nodes for the input layer $\boldsymbol{h}^{(0)}$, $\boldsymbol{h}^{(\ell)}$ are the hidden layers for $\ell = 1, ..., L-1$, $\boldsymbol{h}^{(L)}$ is the output layer, and $\boldsymbol{f}_\ell$ represent the element-wise application of the nonlinear activation function f_ℓ. Each of the hidden layers have dimension H_ℓ, meaning $W^{(\ell)}$ is an $H_\ell \times H_{\ell-1}$ weight matrix and $\boldsymbol{b}^{(\ell)}$ is an H_ℓ-dimensional bias vector. The activation functions f_ℓ introduce nonlinearity to the network, enabling it to learn and approximate complex functions. Without at least one nonlinear activation function, the neural network reduces to a simple linear model.

The output layer of an FNN is commonly distinguished from the hidden layers for two main reasons. Firstly, the output dimension is typically determined by the problem's structure, rather than being a hyperparameter that can be freely chosen. Secondly, the choice of activation function for the output layer is often different from that of the hidden

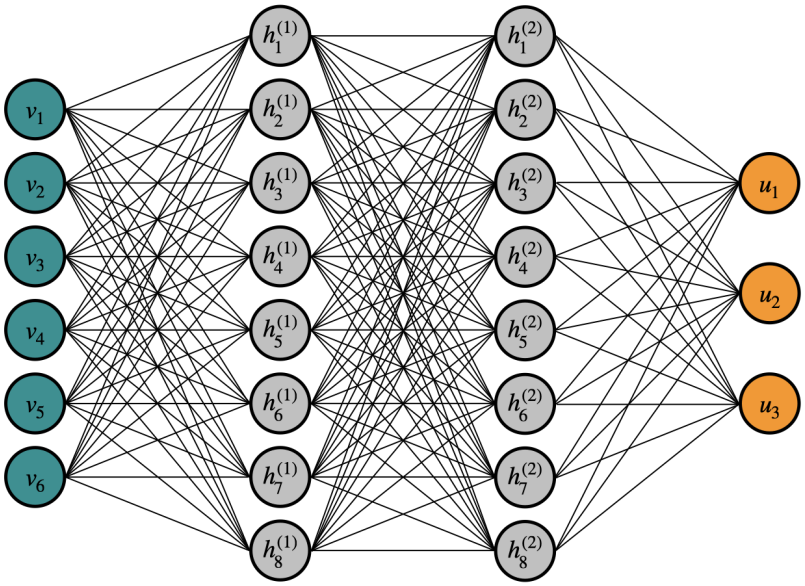

Figure 4.12: A deep feedforward neural network with six inputs (blue circles) and three outputs (orange circles). There are two hidden layers with eight nodes each (gray circles). Adjacent layers are connected by directed connections, as information flows from left to right. Arrowheads for the directed connections (solid black lines) are omitted for visual brevity.

layers. To maintain simplicity in notation, we adopt the indexing notation f_ℓ, where ℓ corresponds to the layer, to represent the activation functions rather than explicitly singling out the output layer. In regression problems, the final activation function f_L is typically linear, while binary-classification problems commonly employ sigmoid or hyperbolic tangent functions. When it comes to the hidden layers, popular choices for activation functions include the sigmoid function, hyperbolic tangent, and the Rectified Linear Unit (ReLU). However, our specific applications require unbounded, twice-continuously differentiable functions due to the evaluation of the local energy. Consequently, we prefer activation functions such as the Gaussian Error Linear Unit (GELU) and the Softplus function.

4.7.2.1 Backpropagation

One of the most attractive features of a deep feedforward neural network is its remarkable ease of training. Since we have written the forward propagation steps in Eq. (4.49) recursively, we can write the backpropagation steps using the chain rule. The derivatives of the output layer with respect to the weights and biases in layer ℓ are

$$\frac{\partial \boldsymbol{h}^{(L)}}{\partial W^{(\ell)}} = \frac{\partial \boldsymbol{h}^{(L)}}{\partial \boldsymbol{h}^{(L-1)}} \frac{\partial \boldsymbol{h}^{(L-1)}}{\partial \boldsymbol{h}^{(L-2)}} \cdots \frac{\partial \boldsymbol{h}^{(\ell+1)}}{\partial \boldsymbol{h}^{(\ell)}} \frac{\partial \boldsymbol{h}^{(\ell)}}{\partial W^{(\ell)}}, \tag{4.50}$$

$$\frac{\partial \boldsymbol{h}^{(L)}}{\partial \boldsymbol{b}^{(\ell)}} = \frac{\partial \boldsymbol{h}^{(L)}}{\partial \boldsymbol{h}^{(L-1)}} \frac{\partial \boldsymbol{h}^{(L-1)}}{\partial \boldsymbol{h}^{(L-2)}} \cdots \frac{\partial \boldsymbol{h}^{(\ell+1)}}{\partial \boldsymbol{h}^{(\ell)}} \frac{\partial \boldsymbol{h}^{(\ell)}}{\partial \boldsymbol{b}^{(\ell)}}. \tag{4.51}$$

To simplify notation, we define the intermediate vector

$$\boldsymbol{g}^{(\ell)} \equiv W^{(\ell)} \boldsymbol{h}^{(\ell-1)} + \boldsymbol{b}^{(\ell)}, \tag{4.52}$$

which represents the linear transformation only. Then evaluating each term in Eqs. (4.50) and (4.51), we have

$$\frac{\partial h_i^{(\ell)}}{\partial h_j^{(\ell-1)}} = f'_\ell \left(g_i^{(\ell)} \right) W_{ij}^{(\ell)}, \tag{4.53}$$

$$\frac{\partial h_i^{(\ell)}}{W_{jk}^{(\ell)}} = f'_\ell \left(g_i^{(\ell)} \right) \delta_{ij} h_k^{(\ell-1)}, \tag{4.54}$$

$$\frac{\partial h_i^{(\ell)}}{b_j^{(\ell)}} = f'_\ell \left(g_i^{(\ell)} \right) \delta_{ij}, \tag{4.55}$$

where δ_{ij} is the Kronecker delta. Proceeding in this way, we can propagate the error from the cost function $\frac{\partial C}{\partial \boldsymbol{h}^{(L)}}$ all the way back to the input layer, updating all the weights and biases along the way.

4.7.2.2 Universal Approximation Theorem

Another contributing factor to the widespread adoption and success of feedforward neural networks is the Universal Approximation Theorem. In simple terms, the theorem states that a feedforward neural network with a single hidden layer, also known as a shallow neural network, can approximate any continuous function on a compact subset of Euclidean space, given there are a sufficient number of hidden nodes and the activation function satisfies certain conditions. This theorem was originally proven for the sigmoid activation function, and since then, there have been efforts to prove the theorem for a number of different activation functions, network depths, and even different types of networks entirely.

We will accept the Universal Approximation Theorem as a foundational principle without proof, although we should exercise caution in its application. While the theorem establishes the existence of a satisfactory approximation for any continuous function using a shallow neural network, it does not guarantee that the required number of hidden nodes scales efficiently. Therefore, applying the theorem in practice is not as straightforward as simply increasing the size of our network. We must also consider other factors that can enhance training stability, capture additional correlations, and improve precision. While the Universal Approximation Theorem has not been formally proven for more complex network architectures, in practical applications, the effectiveness and versatility of these architectures are widely acknowledged nonetheless.

4.7.3 Deep Sets

One of the limitations of a standard feedforward neural network is the requirement to organize inputs into a single vector. This poses a challenge when dealing with inputs such

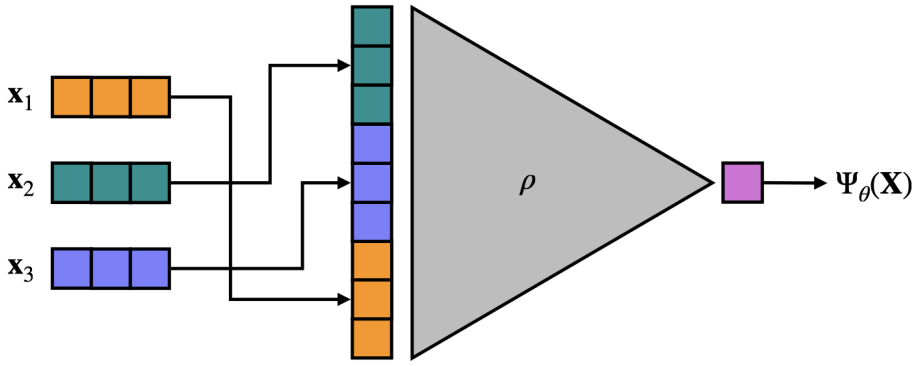

Figure 4.13: An example of a permutation-invariant Jastrow wave function constructed from the single output of a feedforward neural network. To enforce permutation invariance, the input vectors $\boldsymbol{x}_i$ can be sorted according to a chosen rule before concatenating them into a single input vector. The large gray triangle represents a standard feedforward neural network with nine inputs and one output.

as pixels in a 2D image, as flattening the image inherently compromises the spatial structure of the data[2]. Similarly, standard feedforward neural networks encounter challenges when attempting to effectively learn sets that exhibit permutation invariance. Without special treatment, the network would require at least $N!$ times more data points to learn a set of N elements compared to learning on an equivalently sized data set without permutation invariance, in order to effectively capture the inherent redundancies in the data. Even after learning, the network's representation of permutation invariance is only approximate. This becomes problematic when constructing neural-network quantum states, as exact permutation invariance is often required in advance.

Approaches to address this issue can be categorized into two main strategies: data preprocessing and architecture design. The former approach involves sorting the set according to a specified rule, thereby transforming the set into a fixed-dimensional

[2]This challenge leads to the construction of convolutional neural networks (CNNs), a popular choice of neural network that leverages spatial correlations in grid-like data. As CNNs are specifically designed for grid-like data, we will not apply them as neural-network quantum states for continuous-space systems. Consequently, we will not cover them in this chapter, but we will discuss their generalizations, graph neural networks.

sequence. However, this preprocessing step still results in an input vector size that scales linearly with the number of elements in the set, making it challenging to handle large sets. In this section, we focus on the architecture-level approach, where the inherent set structure is ingrained within the neural network architecture. This approach is commonly known as the Deep Set architecture, originally introduced by Zaheer et al. in Ref. [12].

Suppose we wish to learn on sets of N elements $\{\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_N\}$, where each element $\boldsymbol{v}_i$ is a v-dimensional vector. Formally, a function $\boldsymbol{f} : (\mathbb{R}^v)^N \to \mathbb{R}^O$ is a permutation-invariant function of the set $\{\boldsymbol{v}_i\}$ if it satisfies

$$\boldsymbol{f}(\{\boldsymbol{v}_i\}) = \boldsymbol{f}(\hat{P}\{\boldsymbol{v}_i\}), \tag{4.56}$$

for any permutation $\hat{P} \in S_N$. First, we recognize that the function $\boldsymbol{f}$ is permutation-invariant if and only if it can be decomposed as

$$\boldsymbol{f}\big(\{\boldsymbol{v}_i\}\big) = \boldsymbol{\rho}\left(\sum_{i=1}^{N} \boldsymbol{\phi}(\boldsymbol{v}_i)\right), \tag{4.57}$$

for some functions $\boldsymbol{\phi} : \mathbb{R}^v \to \mathbb{R}^L$ and $\boldsymbol{\rho} : \mathbb{R}^L \to \mathbb{R}^O$. The function $\boldsymbol{\phi}$ maps each element of the set to an L-dimensional latent space. Then, the summation destroys the ordering of the set, resulting in an L-dimensional representation of the set. Then the function $\boldsymbol{\rho}$ maps the set representation into an O-dimensional output space.

A Deep Set is realized by replacing each of the functions $\boldsymbol{\phi}$ and $\boldsymbol{\rho}$ with feedforward neural networks. Since feedforward neural networks are universal function approximators, the result of this substitution is a universal approximator for permutation-invariant functions. The summation operation in Eq. (4.57) is the key ingredient in this construction, as it aggregates

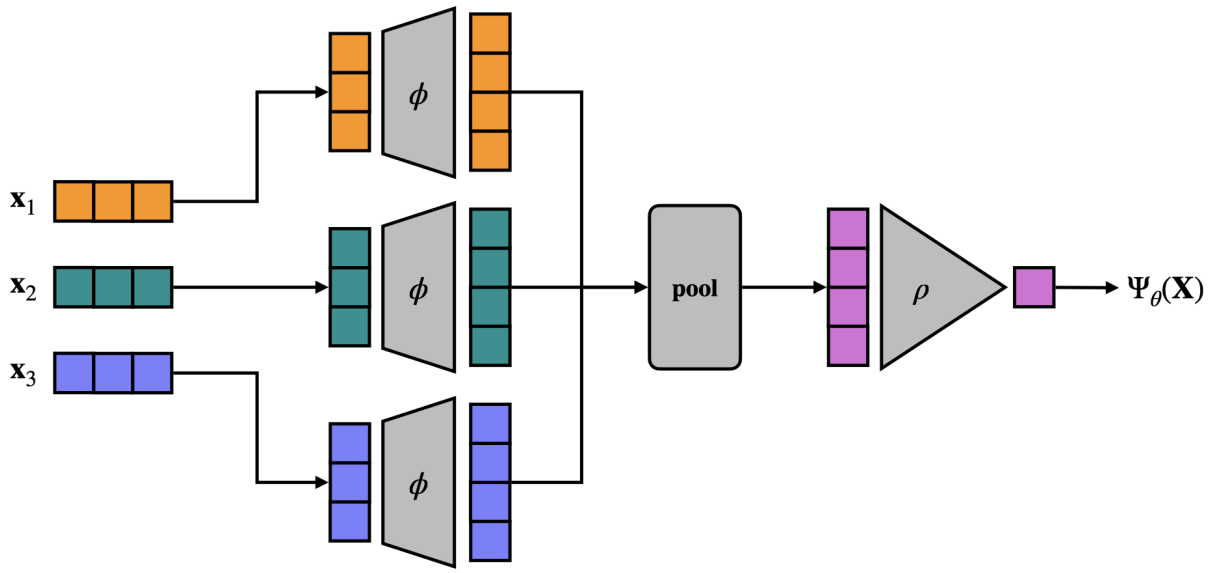

Figure 4.14: A depiction of a permutation-invariant Jastrow wave function constructed from a Deep Set. First, each of the input vectors $\boldsymbol{x}_i$ are passed through identical copies of a standard feedforward neural network ϕ (gray trapezoids). Then the pooling operation (gray rounded box) generates a latent space representation of the set of inputs by destroying the ordering. Finally, the latent space representation is passed through another feedforward neural network ρ (gray triangle) with a single output. The positive-definite Jastrow factor can be obtained by simply exponentiating the single output.

the latent space representations of the elements into a collective set representation. Since other operations can play the same role, we often write our Deep Sets with the more generic form

$$\boldsymbol{f}\big(\{\boldsymbol{v}_i\}\big) = \boldsymbol{\rho}\Big(\boldsymbol{pool}\big(\{\boldsymbol{\phi}(\boldsymbol{v}_i)\}\big)\Big), \tag{4.58}$$

where ***pool*** can be any pooling operation that collects the set of vectors it acts upon, destroys their ordering, and returns a vector of equal size to a single element in the set.

The general idea above can be extended to construct generic permutation-*equivariant* functions as well. The function $\boldsymbol{f} : (\mathbb{R}^v)^N \to (\mathbb{R}^O)^N$ is formally permutation-equivariant if

$$\hat{P}\boldsymbol{f}(\{\boldsymbol{v}_i\}) = \boldsymbol{f}(\hat{P}\{\boldsymbol{v}_i\}) \tag{4.59}$$

holds for all permutations $\hat{P} \in S_N$, with $\boldsymbol{f}(\{\boldsymbol{v}_i\}) \equiv \Big(\boldsymbol{f}_1(\{\boldsymbol{v}_i\}), \boldsymbol{f}_2(\{\boldsymbol{v}_i\}), ..., \boldsymbol{f}_N(\{\boldsymbol{v}_i\})\Big)$. One of the simplest ways to satisfy this condition is to define

$$\boldsymbol{f}_i\big(\{\boldsymbol{v}_j\}\big) = \boldsymbol{\rho}\Big(\boldsymbol{pool}\big(\{\boldsymbol{\phi}(\boldsymbol{v}_j)|j \neq i\}\big)\Big), \tag{4.60}$$

where ρ and ϕ play the same role as before. Another option includes

$$\boldsymbol{f}_i\big(\{\boldsymbol{v}_j\}\big) = \Big(\boldsymbol{v}_i, \boldsymbol{g}\big(\{\boldsymbol{v}_j\}\big)\Big), \tag{4.61}$$

where $\boldsymbol{g}$ is a permutation-invariant function such as a standard Deep Set.

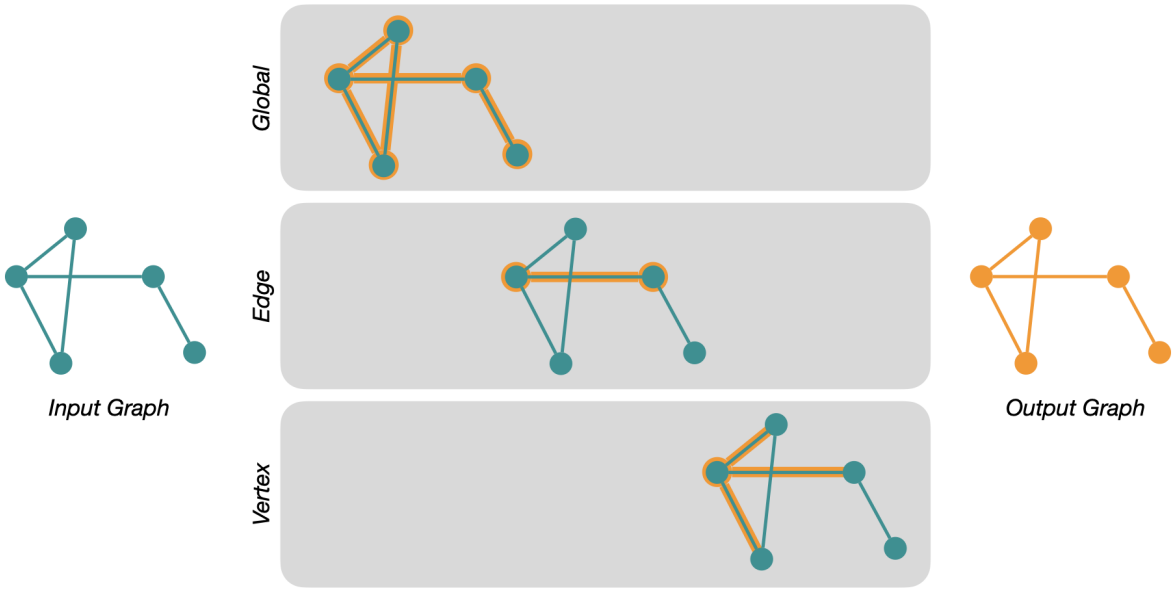

Figure 4.15: One layer of a graph neural network that generates an output graph with the same structure as the original input graph. Three types of transformations can be considered: global (top), edge (middle), and vertex (bottom) transformations. The nodes and edges highlighted in orange show examples of the graph components that contribute to a given transformation.

4.7.4 Graph Neural Networks

Graph neural networks (GNNs) can be seen as a generalization of the popular convolutional neural networks, as they extend the concept of convolutions from grid-like data, such as images, to more general graph-structured data. A graph is a collection of nodes, or vertices, connected by edges. The idea of a graph neural network, specifically the graph-to-graph variant, is to take some input graph and iteratively embed correlations between the edges and nodes to produce a new graph with the same connectivity.

For demonstration purposes, we will construct a generic version of a graph-to-graph GNN acting on three different types of information: vertex features $\boldsymbol{v}_i^{(t)} \in \mathbb{R}^V$, edge features $\boldsymbol{e}_{ij}^{(t)} \in \mathbb{R}^E$, and global features $\boldsymbol{g}^{(t)} \in \mathbb{R}^G$, where t denotes the current iteration of the GNN.

Then global features $\boldsymbol{g}^{(t)}$ can be constructed by taking in all three types of information from the previous graph,

$$\boldsymbol{g}^{(t)} = \boldsymbol{G}_t\Big(\boldsymbol{g}^{(t-1)},\ \boldsymbol{\rho}_t\big(\{\boldsymbol{v}_i^{(t-1)}\}\big),\ \boldsymbol{\phi}_t\big(\{\boldsymbol{e}_{ij}^{(t-1)}\}\big)\Big), \tag{4.62}$$

where $\boldsymbol{G}_t$ is a feedforward neural network with an output dimension of G, and $\boldsymbol{\rho}_t$ and $\boldsymbol{\phi}_t$ are each Deep Sets acting on the set of all previous vertices and edges, respectively. To preserve the original connectivity of the graph, the edge features can be updated as

$$\boldsymbol{e}_{ij}^{(t)} = \boldsymbol{E}_t\Big(\boldsymbol{e}_{ij}^{(t-1)},\ \boldsymbol{v}_i^{(t-1)},\ \boldsymbol{v}_j^{(t-1)},\ \boldsymbol{g}^{(t)}\Big), \tag{4.63}$$

where $\boldsymbol{E}_t$ is another feedforward neural network with output dimension E. Notice that this update includes the global feature $\boldsymbol{g}^{(t)}$ from the current step, rather than the previous step. In addition, the update includes contributions from the two vertices connected by the particular edge. Finally, we can update the vertex features as

$$\boldsymbol{v}_i^{(t)} = \boldsymbol{V}_t\Big(\boldsymbol{v}_i^{(t-1)},\ \boldsymbol{g}^{(t)},\ \boldsymbol{\eta}_t\big(\{\boldsymbol{e}_{ij}^{(t)} \mid j \in \mathcal{N}(i)\}\big)\Big), \tag{4.64}$$

where $\boldsymbol{V}_t$ is a feedforward neural network with output dimension V and $\boldsymbol{\eta}_t$ is a Deep Set acting on the set of the current edges connected to the particular node. In the notation above, $\mathcal{N}(i)$ denotes the set of all nodes in the *neighborhood* of node i.

By updating the global, edge, and vertex features in this way, we can preserve the original structure of the graph and embed highly non-linear correlations between all of the features in only a few iterations of this procedure. In practice, there are many ways of constructing a graph-to-graph GNN, but the overall concept remains the same. We

leverage the permutation invariance of Deep Sets and the universal approximation properties of feedforward neural networks to exchange information between the vertex, edge, and global features. Consequently, neural-networks of this type are commonly called "message-passing" neural networks.

4.7.5 Attention Mechanisms

Attention mechanisms are neural network components originally developed in the field of natural language processing to help decipher the meaning of a sentence based on context. More specifically, they were designed to address the challenge posed by word ambiguity. Humans are able to interpret the meaning of a particular word by leveraging the surrounding words in the sentence. For example, the word "lead" has many different possible definitions, but if it appears alongside words like "metal", "towards", "musical", "dog", or "race", the array of possibilities can be quickly refined.

In the context of quantum many-body problems, attention mechanisms can boost the flexibility of a neural network and help efficiently allocate computational resources to process the most relevant features in the data. To illustrate this concept, let us consider an interaction potential that is only nonzero when a pair of particles have opposite spin. In this scenario, the objective of the attention mechanism could include recognizing that the distance information for a pair of particles is only relevant if their spins are opposite. Of course, there could be other correlations that are non-intuitive for humans to interpret, but the attention mechanism discovers them completely autonomously.

To build an attention mechanism, we begin by organizing our N input sequences $\boldsymbol{x}_i \in \mathbb{R}^d$ column-wise into a matrix $X \in \mathbb{R}^{d \times N}$. Then we define query W_Q, key W_K, and value W_V weight matrices all with dimensions $D \times d$. We apply each of these weight matrices to our

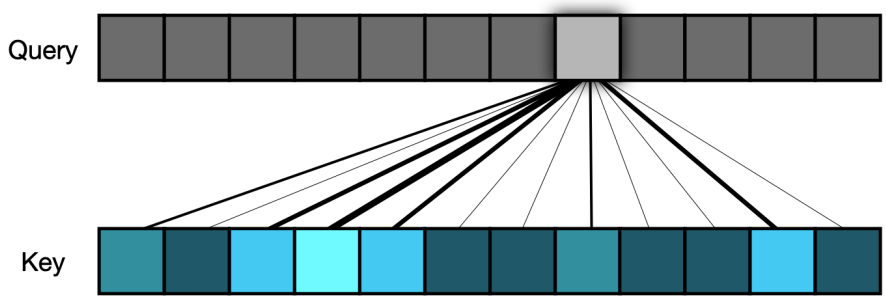

Figure 4.16: A visualization of the attention scores between a single element of a query vector and all the elements of a key vector. Larger attention scores are depicted with darker, bolder lines.

data to obtain corresponding queries, keys, and values

$$Q = W_Q X, \quad K = W_K X, \quad V = W_V X, \tag{4.65}$$

where $Q, K, V \in \mathbb{R}^{D \times N}$. Then the attention scores A, i.e. the normalized overlap between the queries and keys, are commonly computed as

$$A = \textbf{softmax}\left(\frac{QK^T}{\sqrt{D}}\right) \in \mathbb{R}^{D \times D}, \tag{4.66}$$

with **softmax** denoting the row-wise application of the function

$$\textbf{softmax}(\boldsymbol{a})_i = \frac{e^{a_i}}{\sum_{a=1}^{D} e^{a_i}}, \tag{4.67}$$

where $\boldsymbol{a}$ denotes a D-dimensional vector. Scaling the denominator by $\sqrt{D}$ helps prevent

vanishing gradients. Finally, the attention scores are applied to the values as

$$Y = AV \in \mathbb{R}^{D \times N}, \tag{4.68}$$

where Y is the output of the attention mechanism.

The attention mechanism above is specifically known as self-attention, since the queries, keys, and values are all derived from the input data X. Other variations of attention include cross-attention, in which the queries are either constructed from an entirely different external data source X' or treated as a completely trainable matrix. In addition, multiple attention-mechanisms can be implemented in parallel, leading to multi-headed attention.

5 Implementation

Throughout the rest of this dissertation, we will reference two different implementations of neural-network quantum states for variational Monte Carlo, one in C++ and another in Python. As their objectives and approaches significantly differ, we will organize this chapter into two sections, each dedicated to its respective implementation.

5.1 `NeuralAnsatz`: C++ Software for Localized Systems

The first implementation, named `NeuralAnsatz`, is an object-oriented C++ code that leverages distributed-memory programming with `Open MPI` for efficient utilization of multiple CPUs. The code is built from scratch using `Eigen`, a lightweight, header-only linear algebra library, eliminating the need for separate linking during compilation. The primary features and goals for this implementation include: modularity, reproducibility, low memory costs, and exact gradient calculations. The source code can be found in this Github repository or at this url: `https://github.com/kim-jane/NeuralAnsatz`.

5.1.1 Trainer

`NeuralAnsatz` consists of several components, called `Objects`, that are passed to a single universal `Trainer`. The `Object` class is an abstract base class that simply contains relevant names, abbreviations, and simulation information so that the `Trainer` can easily handle files and compile reports. This way, it is straightforward to refer back to a past run, reset the simulation parameters, and reproduce the results. In order to facilitate interdependence between the `Objects` beyond the scope of the `Trainer`, each `Object` must be instantiated as a `std::shared_ptr`. This choice enables other `Objects` to assume ownership when necessary

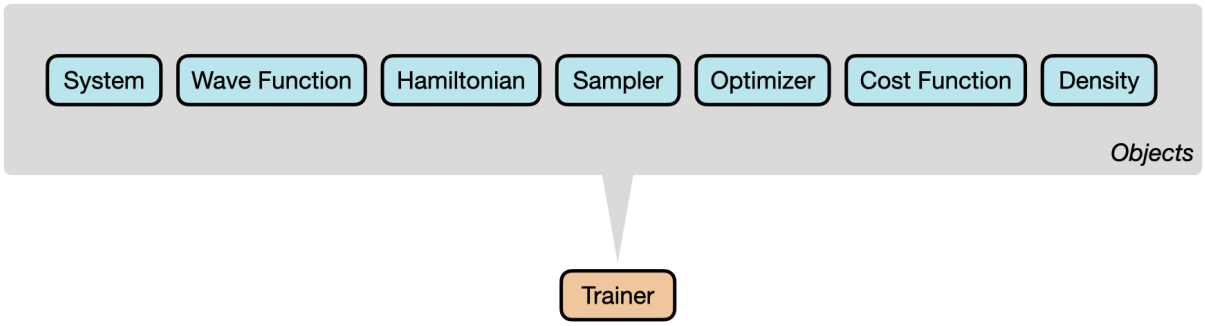

Figure 5.1: The highest-level structure of the `NeuralAnsatz` code, an object-oriented C++ software for neural-network quantum states.

and ensures safe deallocation of the pointers when the `Objects` go out of scope.

```
// main.cpp
std::shared_ptr<Object> pObject = std::make_shared<Object>(...);
```

It is essential to start every driver file (e.g., main.cpp) by initializing the `MPI_COMM_WORLD` communicator and setting a unique seed on each parallel process, prior to instantiating any Objects. These lines only need to be placed at the start of the file, in this particular order, and there is no explicit need to finalize MPI.

```
// main.cpp
mpi::initialize();
rng::initialize();
```

Then after constructing all the desired `Objects`, the trainer can be created as follows:

```
// main.cpp
Trainer T(pSystem, pWaveFunc, pSampler, pOptimizer, pCostFunc, pDensity,
          filename);
T.train();
```

Notice that the `Trainer` accepts a pointer to a `CostFunction` rather than a `Hamiltonian`. This design choice is deliberate because while variational Monte Carlo involves minimizing the energy as a reinforcement learning problem, there are situations where supervised learning can be valuable, in pretraining the `WaveFunction` for example. Our `Trainer` is versatile enough to handle both supervised and reinforcement learning problems.

One of the main jobs of the `Trainer` is to handle all opening, closing, and writing of files. For a given simulation, the `Trainer` sends four different outputs to the `data/` folder:

- `data/history/filename.txt`: This file contains the entire history of the energy, the error in the energy, the norm of the gradient, the norm of the parameter vector, the average positions, the variance of the positions, the acceptance rates for the positions and spins, and the time per iteration. If more than 10000 training epochs are requested, the `Trainer` will skip printing some iterations to prevent the file from becoming too large.

- `data/log/filename.txt`: This file contains snapshots of the contents in `data/history/filename.txt` for a broad, easy-to-interpret overview of the training. Any errors or additional print statements will be sent to this file, including the total execution time of the program.

- `data/params/filename.txt`: This file contains snapshots of the variational parameters throughout training. One can use this file to provide pretrained

parameters for a wave function or a component of a wave function. If an error occurs, one can revert back to a previous stage in the training by deleting the unwanted variational parameters.

- `data/density/filename.txt`: This file contains snapshots of the density during training, provided by the `Density` object. Monitoring the density is helpful for debugging problems during training and, of course, to visualize the training of the wave function.

All files will contain a summary of the simulation details at the top of the file. Because the wave function tends to evolve the most near the beginning of the training process, the iterations at which snapshots are taken follow the pattern 0, 10, 20, ..., 100, 200, ..., 1000, 2000, ..., until the specified number of total training epochs is reached.

5.1.2 Systems

The `System` abstract class is very simple; it contains only the most basic simulation details that most other `Objects` depend on, such as the number of particles, the number of spatial dimensions, the mass, the value of $\hbar$, etc. The present version of `NeuralAnsatz` offers support for localized, continuous-space systems of spin-0 bosons and spin-1/2 fermions in arbitrary spatial dimensions. Although support for nucleons is present in the source code, it has not undergone testing with a Hamiltonian that includes both spin and isospin dependence. Ongoing development is underway to incorporate support for periodic systems, elements of which will be discussed herein.

In order for all `Objects` to be compatible with both bosons and fermions, we store all

spatial and spin degrees of freedom in a single matrix,

$$X \equiv \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \vdots \\ \boldsymbol{x}_N^T \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_1^T & \boldsymbol{s}_1^T \\ \boldsymbol{r}_2^T & \boldsymbol{s}_2^T \\ \vdots & \vdots \\ \boldsymbol{r}_N^T & \boldsymbol{s}_N^T \end{bmatrix} \in \mathbb{R}^{N \times (d+s)} \tag{5.1}$$

where the d left-most columns represent the positions, the s right-most columns represent the spins (and isospins), and the rows correspond to the particles. This construction is simply the matrix form of the notation for $\boldsymbol{X}$ we introduced in Sec. 2.1. The bolded notation will be used to denote vectors, while the unbolded notation will be used for matrices, but they should be understood as interchangeable when used as inputs to functions, e.g. $f(X) = f(\boldsymbol{X})$. `Objects` that are designed specifically for `Bosons`, `Fermions`, or `Nucleons` will be specified in the constructor. Otherwise, all `Objects` are functions of any `System`. Due to the heirarchy of this group, functions that operate on a `System` will be compatible with any derived class, and functions that operate on `Fermions` will be compatible with `Nucleons`. Derived classes can override functions from their base classes by providing their own implementation of the function with the same name and signature.

The `Bosons` class is the simplest derived `System` class, as it contains no special attributes that are specific to bosons, other than their name. The `Fermions` class, on the other hand, determines how many distinct spatial orbitals are needed in a Slater Determinant, given a specific number of spin-up/spin-down particles. It also stores a convenient spin configuration with the desired polarization for use during the Monte Carlo sampling of spins, as discussed in Sec. 3.1.1, and projects configurations onto the pre-determined spin components of the single-particle spin-orbitals.

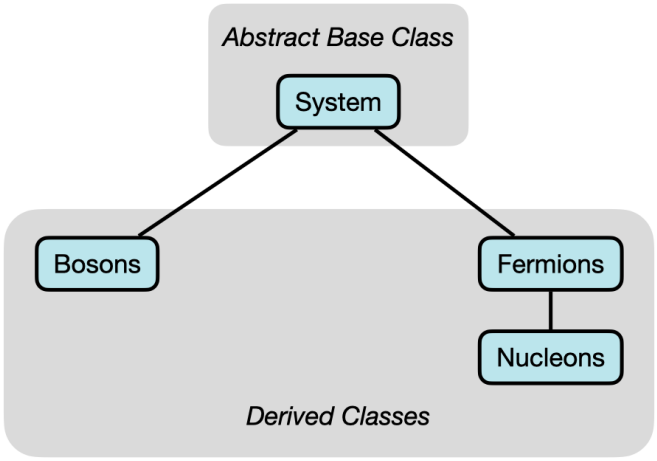

Figure 5.2: A specific example of the hierarchical structure provided by abstract base classes and derived classes. The classes that appear lower on this graph inherit properties from the classes above it.

```
// fermions.cpp
mat Fermions::get_spin_projection(const mat& X)
{
    s_ = X.col(n_dims_);
    for(int i = 0; i < n_parts_; ++i)
    {
        S_.row(i) = 0.5 * (ones_ + spin_orbitals_(i,0) * s_ );
    }
    return S_;
}
```

This code snippet highlights a few additional points about our strategy. As we aim to minimize our memory footprint, we declare known array sizes in the constructor of the class, rather than dynamically allocating them on the fly. The names of all member variables are followed by a trailing underscore. In the above example, `S_` is a square matrix of dimension N that holds the spin projections, and `s_` is a vector that holds the spins from the larger matrix `X`. The variable `spin_orbitals_` is an $N \times s$ matrix that stores the spin and isospin quantum numbers corresponding to each single-particle spin-orbital. The function above is overridden in the `Nucleons` class to account for the isospin projections.

5.1.3 Wave Functions

Classes derived from the abstract base class `WaveFunction` must provide definitions for the following capabilities:

- Compute the logarithm of the trial wave function $\log \Psi_{\boldsymbol{\theta}}(\boldsymbol{X})$, or more generally, $\log |\Psi_{\boldsymbol{\theta}}(\boldsymbol{X})|$ and $\operatorname{sgn}(\Psi_{\boldsymbol{\theta}}(\boldsymbol{X}))$.

- Compute the local gradient of the trial wave function with respect to the coordinates of all particles $\nabla \log \Psi_{\boldsymbol{\theta}}(\boldsymbol{X}) \equiv \Big(\nabla_1 \log \Psi_{\boldsymbol{\theta}}(\boldsymbol{X}), \nabla_2 \log \Psi_{\boldsymbol{\theta}}(\boldsymbol{X}), \cdots, \nabla_N \log \Psi_{\boldsymbol{\theta}}(\boldsymbol{X})\Big)$.

- Compute the sum of the local Laplacians with respect to the coordinates of each particles $\sum_{i=1}^{N} \frac{\nabla_i^2 \Psi_{\boldsymbol{\theta}}(\boldsymbol{X})}{\Psi_{\boldsymbol{\theta}}(\boldsymbol{X})}$. This will allow the `Hamiltonian` class to compute the local energy $E_{\boldsymbol{\theta}}(\boldsymbol{X}) = \frac{\hat{H}\Psi_{\boldsymbol{\theta}}(\boldsymbol{X})}{\Psi_{\boldsymbol{\theta}}(\boldsymbol{X})}$, as defined in Eq. (3.47).

- Compute the local gradient of the trial wave function with respect to the trainable parameters $\boldsymbol{O}_{\boldsymbol{\theta}}(\boldsymbol{X}) = \nabla_{\boldsymbol{\theta}} \log \Psi_{\boldsymbol{\theta}}(\boldsymbol{X})$, as defined in Eq. (3.48).

- Flatten and unflatten the trainable parameters $\boldsymbol{\theta}$.

As implied by the requirements listed above, computations will be written in terms of $\log \Psi_{\boldsymbol{\theta}}(\boldsymbol{X})$ instead of $\Psi_{\boldsymbol{\theta}}(\boldsymbol{X})$, which helps to mitigate the risk of numerical overflow or underflow. In addition, the abstract base class allows snapshots of the parameters to be written to file during training, and conversely, allows pretrained parameters to be loaded from file. All computations are written exactly in terms of the variational parameters $\boldsymbol{\theta}$ and the input data $\boldsymbol{X}$, or more specifically, its matrix form X. We will first discuss the various options for inputs to the neural networks, then the implementations of specific neural-network quantum states.

5.1.3.1 Inputs

The purpose of an `Input` object is to format the raw data X into a single vector $\boldsymbol{v}$ which will be passed into an artificial neural network. The matrix $X \in \mathbb{R}^{N\times(d+s)}$ contains all the spatial and spin degrees of freedom of all N particles, where d and s are the numbers of spatial and spin degrees of freedom per particle, respectively. For instance, nucleons have $d = 3$ and $s = 2$, with $\boldsymbol{x}_i = (\boldsymbol{r}_i, s_i, t_i)$ containing the coordinates, spin projections on the z-axis, and the isospin projections on the z-axis. Separating the `Input` object from the `WaveFunction` object allows for increased modularity, as we can interchange the different types of inputs to the network without affecting the source code of the neural-network quantum states.

One-Body. If the one-body features are the inputs to the neural network, then the input vector $\boldsymbol{v}$ can be divided into N equal parts

$$\boldsymbol{v} = (\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_N) \in \mathbb{R}^V \tag{5.2}$$

with $\boldsymbol{v}_i \in \mathbb{R}^v$, $Nv \equiv V$. To compute the local kinetic energy, we will need the expression for the gradient of $\boldsymbol{v}$ with respect to all of the particles $\boldsymbol{R} = (\boldsymbol{r}_1, \boldsymbol{r}_2, ..., \boldsymbol{r}_N) \in \mathbb{R}^{Nd}$, which can be written in block matrix form as

$$\nabla \boldsymbol{v} = \begin{bmatrix} \nabla_1 \boldsymbol{v}_1 & \nabla_2 \boldsymbol{v}_1 & \cdots & \nabla_N \boldsymbol{v}_1 \\ \nabla_1 \boldsymbol{v}_2 & \nabla_2 \boldsymbol{v}_2 & \cdots & \nabla_N \boldsymbol{v}_2 \\ \vdots & \vdots & \ddots & \vdots \\ \nabla_1 \boldsymbol{v}_N & \nabla_2 \boldsymbol{v}_N & \cdots & \nabla_N \boldsymbol{v}_N \end{bmatrix} \in \mathbb{R}^{Nv\times Nd}, \tag{5.3}$$

with the blocks

$$\nabla_j \boldsymbol{v}_i = \frac{\partial \boldsymbol{v}_i}{\partial \boldsymbol{r}_j} \in \mathbb{R}^{v \times d}. \tag{5.4}$$

The Laplacian of each visible node with respect to each particle can be organized into a similar block matrix

$$\nabla^2 \boldsymbol{v} = \begin{bmatrix} \nabla_1^2 \boldsymbol{v}_1 & \nabla_2^2 \boldsymbol{v}_1 & \cdots & \nabla_N^2 \boldsymbol{v}_1 \\ \nabla_1^2 \boldsymbol{v}_2 & \nabla_2^2 \boldsymbol{v}_2 & \cdots & \nabla_N^2 \boldsymbol{v}_2 \\ \vdots & \vdots & \ddots & \vdots \\ \nabla_1^2 \boldsymbol{v}_N & \nabla_2^2 \boldsymbol{v}_N & \cdots & \nabla_N^2 \boldsymbol{v}_N \end{bmatrix} \in \mathbb{R}^{Nv \times N}, \tag{5.5}$$

with

$$\nabla_j^2 \boldsymbol{v}_i = \frac{\partial}{\partial \boldsymbol{r}_j} \cdot \frac{\partial \boldsymbol{v}_i}{\partial \boldsymbol{r}_j} \in \mathbb{R}^{v \times 1}. \tag{5.6}$$

Writing the Laplacian in this way makes it possible to use the chain rule efficiently later on.

Unsorted, Nonperiodic. In the simplest case, we can take $\boldsymbol{v}$ to be the row-wise, flattened form of X. This type of input vector is suitable for localized systems and Deep Sets, a neural network that enforces permutation symmetry at the architecture level. Then we simply have $v = d + s$ and

$$\boldsymbol{v}_i = \boldsymbol{x}_i, \tag{5.7}$$

$$\nabla_j \boldsymbol{v}_i = \delta_{i,j} \begin{bmatrix} \boldsymbol{I}_{d \times d} \\ \\ \boldsymbol{0}_{s \times d} \end{bmatrix}, \tag{5.8}$$

$$\nabla_j^2 \boldsymbol{v}_i = \boldsymbol{0}. \tag{5.9}$$

Sorted, Nonperiodic. Another way to enforce permutation symmetry is to sort the particles according to some scalar value. For example, one-dimensional particles can be sorted by their position, while two- and three-dimensional particles can be sorted by their Euclidean distance from the origin $r_i = \|\boldsymbol{r}_i\|$. Since permutation symmetry is enforced at the data-processing level, this type of input vector is suitable for any neural network. Let $Z_N = \{1, 2, ..., N\}$ denote the set of all integers from 1 and N, and $m : Z_N \longrightarrow Z_N$ denote the bijective map that sorts the particles according to the chosen rule. Then the result is very similar to before, with $v = d + s$ and

$$\boldsymbol{v}_i = \boldsymbol{x}_{m(i)}, \tag{5.10}$$

$$\nabla_j \boldsymbol{v}_i = \delta_{m(i),j} \begin{bmatrix} \boldsymbol{I}_{d\times d} \\ \boldsymbol{0}_{s\times d} \end{bmatrix}, \tag{5.11}$$

$$\nabla_j^2 \boldsymbol{v}_i = \boldsymbol{0}. \tag{5.12}$$

Unsorted, Periodic. For periodic systems, it is crucial to transform X so that $\boldsymbol{v}$ is explicitly periodic, otherwise the neural networks can learn to break the symmetry, leading to erroneous results. To enforce L-periodicity, we transform the coordinates as

$$\boldsymbol{r}_i \longmapsto \tilde{\boldsymbol{r}}_i = \left(\cos\left(\frac{2\pi \boldsymbol{r}_i}{L}\right),\ \sin\left(\frac{2\pi \boldsymbol{r}_i}{L}\right) \right) \in \mathbb{R}^{2d}, \tag{5.13}$$

such that the components of the input vector are

$$\boldsymbol{v}_i = (\tilde{\boldsymbol{r}}_{\boldsymbol{i}}, \boldsymbol{s}_i) \in \mathbb{R}^{2d+s}, \tag{5.14}$$

implying $v = 2d + s$. Then the derivatives are

$$\nabla_j \boldsymbol{v}_i = \frac{2\pi}{L}\delta_{i,j} \begin{bmatrix} -\text{diag}\left[\sin\left(\frac{2\pi \boldsymbol{r}_i}{L}\right)\right] \\ \text{diag}\left[\cos\left(\frac{2\pi \boldsymbol{r}_i}{L}\right)\right] \\ \mathbf{0}_{s\times d} \end{bmatrix}, \tag{5.15}$$

$$\nabla_j^2 \boldsymbol{v}_i = -\left(\frac{2\pi}{L}\right)^2 \delta_{i,j} \begin{bmatrix} \cos\left(\frac{2\pi \boldsymbol{r}_i}{L}\right) \\ \sin\left(\frac{2\pi \boldsymbol{r}_i}{L}\right) \\ \mathbf{0} \end{bmatrix} = -\left(\frac{2\pi}{L}\right)^2 \delta_{i,j} \begin{bmatrix} \tilde{\boldsymbol{r}}_i \\ \mathbf{0} \end{bmatrix}, \tag{5.16}$$

where $\text{diag}\,[\boldsymbol{r}]$ refers to a diagonal matrix with $\boldsymbol{r}$ on the diagonal, and the trigonometric functions are applied element-wise.

Sorted, Periodic. While the inputs themselves must be periodic in this case, the sorting map does not necessarily have to sort according to periodic positions or distances. We therefore use the same sorting map m as the nonperiodic case. The resulting transformation is similar to the previous case, except with the indicies permuted

$$\boldsymbol{v}_i = \left(\tilde{\boldsymbol{r}}_{m(i)}, \boldsymbol{s}_{m(i)}\right) \in \mathbb{R}^{2d+s}, \tag{5.17}$$

$$\nabla_j \boldsymbol{v}_i = \frac{2\pi}{L}\delta_{m(i),j} \begin{bmatrix} -\text{diag}\left[\sin\left(\frac{2\pi \boldsymbol{r}_{m(i)}}{L}\right)\right] \\ \text{diag}\left[\cos\left(\frac{2\pi \boldsymbol{r}_{m(i)}}{L}\right)\right] \\ \mathbf{0}_{s\times d} \end{bmatrix}, \tag{5.18}$$

$$\nabla_j^2 \boldsymbol{v}_i = -\left(\frac{2\pi}{L}\right)^2 \delta_{m(i),j} \begin{bmatrix} \cos\left(\frac{2\pi \boldsymbol{r}_{m(i)}}{L}\right) \\ \sin\left(\frac{2\pi \boldsymbol{r}_{m(i)}}{L}\right) \\ \mathbf{0} \end{bmatrix} = -\left(\frac{2\pi}{L}\right)^2 \delta_{m(i),j} \begin{bmatrix} \tilde{\boldsymbol{r}}_{m(i)} \\ \mathbf{0} \end{bmatrix}, \tag{5.19}$$

with $v = 2d + s$.

Two-Body. If the inputs to the neural network are two-body features, then $\boldsymbol{v}$ can be divided into $P = N(N-1)/2$ equal parts

$$\boldsymbol{v} = (\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_P) \in \mathbb{R}^V, \tag{5.20}$$

where $\boldsymbol{v} \in \mathbb{R}^v$ and $Pv \equiv V$. The gradient of $\boldsymbol{v}$ with respect to all of the particles $\boldsymbol{r} = (\boldsymbol{r}_1, \boldsymbol{r}_2, ..., \boldsymbol{r}_N) \in \mathbb{R}^{ND}$ can be written in block matrix form as

$$\nabla \boldsymbol{v} = \begin{bmatrix} \nabla_1 \boldsymbol{v}_1 & \nabla_2 \boldsymbol{v}_1 & \cdots & \nabla_N \boldsymbol{v}_1 \\ \nabla_1 \boldsymbol{v}_2 & \nabla_2 \boldsymbol{v}_2 & \cdots & \nabla_N \boldsymbol{v}_2 \\ \vdots & \vdots & \ddots & \vdots \\ \nabla_1 \boldsymbol{v}_P & \nabla_2 \boldsymbol{v}_P & \cdots & \nabla_N \boldsymbol{v}_P \end{bmatrix} \in \mathbb{R}^{Pv \times Nd}, \tag{5.21}$$

with the blocks

$$\nabla_k \boldsymbol{v}_p = \frac{\partial \boldsymbol{v}_p}{\partial \boldsymbol{r}_k} \in \mathbb{R}^{v \times d}. \tag{5.22}$$

The Laplacian of each visible node with respect to each particle can also be organized into a block matrix

$$\nabla^2 \boldsymbol{v} = \begin{bmatrix} \nabla_1^2 \boldsymbol{v}_1 & \nabla_2^2 \boldsymbol{v}_1 & \cdots & \nabla_N^2 \boldsymbol{v}_1 \\ \nabla_1^2 \boldsymbol{v}_2 & \nabla_2^2 \boldsymbol{v}_2 & \cdots & \nabla_N^2 \boldsymbol{v}_2 \\ \vdots & \vdots & \ddots & \vdots \\ \nabla_1^2 \boldsymbol{v}_P & \nabla_2^2 \boldsymbol{v}_P & \cdots & \nabla_N^2 \boldsymbol{v}_P \end{bmatrix} \in \mathbb{R}^{Pv \times N}, \tag{5.23}$$

with

$$\nabla_k^2 \boldsymbol{v}_p = \frac{\partial}{\partial \boldsymbol{r}_k} \cdot \frac{\partial \boldsymbol{v}_p}{\partial \boldsymbol{r}_k} \in \mathbb{R}^{v \times 1}. \tag{5.24}$$

In the following, we will assume that the two-body features are symmetric, meaning the feature for the pair i, j is the same as the feature for pair j, i. We store the mapping between a particular pair (i, j) and their pair index p as $i(p)$ and $j(p)$, always assuming $i < j$. To simplify notation, we will use

$$\boldsymbol{r}_p = \boldsymbol{r}_{i(p)} - \boldsymbol{r}_{j(p)} \tag{5.25}$$

to denote the separation vector between particles $i(p)$ and $j(p)$ corresponding to the pair p, and

$$r_p = \|\boldsymbol{r}_p\| \tag{5.26}$$

to denote the distance between the pair.

Unsorted, Nonperiodic. The components of the input vector are

$$\boldsymbol{v}_p = (r_p,\ \boldsymbol{s}_{i(p)} * \boldsymbol{s}_{j(p)}) \in \mathbb{R}^{1+s}, \tag{5.27}$$

where $*$ denotes an element-wise product of the spin degrees of freedom and $v = 1 + s$. The corresponding derivatives are

$$\nabla_k \boldsymbol{v}_p = \left(\delta_{i(p),k} - \delta_{j(p),k}\right) \frac{1}{r_p} \begin{bmatrix} \boldsymbol{r}_p^T \\ \mathbf{0}_{Q \times D} \end{bmatrix}, \tag{5.28}$$

$$\nabla_k^2 \boldsymbol{v}_p = \left(\delta_{i(p),k} + \delta_{j(p),k}\right) \frac{2}{r_p} \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix}. \tag{5.29}$$

Sorted, Nonperiodic. Let $m : Z_P \longrightarrow Z_P$ be the mapping that sorts the pairs according to the distance between them r_p. Then,

$$\boldsymbol{v}_p = (r_{m(p)},\ \boldsymbol{s}_{i(m(p))} * \boldsymbol{s}_{j(m(p))}), \tag{5.30}$$

$$\nabla_k \boldsymbol{v}_p = \left(\delta_{i(m(p)),k} - \delta_{j(m(p)),k}\right) \frac{1}{r_{m(p)}} \begin{bmatrix} \boldsymbol{r}_{m(p)}^T \\ \mathbf{0}_{Q\times D} \end{bmatrix}, \tag{5.31}$$

$$\nabla_k^2 \boldsymbol{v}_p = \left(\delta_{i(m(p)),k} + \delta_{j(m(p)),k}\right) \frac{2}{r_{m(p)}} \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix}. \tag{5.32}$$

Unsorted, Periodic. The distances between pairs are transformed to their periodic counterparts,

$$r_p \longmapsto \tilde{r}_p = \left\| \sin\left(\frac{\pi \boldsymbol{r}_p}{L}\right) \right\|, \tag{5.33}$$

making the components of the input vector become

$$\boldsymbol{v}_p = (\tilde{r}_p,\ \boldsymbol{s}_{i(p)} * \boldsymbol{s}_{j(p)}) \in \mathbb{R}^{1+Q}, \tag{5.34}$$

where $\boldsymbol{s}_i \in \mathbb{R}^Q$ is a vector containing the spin degrees of freedom of the i-th particle and $*$ denotes an element-wise product.

$$\nabla_k \boldsymbol{v}_p = \left(\delta_{i(p),k} - \delta_{j(p),k}\right) \frac{\pi}{2L\tilde{r}_p} \begin{bmatrix} \sin\left(\frac{2\pi \boldsymbol{r}_p}{L}\right)^T \\ \mathbf{0}_{Q\times D} \end{bmatrix} \in \mathbb{R}^{(1+Q)\times D} \tag{5.35}$$

$$\nabla_k^2 \boldsymbol{v}_p = \left(\delta_{i(p),k} + \delta_{j(p),k}\right) \frac{2}{\tilde{\imath}_p} \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{1+Q} \tag{5.36}$$

Sorted, Periodic. Let $m : Z_P \longrightarrow Z_P$ be the mapping that sorts the pairs according to the periodic distance between them $\tilde{\imath}_k$. Then,

$$\boldsymbol{v}_p = (\tilde{\imath}_{m(p)},\ \boldsymbol{s}_{i(m(p))} * \boldsymbol{s}_{j(m(p))}) \in \mathbb{R}^{1+Q}, \tag{5.37}$$

$$\nabla_k \boldsymbol{v}_p = \left(\delta_{i(m(p)),k} - \delta_{j(m(p)),k}\right) \frac{\pi}{2L\tilde{\imath}_{m(p)}} \begin{bmatrix} \sin\left(\frac{2\pi \boldsymbol{\imath}_{m(p)}}{L}\right)^T \\ \mathbf{0}_{Q\times D} \end{bmatrix} \in \mathbb{R}^{(1+Q)\times D} \tag{5.38}$$

$$\nabla_k^2 \boldsymbol{v}_p = \left(\delta_{i(m(p)),k} + \delta_{j(m(p)),k}\right) \frac{2}{\tilde{\imath}_{m(p)}} \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{1+Q} \tag{5.39}$$

To compute $\boldsymbol{v}$, $\nabla \boldsymbol{v}$, and $\nabla^2 \boldsymbol{v}$ in terms of the inputs X inside a particular `WaveFunction` class, we call the `format` function using a pointer to the `Input` object. There are three versions of the same `format` function since $\boldsymbol{v}$, $\nabla \boldsymbol{v}$, and $\nabla^2 \boldsymbol{v}$ are not necessary for every computation. All derived classes, e.g. `OneBodyInputs` and `TwoBodyInputs`, must provide definitions for these functions.

```
// input.hpp
virtual void format(const mat& X, vec& v) = 0;
virtual void format(const mat& X, vec& v, mat& dv) = 0;
virtual void format(const mat& X, vec& v, mat& dv, mat& d2v) = 0;
```

5.1.3.2 Continuous Restricted Boltzmann Machines

The various versions of multivariate Gaussian-Binary and Gaussian-Uniform restricted Boltzmann machines are implemented under a single class called `ContinuousRBM`. In Sec. 4.7.1, we defined a new function called $f(x)$ that allowed all the log-likelihoods and their gradients with respect to the parameters to have the same form. Since we already need to define activation functions to implement feedforward neural networks, we implement $f(x)$ as activation functions as well, in order to streamline the `ContinuousRBM` class. More information on the implementation of activation functions will be provided in the next section.

We will use `ContinuousRBM` as a symmetric, positive-definite Jastrow wave function for bosons, or as a Jastrow correlator for fermions, combined with an antisymmetric Slater determinant. Recall the form of the log-likelihood for a continuous RBM written in terms of the trainable parameters $\boldsymbol{\theta} = (\boldsymbol{a}, \mathrm{triu}(S), \boldsymbol{b}, W)$,

$$\log \mathcal{P}(\boldsymbol{v}) = -\frac{1}{2}(\boldsymbol{v}-\boldsymbol{a})^T \exp(S)(\boldsymbol{v}-\boldsymbol{a}) + \sum_{j=1}^{H} f(z_j(\boldsymbol{v})) + C, \tag{5.40}$$

$$\boldsymbol{z}(\boldsymbol{v}) = \boldsymbol{b} + W^T \exp(S)\boldsymbol{v}. \tag{5.41}$$

where C is a constant. Since $\mathcal{P}(\boldsymbol{v})$ is the marginal probability distribution of the visible nodes $\boldsymbol{v}$, it is natural to define our trial wave function as

$$\log \Psi_{\boldsymbol{\theta}}(\boldsymbol{X}) \equiv \log \mathcal{P}(\boldsymbol{v}(\boldsymbol{X})). \tag{5.42}$$

In fact, we could make the relationship between the probability amplitude $\Psi_{\boldsymbol{\theta}}(\boldsymbol{X})$ and the

probability distribution more exact by writing $\log|\Psi_{\boldsymbol{\theta}}(\boldsymbol{X})| \equiv \frac{1}{2}\log\mathcal{P}(\boldsymbol{v}(\boldsymbol{X}))$, but due to the presence of all the trainable parameters, the two forms are practically equivalent.

Computing the local gradient with respect to $\boldsymbol{R} = (\boldsymbol{r}_1, ..., \boldsymbol{r}_N)$ becomes straightforward because of the way the `Input` object organizes $\nabla\boldsymbol{v}(\boldsymbol{X})$ into blocks. Namely,

$$\nabla \log \Psi_{\boldsymbol{\theta}}(\boldsymbol{X}) = \Big[-(\boldsymbol{v}-\boldsymbol{a})^T + \boldsymbol{f}'(\boldsymbol{z}(\boldsymbol{v}(\boldsymbol{X})))^T W^T \Big] \exp(S) \nabla \boldsymbol{v}(\boldsymbol{X}). \tag{5.43}$$

To compute the sum of local Laplacians, first notice that

$$\frac{1}{\Psi_{\boldsymbol{\theta}}(\boldsymbol{X})} \nabla_i^2 \Psi_{\boldsymbol{\theta}}(\boldsymbol{X}) = \nabla_i^2 \log \Psi_{\boldsymbol{\theta}}(\boldsymbol{X}) + \big(\nabla_i \log \Psi_{\boldsymbol{\theta}}(\boldsymbol{X})\big)^2. \tag{5.44}$$

If we again, organize our Laplacians into blocks, we can simplify the computation to be in terms of matrix and array operations only, eliminating the need for any for-loops.

```
// continuous_rbm.cpp
long double ContinuousRBM::get_local_laplacian(const mat& X)
{
    pInput_->format(X, v_, dv_, d2v_);

    K_ = exp(S_);
    va_ = v_ - a_;
    dA_ = - va_.transpose() * K_ * dv_;
    d2A_ = - ( va_.transpose() * K_ * d2v_ ).sum()
         + - ( dv_.transpose() * K_ * dv_ ).trace();

    z_ = b_ + W_.transpose() * K_ * v_;
    dz_ = W_.transpose() * K_ * dv_;
    d2z_ = W_.transpose() * K_ * d2v_;

    df_ = pFunc_->get_df(z_);
    d2f_ = pFunc_->get_d2f(z_);

    dB_ = df_.transpose() * dz_;
    d2B_ = df_.dot( d2z_.rowwise().sum() )
         + d2f_.dot( dz_.rowwise().squaredNorm() );

    return d2A_ + d2B_ + (dA_ + dB_).squaredNorm();
}
```

In the above code snippet, `pInput_` is the member pointer to the `Input` object and `pFunc_` is the member pointer to the (activation) function $f(x)$. Finally, because we defined our trial wave function as the marginal probability distribution of the visible nodes, the derivatives of the log-likelihood in Eq. (4.37) coincide with the derivatives of $\log \Psi_{\boldsymbol{\theta}}(\boldsymbol{X})$.

In general, the variational parameters of any restricted Boltzmann machine can be initialized to small random values near zero. However, because RBMs are highly interpretable models, we can provide much better guesses for the initial parameters based on our physical intuition. As a reminder, the variational parameters of an mGB-RBM and an mGU-RBM are $\boldsymbol{\theta} = (\boldsymbol{a}, \mathrm{triu}(S), \boldsymbol{b}, W)$, where $\boldsymbol{a}$ are the biases of the visible nodes, S is

used to reparameterize the inverse covariance matrix

$$\Sigma_{ii}^{-1} = \exp(S_{ii}), \text{ and } \Sigma_{ij}^{-1} = S_{ij} \text{ for } j \neq i,$$

$\boldsymbol{b}$ are the biases of the hidden nodes, and W is a weight matrix connecting the visible and hidden nodes. If $\boldsymbol{b}$ and W are both identically zero, the marginal probability distribution of both RBMs become a multivariate Gaussian. Thus setting $\boldsymbol{b}$ and W to small random values near zero represents small deviations from the multivariate Gaussian. For small $\boldsymbol{b}$ and W, the individual peaks of the Gaussian nearly coincide with the biases $\boldsymbol{a}$. Therefore, we can set the initial values of $\boldsymbol{a}$ based on our guesses for the most likely positions of the particles.

The initialization of the matrix S is the most important, as choosing convenient values can prevent the multivariate Gaussian from becoming multi-modal early on during training, leading to the collapse of the sampling process. In addition, the above reparametrization guarantees the invertibility of Σ^{-1} but it does not guarantee that the diagonals of Σ are positive. Since negative variances are unphysical, we bias our network to provide positive variances at the beginning of training. Then the RBM naturally learns to maintain positive variances on its own. This is accomplished by choosing the diagonal values of S as small positive values, and the off diagonal values as small negative values.

5.1.3.3 Feedforward Neural Networks

Deep feedforward neural networks are implemented under a class called `DeepFNN`. This class assumes the trial wave function is given by

$$\log \Psi_{\boldsymbol{\theta}}(\boldsymbol{X}) \equiv \rho(\boldsymbol{v}(\boldsymbol{X})), \tag{5.45}$$

where ρ is a feedforward neural network with a single output, and $\boldsymbol{v}(\boldsymbol{X})$ is constructed by the `format` function belonging to the `Input` object. In addition, ρ is assumed to be rectangular in structure, meaning all the hidden layers have the same number of nodes.

`DeepFNN` consists of a sequence of `DenseLayer` and `ActivationLayer` objects, which represent the affine transformations and the nonlinear, element-wise transformations, respectively. Both `DenseLayer` and `ActivationLayer` are derived from the abstract base class `Layer`. In addition to simple functions that facilitate the flattening and unflattening of the parameters, the `Layer` objects have the following key member functions:

```
// layer.hpp
virtual void forward(vec& h) = 0;
virtual void forward(vec& h, mat& dh) = 0;
virtual void forward(vec& h, mat& dh, mat& d2h) = 0;
virtual void backward(int& iter, vec& gradient, vec& du_dh) = 0;
```

The `forward` functions are used during the computations of the (log of the) wave function, the local gradient with respect to the coordinates, and the sum of local Laplacians. The `backward function` is used during the computation of the local gradient of the trial wave function with respect to the variational parameters.

Even though activation functions are not formally considered layers of the network themselves, it is convenient to implement them as layers because the forward-passing and backward-passing steps can be written extremely simply. More specifically, if we store all the `DenseLayer` and `ActivationLayer` objects into a single `std::vector` of pointers to each layer, we can write the wave function computation as the following:

```
// deepfnn.cpp
long double DeepFNN::get_logpsi(const mat& X)
{
    pInput_->format(X, h_);
    for(int l = 0; l < n_layers_; ++l) pLayers_[l]->forward(h_);
    return h_(0);
}
```

Similarly, the gradient with respect to the parameters is computed as the following:

```
// deepfnn.cpp
vec DeepFNN::get_param_gradient(const mat& x)
{
    // forward pass to store intermediate values
    pInput_->format(x, h_);
    for(int l = 0; l < n_layers_; ++l) pLayers_[l]->forward(h_);

    // backward pass to calculate gradient
    iter_ = 0;
    du_dh_ = vec::Ones(1);
    for(int l = n_layers_-1; l >= 0; --l) pLayers_[l]->backward(iter_,
                                                                param_grad_,
                                                                du_dh_);
    return param_grad_;
}
```

Notice that the above gradient calculation includes a forward pass. This is because the gradient of the output depends on the input for both `DenseLayer` and `ActivationLayer`. Thus, during the forward step, a copy of the input vector is stored for later use.

The forward pass for the `DenseLayer` object is exceptionally simple, as it consists of only a linear transformation.

```
// dense.cpp
void DenseLayer::forward(vec& h, mat& dh, mat& d2h)
{
    h = W_ * h + b_;
    dh = W_ * dh;
    d2h = W_ * d2h;
}
```

Similarly, the backward pass can be written simply as well.

```
// dense.cpp
void DenseLayer::backward(int& iter, vec& gradient, vec& du_dh)
{
    du_dW_ = du_dh * h_in_.transpose();
    du_db_ = du_dh;
    du_dh = W_.transpose() * du_dh;

    gradient.segment(iter, n_params_) = concat(flatten(du_dW_), du_db_);
    iter += n_params_;
}
```

In the above, the character `u` represents the single output of the entire network, written as ρ in Eq. (5.45). The first three lines compute the gradient of the output with respect to the weights and biases, as well as the gradient with respect to the hidden nodes. Then the gradients with respect to the parameters are flattened and stored in their respective locations in the incoming `gradient` vector, which stores the gradients with respect to *all* weights and biases in the network. These weights and biases are initialized using the Glorot normal scheme and the biases are initialized to zero.

On the other hand, the forward step for the `ActivationLayer` is slightly more complicated because of the nonlinearity and the block organization of the gradient and Laplacian.

```
// activation.cpp
void Activation::forward(vec& h, mat& dh, mat& d2h)
{
    for(int i = 0; i < h.size(); ++i)
    {
        d2h.row(i) *= get_df(h(i));
        for(int p = 0; p < n_parts_; ++p)
        {
            d2h(i,p) += get_d2f(h(i))
                        * dh.row(i).segment(p*n_dims_, n_dims_).squaredNorm();
        }
        dh.row(i) *= get_df(h(i));
        h(i) = get_f(h(i));
    }
}
```

Here, the functions `get_f`, `get_df`, and `get_d2f` are pure virtual functions representing $f(x)$, $f'(x)$ and $f''(x)$, respectively. They must be defined in each class derived from `ActivationLayer`. Luckily, the backward step is even simpler than for the `DenseLayer` because there are no trainable parameters.

```
// activation.cpp
void Activation::backward(int& iter, vec& gradient, vec& du_dh)
{
    for(int i = 0; i < du_dh.size(); ++i) du_dh(i) *= get_df(h_in_(i));
}
```

Again, the backward pass involves the original input vector, which was saved in the member `h_in_`.

As a final note, `DeepFNN` is often accompanied by an enveloping `FixedGaussian`, but we will omit discussion on the latter as it is straightforward to implement and contains no variational parameters. Products of `DeepFNN` and `FixedGaussian` are generated by `ProductWaveFunction`, discussed in Sec. 5.1.3.6.

5.1.3.4 Deep Sets

The present version of `NeuralAnsatz` also provides a simple implementation of Deep Sets in a class called `DeepSet`. In this case, the trial wave function is given by

$$\log \Psi_{\boldsymbol{\theta}}(\boldsymbol{X}) \equiv \rho\Big(\boldsymbol{pool}\big(\{\boldsymbol{\phi}(\boldsymbol{v}_i(\boldsymbol{X}))\}\big)\Big), \tag{5.46}$$

where ρ is a feedforward neural network with a single output, $\boldsymbol{\phi}$ is feedforward neural network with a vector output, and ***pool*** is a pooling operation specified by a `PoolingLayer`. Additionally, the index i in $\boldsymbol{v}_i(\boldsymbol{X})$ may run over the total number of particles N *or* the total number of pairs $N(N-1)/2$, depending on which `Input` object is chosen. The overall structure of `DeepSet` is very similar to `DeepFNN` except that there are now *two* `std::vector` members of pointers to the `Layer` objects corresponding to ρ and $\boldsymbol{\phi}$, and an additional pointer to the `PoolingLayer`. For example, the local laplacian is implemented as the following:

```
// deepset.cpp
long double DeepSet::get_local_laplacian(const mat& X)
{
    pInput_->format(X, h_, dh_, d2h_);
    for(int l = 0; l < n_layers1_; ++l) pLayers1_[l]->forward(h_, dh_, d2h_);
    pPooling_->forward(h_, dh_, d2h_);
    for(int l = 0; l < n_layers2_; ++l) pLayers2_[l]->forward(h_, dh_, d2h_);
    return dh_.squaredNorm() + d2h_.sum();
}
```

As a side note, the elements of `pLayers1_` include pointers to objects derived from `DenseLayer`, called `DenseSet`, that passes elements of the set forward and backward in parallel rather than the entire input at once.

The more important change to discuss is the inclusion of the `PoolingLayer`. As a simple illustration, see the implementation of `SumPooling` below.

```
// sum.cpp
void SumPooling::forward(vec& h)
{
    h_out_.setZero();
    for(int s = 0; s < n_set_; ++s) h_out_ += h.segment(s * n_out_, n_out_);
    h = h_out_;
}
```

Since the ordering of the set elements is destroyed in the forward process, and each set element is given equal weight to the eventual output, the backward direction can be completed by essentially creating `n_set_` copies of the gradient and passing it back to the `DenseSet` sequence. Then the `DenseSet` layers average the contributions to the gradient from each element of the set. If the set elements have unequal weights in the eventual output, the inputs need to be stored to compute the corresponding weights in the backward direction.

5.1.3.5 Slater Determinants

The implementation of `SlaterDeterminant` in `NeuralAnsatz` is limited by the fact that our sampling procedure involves perturbing all particles in a single Monte Carlo step, rather than perturbing one particle at a time. In addition, we include the option to sample spins as well, where the number of spin-up and spin-down particles are not necessarily equal. Consequently, we cannot take advantage of commonly-employed tricks involving updating a single row or single column of the determinant matrix, nor is it straightforward to split the determinant into a product of two determinants corresponding to the spin-up and spin-down particles. Then, the analytical computations of the gradients and Laplacians of the Slater

determinant with respect to the particle coordinates require taking the determinant *and* the inverse of the matrix. According to `Eigen`'s documentation, both of these operations are only guaranteed to be stable for at most 4×4 matrices. Nonetheless, we describe the implementation of `SlaterDeterminant` here, but we will exercise caution for systems larger that $N = 4$.

A Slater determinant is given by

$$\Psi(\boldsymbol{X}) \equiv \frac{1}{\sqrt{N!}} \det\left[\varphi_\alpha(\boldsymbol{x}_i)\right] = \frac{1}{\sqrt{N!}} \det \begin{bmatrix} \varphi_1(\boldsymbol{x}_1) & \varphi_1(\boldsymbol{x}_2) & \cdots & \varphi_1(\boldsymbol{x}_N) \\ \varphi_2(\boldsymbol{x}_1) & \varphi_2(\boldsymbol{x}_2) & \cdots & \varphi_2(\boldsymbol{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_N(\boldsymbol{x}_1) & \varphi_N(\boldsymbol{x}_2) & \cdots & \varphi_N(\boldsymbol{x}_N) \end{bmatrix}, \tag{5.47}$$

where φ_α are single-particle spin-orbitals indexed by α, and we have omitted the subscript $\boldsymbol{\theta}$ on the ansatz $\Psi_{\boldsymbol{\theta}}(\boldsymbol{X})$ because we do not consider trainable spin-orbitals in this implementation. As mentioned in Sec. 5.1.2, when a system of `Fermions` is constructed, the number of distinct spatial orbitals that is required in the Slater determinant is automatically stored. The member `orbital_index_` acts as the mapping between orbital index α and the quantum numbers of the spatial orbitals. At the same time, the spin components of the spin-orbitals are determined and stored in the member `spin_orbitals_`. Together, `orbital_index_` and `spin_orbitals_` determine each unique spin-orbital.

The orbitals are computed by a `Basis` object which takes the quantum numbers of the spatial orbitals stored in `orbital_index_` and evaluates the orbitals for all particles. For the spin components, the `Fermions` class computes the spin projections using the information stored in `spin_orbitals_`. One can simply take the element-wise product to fill the elements

of the matrix, and then take the determinant.

```
// slaterdeterminant.cpp
long double SlaterDeterminant::get_logpsi(const mat& X)
{
    D_ = (pBasis_->get_states(X)).cwiseProduct(pFermions_->get_spin_projection
        (X));
    det_ = D_.determinant();
    return logl(fabsl(det_)) + lognorm_;
}
```

Storing a copy of the determinant in `det_` allows the `Hamiltonian` to access the sign of the determinant if the interaction is spin dependent.

To write the local gradient with respect to a particle's coordinates, we first define the matrix

$$D(\boldsymbol{X}) \equiv \left[\varphi_\alpha(\boldsymbol{x}_i)\right], \tag{5.48}$$

for brevity, and use Jacobi's formula,

$$\frac{1}{\Psi(\boldsymbol{X})}\boldsymbol{\nabla}_i\Psi(\boldsymbol{X}) = \mathrm{tr}\left(D(\boldsymbol{X})^{-1}\boldsymbol{\nabla}_i D(\boldsymbol{X})\right), \tag{5.49}$$

where $\boldsymbol{\nabla}_i D(\boldsymbol{X})$ means to evaluate the derivative for each spatial dimension of particle i, and the subsequent trace is taken separately for each direction. For the local Laplacian, we also use that $\boldsymbol{\nabla}_i D(\boldsymbol{X})^{-1} = -D(\boldsymbol{X})^{-1}\boldsymbol{\nabla}_i D(\boldsymbol{X})D(\boldsymbol{X})^{-1}$,

$$\begin{aligned}\frac{1}{\Psi(\boldsymbol{X})}\boldsymbol{\nabla}_i^2\Psi(\boldsymbol{X}) = \mathrm{tr}\left(D(\boldsymbol{X})^{-1}\boldsymbol{\nabla}_i^2 D(\boldsymbol{X})\right) + \mathrm{tr}^2\left(D(\boldsymbol{X})^{-1}\boldsymbol{\nabla}_i D(\boldsymbol{X})\right) \\ - \mathrm{tr}\left(D(\boldsymbol{X})^{-1}\boldsymbol{\nabla}_i D(\boldsymbol{X})D(\boldsymbol{X})^{-1}\boldsymbol{\nabla}_i D(\boldsymbol{X})\right).\end{aligned} \tag{5.50}$$

The computation of the spatial parts of $\boldsymbol{\nabla}_i D(\boldsymbol{X})$ and $\boldsymbol{\nabla}_i^2 D(\boldsymbol{X})$ are handled by the `Basis`.

5.1.3.6 Products

The wrapper class `ProductWaveFunction`, derived from `WaveFunction`, enables the creation of products involving any desired number F of `WaveFunction` factors,

$$\Psi_{\boldsymbol{\theta}}(\boldsymbol{X}) \equiv \prod_{f=1}^{F} \Psi_{\boldsymbol{\theta}_f}(\boldsymbol{X}), \tag{5.51}$$

where the parameters are concatenated together $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_F)$. It is straightforward to show that the local gradient with respect to a particle's coordinates is given by

$$\frac{1}{\Psi_{\boldsymbol{\theta}}(\boldsymbol{X})} \nabla_i \Psi_{\boldsymbol{\theta}}(\boldsymbol{X}) = \sum_{f=1}^{F} \frac{1}{\Psi_{\boldsymbol{\theta}_f}(\boldsymbol{X})} \nabla_i \Psi_{\boldsymbol{\theta}_f}(\boldsymbol{X}), \tag{5.52}$$

and the local Laplacians are given by

$$\begin{aligned} \frac{1}{\Psi_{\boldsymbol{\theta}}(\boldsymbol{X})} \nabla_i \Psi_{\boldsymbol{\theta}}(\boldsymbol{X}) = & \sum_{f=1}^{F} \frac{1}{\Psi_{\boldsymbol{\theta}_f}(\boldsymbol{X})} \nabla_i^2 \Psi_{\boldsymbol{\theta}_f}(\boldsymbol{X}) \\ & + 2 \sum_{f<g}^{F} \left(\frac{1}{\Psi_{\boldsymbol{\theta}_f}(\boldsymbol{X})} \nabla_i \Psi_{\boldsymbol{\theta}_f}(\boldsymbol{X}) \right) \cdot \left(\frac{1}{\Psi_{\boldsymbol{\theta}_g}(\boldsymbol{X})} \nabla_i \Psi_{\boldsymbol{\theta}_g}(\boldsymbol{X}) \right). \end{aligned} \tag{5.53}$$

The utilization of the `ProductWaveFunction` becomes necessary when working with a `SlaterDeterminant` or when employing a `FixedGaussian` as an enveloping function. Furthermore, different neural-network quantum states can be combined to achieve the best of both (or all) worlds.

5.1.4 Hamiltonian

The `Hamiltonian` class is an abstract base class that takes `System` and `WaveFunction` objects as input to its constructor. By default, it is the free Hamiltonian which just contains the

kinetic energy operator. Its virtual member functions `get_external` and `get_interaction` both return zero, unless they are overridden by derived classes.

```
// hamiltonian.cpp
long double Hamiltonian::get_local_energy(const mat& X)
{
    return get_local_kinetic(X) + get_external(X) + get_interaction(X);
}

long double Hamiltonian::get_local_kinetic(const mat& X)
{
    return -hbar2m_ * pWaveFunc_->get_local_laplacian(X);
}
```

If the interaction potential contains a singularity, it is strongly-recommended to regularize the singularity by introducing a hyperparameter r_0 and transforming as

$$\frac{1}{r^k} \mapsto \frac{f^k(r/r_0)}{r^k}, \tag{5.54}$$

where $f(r)$ is a function that scales as $\sim r$ for small r and approaches 1 for large r. By doing so, the long-range behavior of the original interaction is preserved, but at $r = 0$, the potential is finite, taking a value of $(1/r_0)^k$. We typically choose $f(r) \equiv \tanh(r)$. Pretraining the neural-network quantum states on a softer potential before lowering the value of r_0 to tackle harder potentials has been found to significantly stabilize and accelerate performance. In addition, this allows the neural-network quantum states to discover the cusp condition on it's own, rather than providing it from the very beginning, demonstrating their representation power.

It should be noted that adding new `Hamiltonian` objects is extremely straightforward, as one only needs to specify the external and interaction potentials for a single

configuration sample X. Since the neural-network quantum states are entirely separate from the Hamiltonian, one can swap different interactions without affecting the states.

5.1.5 Sampler

`Metropolis` and `Importance` are classes derived from the abstract `Sampler` class. The `Sampler` class also provides the option to sample the spins as well. The implementations follow the descriptions provided in Sec. 3.1.1.

In addition to pointers to the `System` and `WaveFunction` objects, the class constructor takes `n_eq_init`, `n_eq`, `n_void`, `sigma0`, `step` as arguments. The integer `n_void` specifies the number of samples to skip between each effective sample, constituting one Monte Carlo cycle. At the very beginning of the simulation, initial positions are drawn from a Gaussian with standard deviation `sigma0`, and the sampler is equilibrated using `n_eq_init` Monte Carlo cycles. Instead of equilibrating the sampler from scratch at the beginning of each training epoch, we recover the last configuration and equilibrate using only `n_eq` cycles. Typically, `n_eq` can be one or two orders of magnitude smaller than `n_eq_init`. This construction assumes the learning rate is not so large that the wave function changes significantly in a single epoch. The parameter `step` determines the size of the spatial perturbations.

5.1.6 Optimizer

The abstract `Optimizer` class takes a pointer to the `WaveFunction` object, the initial learning rate `eta0`, and the total number of epochs `n_epochs` as input. Derived classes only need to provide a definition for the following pure virtual function.

```
// optimizer.hpp
```

```
virtual void update_params(vec gradient) = 0;
```

This function simply takes a gradient vector and updates the parameters of the `WaveFunction` object. The specific optimization algorithms were discussed in Sec. 3.2.2.

5.1.7 Cost Functions

The `CostFunction` computes both the cost and the gradient of the cost with respect to the parameters. Derived classes can be designed for reinforcement learning or for supervised learning. In variational Monte Carlo, we use the `Energy` as the cost, which has the following constructor signature.

```
// energy.hpp
Energy(int n_samples,
       long double epsilon,
       std::shared_ptr<System> pSystem,
       std::shared_ptr<WaveFunction> pWaveFunc,
       std::shared_ptr<Sampler> pSampler,
       std::shared_ptr<Hamiltonian> pHamiltonian);
```

If the parameter `epsilon` is larger than zero, stochastic reconfiguration is used to precondition the gradient. To minimize the memory footprint of the program, samples are taken on the fly and expectation values are computed as running averages. Once all parallel processes have completed the computations of their local expectation values, `Open MPI` is used communicate between processes and aggregate the results. Since the `Energy` requires the `Hamiltonian` object as input, there is no need for the `Trainer` to have explicit ties to the `Hamiltonian`. The gradient calculation for the `Energy` is shown below.

```
// energy.cpp
vec Energy::get_gradient()
```

```
{
    reset_gradient_estimator();
    pSampler_->equilibrate();
    for(int n = 0; n < n_samples_loc_; ++n)
    {
        X_ = pSampler_->get_next_config();
        E_ = pHamiltonian_->get_local_energy(X_);
        O_ = pWaveFunc_->get_param_gradient(X_);
        R_ = X_.leftCols(n_dims_);

        E_avg_loc_ += E_;
        E2_avg_loc_ += E_ * E_;
        O_avg_loc_ += O_;
        EO_avg_loc_ += E_ * O_;
        r_avg_loc_ += R_.sum();
        r2_avg_loc_ += R_.squaredNorm();

        if(use_stochastic_reconfig_) O2_avg_loc_ += O_ * O_.transpose();
        if(pSampler_->positions_accepted()) r_acc_loc_ += 1;
        if(pSampler_->spins_accepted()) s_acc_loc_ += 1;
    }
    finalize_gradient_estimator();
    return gradient_;
}
```

If supervised learning of the `WaveFunction` is desired, the `CostFunction` needs to take a target `WaveFunction` object as input in place of the `Hamiltonian`. For instance, the constructor for the `Overlap` between two wave functions has the following signature.

```
// overlap.hpp
 Overlap(int n_samples,
         long double sigma,
         std::shared_ptr<System> pSystem,
         std::shared_ptr<WaveFunction> pWaveFunc,
         std::shared_ptr<Sampler> pSampler,
         std::shared_ptr<WaveFunction> pTargetWaveFunc);
```

The parameter `sigma` specifies Gaussian probability distribution from which samples are drawn.

5.1.8 Density

The `Density` object takes snapshots of the spatial distribution during training. If the `WaveFunction` is a product, it is possible to take snapshots of each individual factor as well. The constructor for the abstract class is given by the following.

```
// density.hpp
Density(int n_samples,
        int n_bins,
        long double bounds,
        std::shared_ptr<System> pSystem,
        std::shared_ptr<WaveFunction> pWaveFunc,
        std::shared_ptr<Sampler> pSampler,
        bool sample_factors = false);
```

Since obtaining an accurate density requires a large number of samples compared to the number of samples required to calculate the cost, the `Density` object takes its own `n_samples` parameter. While taking these samples, the `OneBodyDensity` object bins them according to their position in one-dimension, or their distance from the origin in two- and three-dimensions. Alternatively, the `TwoBodyDensity` pair distribution functions can be computed by binning the distances between all pairs.

5.1.9 Running `NeuralAnsatz`

First, clone a copy of the Github repository at this url: `https://github.com/kim-jane/NeuralAnsatz`. Ensure that both `make` and `Open MPI` are installed. On Ubuntu, this can be accomplished by running the following commands in the terminal:

```
sudo apt-get update
sudo apt install build-essential
sudo apt install make
sudo apt install openmpi-bin
```

On MacOS, one can instead run the following:

```
brew update
brew upgrade
brew install gcc
brew install make
brew install open-mpi
```

Alternatively, the appropriate modules can be loaded on a high-performance computing cluster. Next, enter repository and compile the `NeuralAnsatz` library.

```
cd ~/NeuralAnsatz
make
```

Driver files in the `examples/` folder will be compiled into corresponding executables in the `bin/` folder. For example, if there is a driver file named `examples/myprogram.cpp`, the code can be run by typing the command below.

```
mpirun -np <number of parallel tasks> ./bin/myprogram
```

Once completed, remove all binaries and build files.

```
make clean
```

5.2 Python Software for Periodic Systems

In this section, we introduce a different implementation of neural-network quantum states that leverages the flexibility and high-performance computing capability of `JAX`. This code was initially developed by Alessandro Lovato for nuclei and has subsequently been adapted by Bryce Fore and myself for periodic systems. Since some of the over-arching details of the simulation are similar to the `NeuralAnsatz` code, we will focus only on the key differences here. In addition, we will discuss the basic ideas required to implement the neural-network quantum states relevant to the work in Chapters 9 and 7.

5.2.1 `JAX`: High-Performance Array Computing

`JAX` is an open-source software that combines the flexibility and east of use of `numpy` with the efficiency of compiled languages like C++. The key features of `JAX` include:

- Compute gradients with `grad`: Easily compute gradients of arbitrary functions by creating a new function.
- Just-in-time compilation with `jit`: Functions decorated with `jit` will be dynamically compiled and cached upon first encounter.
- Vectorization with `vmap`: Apply functions elementwise to one or more arrays for efficient batch operations.

- Parallelization with `pmap`: Execute functions in parallel across multiple devices, maximizing hardware resources.

5.2.1.1 Building Neural Networks

The functionalities provided by JAX allows for far more freedom in network structure, as one only needs to worry about computing the forward direction. We begin by importing a few required modules.

```
import jax
import jax.numpy as jnp
from jax.example_libraries import stax
```

All layer construction functions within `stax` return a pair of functions `(init_func, apply_func)` that represent the initialization of a layer with random weights and the application of a layer, respectively. All initialization functions, `init_func`, take the input shape and a `jax.random.key`, and subsequently returns the new output shape and the initialized parameters. All application functions, `apply_func`, take a set of parameters and input data, then evaluates the forward direction of the layer. Multiply layers can be joined together by using `stax.serial`.

```
init_func, apply_func = stax.serial(*layers)
```

If one defines the list `layers` as an alternating sequence of `stax.Dense` layers and activation layers, e.g. `stax.Gelu`, then `stax.serial` generates the constructor functions of a standard feedforward neural network.

The separation of the `init_func` from the `apply_func` is beneficial if many copies of

identical neural network structures are needed, where each copy is initialized with different parameters. For instance, the single-particle orbitals of a Slater determinant can have one common `apply_func`, but N different sets of unique parameters by simply calling `init_func` N times.

To construct a Deep Set, on the other hand, one must create two sets of constructor functions, one that maps to the latent space and another that maps to the output space.

```
lat_init, lat_apply = stax.serial(stax.Dense(n), stax.Gelu, stax.Dense(n))
out_init, out_apply = stax.serial(stax.Dense(n), stax.Gelu, 1)
```

In the above, the integer `n` is the number of nodes in the layer. Computing the forward direction of the Deep Set amounts to calling lat_apply and out_apply with an appropriate pooling function in between. Below is an example of a Deep Set used as a Jastrow factor, where `phi_params` and `rho_params` refer to the variational parameters in the mappings to and from the latent space.

```
phi = self.lat_apply(phi_params, X)
phi = jnp.sum(phi, axis=0)
logpsi = self.out_apply(rho_params, phi)
```

Virtually all artificial neural networks, no matter how complicated the connections, can be constructed by appropriately combining feedforward neural networks, concatenation operations, and pooling operations.

5.2.2 Symmetries and Boundary Conditions

Now that we have established the basics of building a network, we must discuss the specific requirements the network must satisfy if it is to be treated as a wave function. If either of

these elements are broken, the wave function provide erroneous results. For periodic systems, we must transform the coordinate or distances such that obey strict periodicity at all times. Some options for periodic inputs with L-periodicity include:

```
# periodic positions
cos_i = jnp.cos( 2 * jnp.pi * r / L )
sin_i = jnp.sin( 2 * jnp.pi * r / L )

# periodic separation vectors
r_ij = r[ ip[k] ] - r[ jp[k] ]
r_ij = r_ij - jnp.rint( r_ij / self.L ) * self.L
cos_ij = jnp.cos( 2 * jnp.pi * r_ij / self.L )
cos_ij = jnp.sin( 2 * jnp.pi * r_ij / self.L )

# periodic distance
d_ij = jnp.sin( jnp.pi * r_ij / self.L )
d_ij = jnp.sum( d_ij**2 )
```

In addition to the periodicity and the antisymmetry of the fermionic wave function, which is constrained by a Slater determinant or a Pfaffian, discrete symmetries such as parity and time-reversal can be enforced. Below is an example of parity and time-reversal enforcement for a system with even parity and odd time-reversal symmetry.

```
@jit
def logsignpsi_PT(params, r, sz):

    r_PT = jnp.stack((r, -r, r, -r))
    sz_PT = jnp.stack((sz, sz, -sz, -sz))

    sign, logpsi = vmap(logsignpsi, in_axes=(None, 0, 0))(params, r_PT, sz_PT)
    sign = sign.at[2:].set( -sign[2:])

    logpsi_PT, sign_PT = jax.scipy.special.logsumexp(a=logpsi,
                                                     b=sign,
                                                     return_sign=True)
    return sign_PT, logpsi_PT
```

5.2.3 Computing the Pfaffian

Unfortunately, `JAX` does not provide built-in implementation of the Pfaffian of a matrix, nor does it provide convenient decomposition schemes useful for the Pfaffian efficiently. For our work on ultra-cold Fermi gases, we implemented the Pfaffian ourselves, complete with pivoting. In the code below, we assume `A` is even dimensional and symmetric. The algorithm is implemented according to Ref. [13].

```
@jit
def pfaffian(self, A):

    def pivot(kp, k, A, pf_A):

        # exchange rows
        temp = A.at[k+ 1, :].get()
        A = A.at[k + 1, :].set(A.at[kp, :].get())
        A = A.at[kp, :].set(temp)

        # exchange columns
        temp = A.at[:, k + 1].get()
        A = A.at[:, k + 1].set(A.at[:, kp].get())
        A = A.at[:, kp].set(temp)

        # flip sign
        pf_A *= -1.0
        return A, pf_A

    def no_pivot(kp, k, A, pf_A):
        return A, pf_A

    pf_A = 1.0
    for k in range(0, A.shape[0]-1, 2):

        # find pivot index
        kp = k + 1 + jnp.argmax(jnp.abs(A[k + 1:, k]))

        # pivot if necessary
        A, pf_A = cond(kp != k + 1, pivot, no_pivot, kp, k, A, pf_A)

        # update pfaffian
        pf_A *= A[k, k + 1]

        # update matrix
        mu = A.at[k, :].get() / A[k, k+1]
        nu = A.at[:, k+1].get()
        A = A + jnp.outer(mu, nu) - jnp.outer(nu, mu)

    return jnp.sign(pf_A), jnp.log(jnp.abs(pf_A))
```

5.2.4 Stochastic Reconfiguration with RMSprop Regularization

One of the unique components of the `JAX`-based VMC code is the alternative regularization scheme used for the Fisher-information matrix. As discussed in Sec. 3.2.2.5, the inversion of the Fisher-information matrix S is typically stabilized by offsetting the diagonal elements by a small ϵ, usually chosen between $10^{-5} - 10^{-2}$. However, by choosing a constant shift, the relative magnitudes of the variational parameters are ignored, ultimately leading to an inefficient optimization process. In the alternative scheme based on RMSprop, a running average of the square of the gradients are stored to normalize the diagonal shift according to each parameter. When the optimizer is initialized, $\boldsymbol{v}$ is set to zero. Then during each parameter update,

$$\boldsymbol{v} \leftarrow\!\shortmid\, \beta \boldsymbol{v} + (1-\beta)\big(\boldsymbol{\nabla}_{\boldsymbol{\theta}} \langle E_{\boldsymbol{\theta}}(\boldsymbol{X})\rangle\big)^2, \tag{5.55}$$

$$\boldsymbol{\theta} \leftarrow\!\shortmid\, \boldsymbol{\theta} - \eta \left(S + \operatorname{diag}(\sqrt{\boldsymbol{v}} + \epsilon)\right)^{-1} \boldsymbol{\nabla}_{\boldsymbol{\theta}} \langle E_{\boldsymbol{\theta}}(\boldsymbol{X})\rangle. \tag{5.56}$$

6 The Calogero-Sutherland Model

To evaluate the performance of the C++ software discussed in Sec. 5.1, we benchmark against the Calogero-Sutherland model, an exactly-solvable model of interacting bosons trapped in a one-dimensional harmonic oscillator well. This model provides an excellent basis to assess our neural-network quantum states (NQS) for several reasons. Firstly, the interaction contains a singularity. Because the bosons are trapped, the effects of the singularity will be particularly important, as the likelihood of two bosons becoming close to one another is high. Learning how to handle this singularity will prove to be great practice for more realistic systems. Secondly, one-dimensional systems are easy to visualize, giving us the opportunity to gain an intuitive understanding of the two types of neural-network quantum states we will test. Lastly, the availability of exact analytical solutions enables us to validate the accuracy of our implementation.

6.1 Hamiltonian

The Hamiltonian of the Calogero-Sutherland model is given by

$$\hat{H}^{\mathrm{cs}} = \sum_{i=1}^{N} \left(-\frac{1}{2}\frac{\partial^2}{\partial x_i^2} + \frac{1}{2}\omega^2 x_i^2 \right) + \sum_{i<j}^{N} \frac{\beta(\beta-1)}{x_{ij}^2}, \tag{6.1}$$

where β is an interaction parameter, $x_{ij} = |x_i - x_j|$ is the distance between particles i and j, and we have set $\hbar = m = 1$ for convenience. During our discussion on Kato's cusp condition in Sec. 3.2.1.1, we showed that requiring the local energy to be finite as any two particles

approach each other led us to the exact ground state of the system

$$\Psi_0^{\text{cs}}(\boldsymbol{X}) = \exp\left(-\sum_{i=1}^{N}\frac{1}{2}\omega x_i^2\right)\prod_{i<j}x_{ij}^{\beta}, \tag{6.2}$$

with the corresponding energy

$$E_0^{\text{cs}} = \frac{1}{2}N\omega + \frac{1}{2}\beta N(N-1)\omega = \frac{1}{2}N\omega\big(1+\beta(N-1)\big). \tag{6.3}$$

Throughout this work, we will set $N = 6$ and $\beta = 2$, resulting in an exact interacting energy that is 11 times greater than the noninteracting harmonic oscillator ground state energy.

To prevent numerical instabilities caused by the singularity in Eq. (6.1), we regularize the two-body potential as

$$\frac{1}{x^2} \mapsto \frac{\tanh^2(x/x_0)}{x^2}, \tag{6.4}$$

where x_0 is a regularization hyperparameter that smoothly controls the height of the potential, while preserving the long-range behavior. As $x_0 \to 0$, the regularized potential converges to the original potential containing the singularity. See Fig. 6.1 to compare the regularized potential using different values of x_0 and $\beta = 2$.

Our general strategy is to first train the neural-network quantum states using a fairly large value of the regularization parameter x_0. Then after the training has stabilized, we progressively decrease the value of x_0 until the energy converges. Another possibility could be to fix x_0 to a small value, then gradually increase β from a value slightly greater than 1 to our desired value of 2. Both of these strategies are examples of transfer learning, as they involve pretraining the NQS on easier tasks before transitioning to more challenging ones.

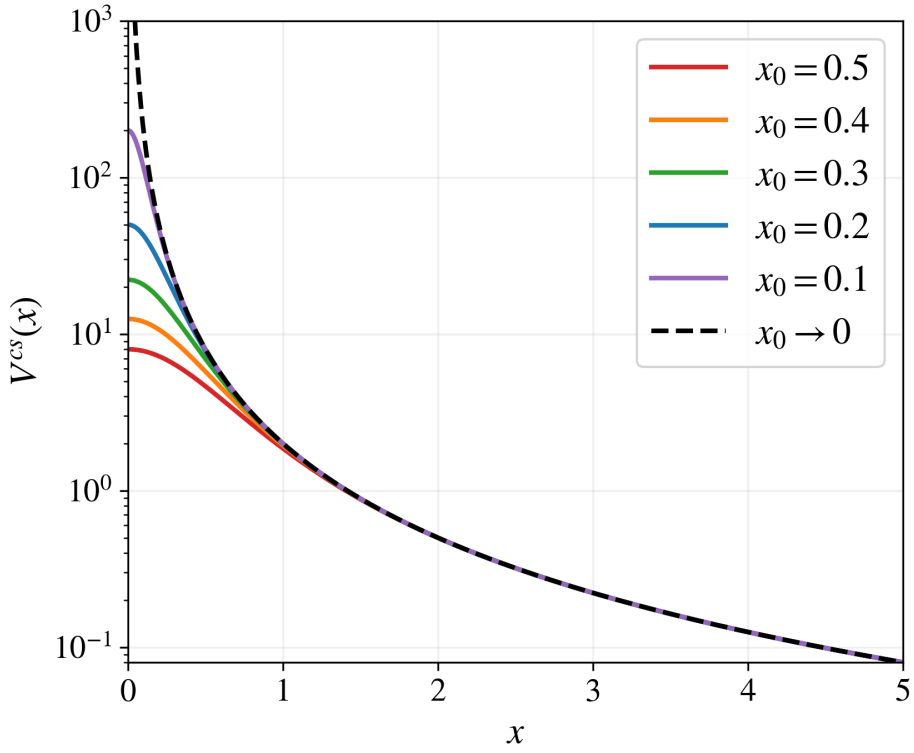

Figure 6.1: The regularized two-body potential for the Calogero-Sutherland model, where x_0 is the regularization parameter (Eq. (6.4)). The long-range behavior of the original potential (black, dashed line) is preserved, but the height of the potential decreases with larger values of x_0. Here, we have used an interaction parameter value of $\beta = 2$.

6.2 Neural-Network Quantum States

This work focuses on a comparative analysis of two distinct types of neural-network quantum states: feedforward neural networks and continuous restricted Boltzmann machines. The inputs for both networks comprise the positions of all particles, with permutation invariance guaranteed by sorting the positions. Sorting the inputs was discussed in depth in Sec. 5.1.3.1. For brevity, we will use the notation $\boldsymbol{v}(\boldsymbol{X})$ to imply the positions $\boldsymbol{X} = (x_1, x_2, ..., x_N)$ have been sorted and serve as the visible nodes $\boldsymbol{v}$ of the networks.

6.2.1 Feedforward Neural Networks

Inspired by Saito in Ref. [14], we take our positive-definite Jastrow ansatz to be

$$\Psi_{\boldsymbol{\theta}}^{\text{fnn}}(\boldsymbol{X}) = \exp\left(\rho(\boldsymbol{v}(\boldsymbol{X}))\right) \exp\left(-\frac{\boldsymbol{X}^2}{2\sigma^2}\right), \tag{6.5}$$

where $\rho(\boldsymbol{v}(\boldsymbol{X}))$ is the single output of a feedforward neural network, which takes sorted positions as input. Since neural networks are initialized with small random values around zero, ρ initially takes a value around zero as well. To ensure we sample from a localized distribution rather than a uniform distribution of infinite range, our ansatz includes a Gaussian envelope with a variance σ^2 large enough to not significantly affect the training of the feedforward neural network. In this work with $N = 6$, we have set $\sigma^2 = 16^2 = 256$. Alternatively, one could remove the Gaussian envelope and pretrain the ansatz to a Gaussian instead, either by turning off the interaction entirely, leaving a harmonic oscillator potential, or through supervised training of the wave function. We opt for the Gaussian envelope as it removes the need for pretraining, but the boundary conditions imposed by the alternative approaches are generally upheld by the network during the

reinforcement learning process.

6.2.2 Continuous Restricted Boltzmann Machines

We also employ neural-network quantum states parameterized by the various types of multivariate Gaussian-binary and Gaussian-uniform restricted Boltzmann machine, discussed in Sec. 4.7.1.2 and Sec. 4.7.1.3. Since they are all positive-definite by default, we can simply write

$$\Psi_{\boldsymbol{\theta}}^{\mathrm{rbm}}(\boldsymbol{X}) = \mathcal{P}(\boldsymbol{v}(\boldsymbol{X})), \tag{6.6}$$

where $\mathcal{P}(\boldsymbol{v}(\boldsymbol{X}))$ denotes the marginal probability distribution of the visible nodes, with $\boldsymbol{v}(\boldsymbol{X})$ representing the sorted positions. When the variational parameters are set to small random values, the initial state of the RBMs are similar to a Gaussian. We can bias our initial state to more closely resemble the eventual wave function we expect by initializing the visible biases $\boldsymbol{a}$ across a much wider range. Moreover, we sort the visible biases in order to bring them closer to the mean of the corresponding sorted positions.

6.3 Results

We begin our analysis by testing four commonly employed activation functions for a fixed feedforward neural network (FNN) architecture. For $N = 6$, we choose to use a network with two hidden layers and 12 hidden nodes each. The regularization parameter is initially set to $x_0 = 0.5$ in order to guarantee fast and stable training. Additionally, we utilize the stochastic reconfiguration (SR) algorithm, with a diagonal offset of $\epsilon = 0.001$, which accelerates training significantly. As the SR algorithm manipulates the gradient such that the curvature of the energy landscape is encoded, we have found that a simple stochastic gradient descent optimizer with a constant learning rate of $\eta = 0.001$ provides the most

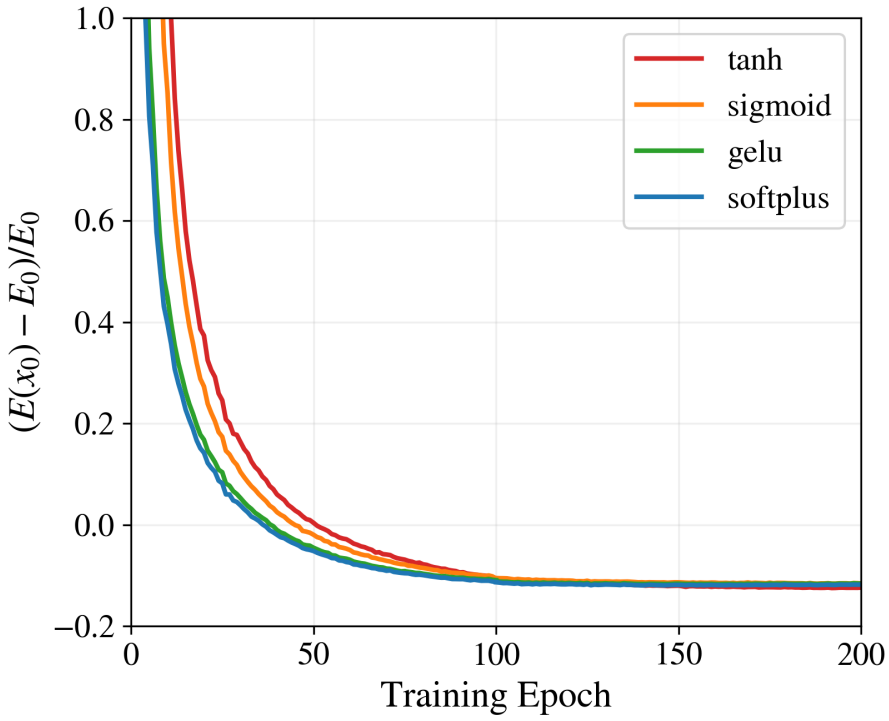

Figure 6.2: Training curves for feedforward neural networks with different activation functions. In this initial pretraining stage, $x_0 = 0.5$.

stable training.

In Fig. 6.2, we validate the efficient training of the feedforward neural networks. For each activation function, about 200 iterations were sufficient to reach convergence. Notice that the converged value of the energy is well below 0, an artifact of the regularization of the potential. At this initial stage, the performances of all activation functions are essentially equal, with the sigmoid activation function yielding an energy slightly greater hyperbolic tangent, the Gaussian error linear unit (gelu), and softplus.

We also validate the training of the Gaussian-binary and Gaussian-uniform restricted Boltzmann machines, where we consider both non-symmetric and symmetric hidden node values for each type. Since the computed marginal probability distribution of the visible

nodes for each RBM type only differed by the definition of a function we simply called "$f(x)$," one could interpret these functions as activation functions. In Fig. 6.3, we show the training curves for all four variants, each with 6 hidden nodes. Similar to the FNN case, we use the SR algorithm with $\epsilon = 0.001$ to compute the preconditioned gradient. However, for the RBMs we use a smaller learning rate of $\eta = 0.0005$.

While the RBMs yield an initial energy that is much closer to the eventual converged value, training still required about 3000 iterations, an order of magnitude more than the FNNs. The converged value of the energy still did not reach quite as low as the FNNs, reaching a minimum percent difference around $(E(x_0) - E_0)/E_0 = -0.0941$ for all RBM types. Here, E_0 is the exact ground state energy in the limit $x_0 \to 0$. In constrast, the FNNs reached a minimum between $(E(x_0) - E_0)/E_0 = -0.113 \sim -0.119$. In addition, notice that the training curves for the RBMs with symmetrically-valued hidden nodes (shown in orange and blue) show slight deviations from the nonsymmetrically-valued hidden nodes (red and green) after about ~ 250 training iterations. This may suggest that both needed to overcome a local minimum before proceeding further to the ground state. During this initial pretraining stage, the multivariate Gaussian-Uniform restricted Boltzmann machine with hidden node values between 0 and 1 had the most stable training curve.

In order to better predict the ground state of the full Calogero-Sutherland Hamiltonian, we progressively decrease the value of the regularization parameter x_0 from a value of 0.5 to 0.01. In Figures 6.4 and 6.5, we show the converged energies as a function of x_0, both the FNNs and RBMs, respectively. For the FNNs, most of the activation functions exhibit similar behavior as x_0 is decreased, except for the sigmoid activation function. This difference in performance is somewhat expected, as the sigmoid activation function is known to be prone to the vanishing-gradient problem. The other three activation functions perform similarly as

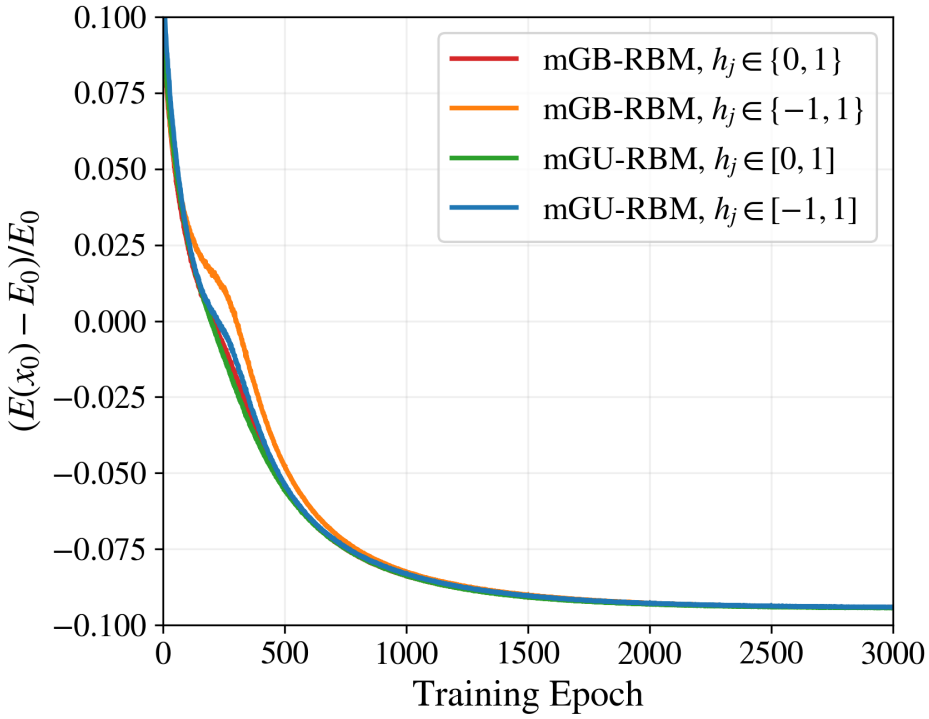

Figure 6.3: Training curves for the different variants of Gaussian-binary and Gaussian-uniform restricted Boltzmann machines, with $x_0 = 0.5$.

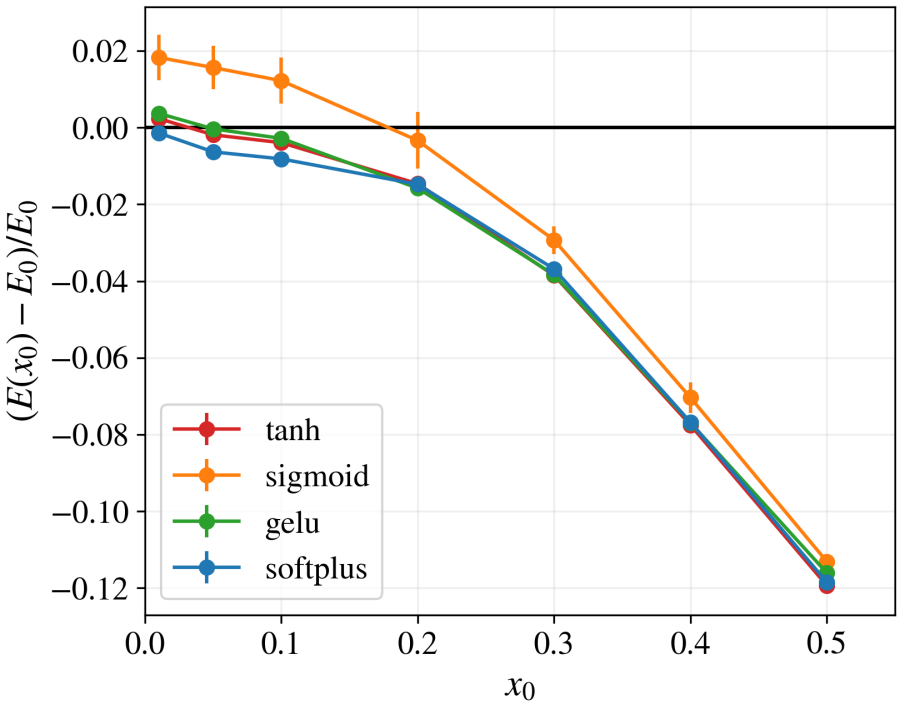

Figure 6.4: The converged energy as a function of the regularization parameter x_0 for feedforward neural networks with different activation functions.

x_0 is decreased, with the softplus function consistently producing the lowest energies. The tanh, gelu, and softplus functions all seem to converge to the exact ground state as expected.

On the other hand, the different restricted Boltzmann machines exhibit almost identical behavior as x_0 is decreased, with slight deviations occuring for the smallest values of x_0. In particular, both of the Gaussian-uniform RBMs perform slightly better than the Gaussian-binary RBMs for $x_0 = 0.01$. However, unlike the FNNs which usually converge to the exact ground state

Finally, in Fig. 6.6, we plot the one-body densities for the symmetric multivariate Gaussian-Uniform restricted Boltzmann machine and the feedforward neural network with the softplus activation function. The top row shows the initial state for both networks.

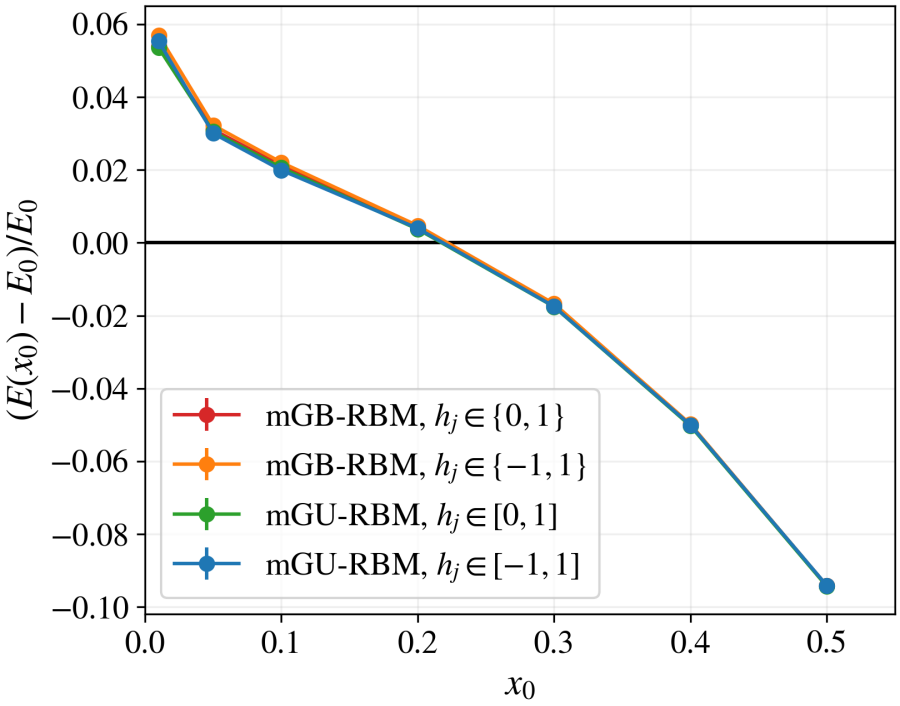

Figure 6.5: The converged energy as a function of the regularization parameter x_0 for the various Gaussian-binary and Gaussian-uniform restricted Boltzmann machines.

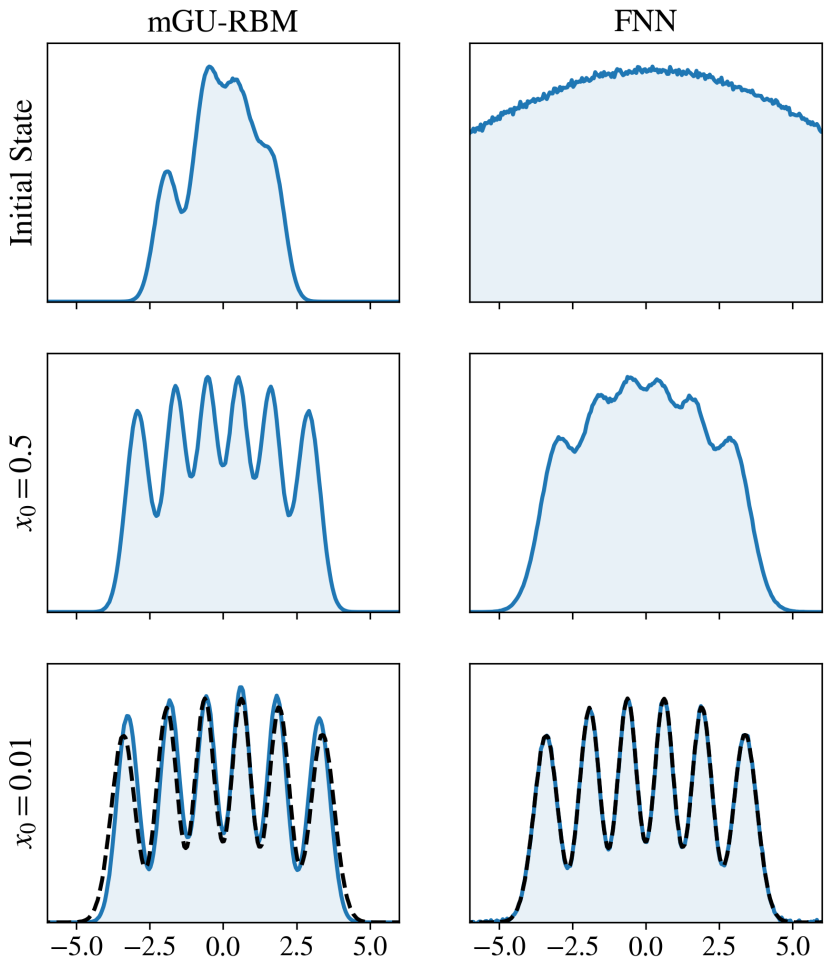

Figure 6.6: One-body densities during three different points during training: the initial stage, after pretraining with $x_0 = 0.5$, and after training with $x_0 = 0.01$. The black dashed line represents the one-body distribution from the exact ground state wave function.

The RBM naturally provides localized initial state with some distinguished peaks, demonstrating the highly-biased nature of the ansatz. As x_0 is decreased, the location of the peaks match well with the exact ground state, but there are small deviations in the shape of the peaks and in the tails of the distribution. The FNN is a low-bias model, and consequently, we used a Gaussian envelope to localize the distribution around 0. After the initial pretraining stage using $x_0 = 0.5$ the wave function learned by the FNN has significantly different shape to the one learned by the RBM, as the peaks are far less distinguished in the former. After decreasing the regularization parameter to $x_0 = 0.05$, the FNN ansatz matches the exact ground state almost perfectly.

6.4 Conclusions

In this illustrative example of neural-network quantum states, we demonstrate a transfer learning technique beneficial for handling singularities in the potential. By comparing different activation functions, we confirm that the sigmoid function suffers from a vanishing gradient, but nevertheless, still performs better than all the RBMs tested near the $x_0 \to 0$ limit.

The RBMs perform similarly to one another, with the Gaussian-uniform variants performing slightly better as x_0 decreases. However, due to the high-biased nature of the model, RBMs are limited in being able to accurately capture all behaviors in the wave function. We visualize this fact by investigating the evolution of the one-body distributions throughout the training process. Overall, a feedforward neural network proves to be the more accurate model and is therefore recommended for use as a neural-network quantum state, even if it requires comparatively more variational parameters.

Initial investigations of products of RBMs and FNNs show promise, as this eliminates

the need for a Gaussian envelope and reduced the overall number of parameters required in the FNN. We leave a more thorough report on these products for future works.

7 Dilute Neutron Matter

The following article titled '**Dilute neutron star matter from neural-network quantum states**' was published in Physical Review Research (Vol. 5, No. 3) on July 31, 2023 [15].

Low-density neutron matter is characterized by fascinating emergent quantum phenomena, such as the formation of Cooper pairs and the onset of superfluidity. We model this density regime by capitalizing on the expressivity of the hidden-nucleon neural-network quantum states combined with variational Monte Carlo and stochastic reconfiguration techniques. Our approach is competitive with the auxiliary-field diffusion Monte Carlo method at a fraction of the computational cost. Using a leading-order pionless effective field theory Hamiltonian, we compute the energy per particle of infinite neutron matter and compare it with those obtained from highly realistic interactions. In addition, a comparison between the spin-singlet and triplet two-body distribution functions indicates the emergence of pairing in the $^1S_0$ channel.

7.1 Introduction

Multi-messenger astronomy has opened new windows into the state of matter at densities and isospin asymmetries that cannot be directly probed by terrestrial experiments [16, 17, 18, 19]. Concurrently, nuclear many-body theory has made considerable progress in computing the nucleonic-matter equation of state at densities corresponding to the inner core of neutron stars starting from realistic Hamiltonians [20, 21, 22, 23, 24, 24, 25]. Comparisons between theoretical predictions and astrophysical observation pose stringent constraints on models of

nuclear dynamics, particularly three-nucleon forces [26].

In this work, we focus on lower densities, $\rho \lesssim 0.04$ fm^{-3}, which are relevant to the phenomenology of the stellar inner crust and outer core. In this region, both conditions for superfluidity — strong Fermi degeneracy and an attractive interaction between neutron pairs in the $^1S_0$ channel — are believed to be met [27, 28, 29]. In addition to lowering the system's energy, the formation of Cooper pairs plays a critical role in neutrino emission [30, 31], and the phenomenology of glitches [32]. Pairing is also relevant in modeling neutron-rich nuclei, which are the subject of intense experimental activities [33].

Quantum Monte Carlo approaches [34], and in particular the auxiliary-field diffusion Monte Carlo (AFDMC) method [35] have been extensively applied to accurately compute neutron-matter properties [22, 21, 36]. In the low-density regime, AFDMC calculations have convincingly shown a depletion of the superfluid gap with respect to the Bardeen–Cooper–Schrieffer theory [37, 38]. However, because of the fermion sign problem, AFDMC predictions depend upon the starting variational wave function. For instance, the superfluid phase must be assumed *a priori* by using pfaffian wave functions [39].

Neural-network quantum states [1] (NQS) have gained popularity in solving the Schrödinger equation of atomic nuclei both in real space [40, 41, 42, 43, 44] and in the occupation-number formalism [45]. In this work, we introduce a periodic NQS suitable to model both the normal and superfluid phases of neutron matter. The ansatz is based on the "hidden-nucleon" architecture, which can model the ground-state wave functions of nuclei up to ^{16}O with high accuracy [43]. Inspired by chemistry applications [46, 47], we further improve the expressivity of the hidden-nucleon NQS using generalized backflow correlations, which generalize both the pfaffian and the spin-dependent backflow of Ref. [48].

Our model of nuclear dynamics is the leading-order pionless effective field theory ($\not\pi$EFT) Hamiltonian of Ref. [49], which qualitatively reproduces the binding energies of nuclei with up to $A = 90$ nucleons. Arguments based on the expansion around the unitary limit [50], and Brueckner-Hartree-Fock calculations of infinite nuclear matter [51], indicate that $\not\pi$EFT should provide accurate energies of dilute neutron matter. We quantitatively address this point by carrying out variational Monte Carlo calculations based on NQS that are specifically designed to model wavefunctions of nuclear systems in the presence of spatial periodicity. We compare the $\not\pi$EFT energy per particle against the phenomenological Argonne v_{18} [52] plus Urbana IX [53] (AV18+UIX) Hamiltonian used in the Akmal-Pandharipande-Ravenhall (APR) [54] equation of state. We additionally consider the local, Δ-full chiral-EFT potentials that include tritium β-decay in the fitting procedure and do not yield self-bound neutron matter [21, 36].

To better quantify the role of dynamical correlations, we evaluate the two-body spatial distribution functions, separating the spin-triplet and spin-singlet channels. We analyze the self-emergence of pairing correlations, not explicitly included in the NQS ansatz, as a function of neutron-matter density.

7.2 Method

We model the interactions among neutrons through the leading-order $\not\pi$EFT Hamiltonian "o" of Ref. [49]. The two-body contact potential is designed to reproduce the np scattering lengths and effective ranges in the $S/T = 0/1$ and $1/0$ channels. Thus, it yields a neutron-neutron scattering length of $a_{nn} = -22.5$ fm, slightly larger than the experimental value of $-18.9(4)$ fm, see [55] and references therein, while the effective range is well reproduced. The Hamiltonian also contains a repulsive three-body force that ensures the stability of nuclei

with $A \leq 3$ nucleons. As for the latter, we take the parameterization with $R_3 = 1.0$ fm, as it reproduces the binding energies of nuclei in the $A \leq 90$ mass range reasonably well. For benchmarking purposes, we also consider the leading-order ≠EFT Hamiltonian "a" with $R_3 = 1.0$ fm, which describes the trend of binding energies of light and medium-mass nuclei.

We approximate the ground-state solution of the nuclear many-body problem with an NQS ansatz that belongs to the hidden-fermion family [56], recently generalized to continuum Hilbert spaces and applied to atomic nuclei in Ref. [43]. In addition to the visible spatial and spin coordinates of the A neutrons, $R = \{\mathbf{r}_1 \dots \mathbf{r}_A\}$ and $S = \{s_1^z \dots s_A^z\}$, the Hilbert space contains fictitious A_h hidden-nucleon degrees of freedom. In this work we use $A_h = A = 14$ so that the system is as flexible as possible, but in practice we have also found using as few as 8 hidden nucleons gives very similar results. The wave function can be conveniently expressed in a block matrix form as

$$\Psi_{HN}(R,S) \equiv \det \begin{bmatrix} \phi_v(R,S) & \phi_v(R_h,S_h) \\ \chi_h(R,S) & \chi_h(R_h,S_h) \end{bmatrix} . \tag{7.1}$$

As in Ref. [43], $\phi_v(R,S)$ is the $A \times A$ matrix representing visible single-particle orbitals computed on the visible coordinates while the $A_h \times A_h$ matrix $\chi_h(R_h,S_h)$ yields the amplitudes of hidden orbitals evaluated on the coordinates of the A_h hidden nucleons. Finally, $\chi_h(R,S)$ and $\phi_v(R_h,S_h)$ are $A_h \times A$ and $A \times A_h$ matrices giving the amplitudes of hidden orbitals on visible coordinates and visible orbitals on hidden coordinates, respectively. All the above matrices are expressed in terms of deep neural networks with differentiable activation functions — see Ref. [43] for additional details. To respect the Pauli principle, the coordinates of the hidden nucleons must be permutation-invariant functions of the visible ones. We enforce this symmetry by using a Deep-Sets

architecture [12, 57] with *logsumexp* pooling. Additionally, the discrete parity and time reversal symmetries, are enforced in the same manner as Ref. [43].

Inspired by recent developments in quantum-chemistry NQS [58, 46, 47], we augment the flexibility of the ansatz by performing a generalized backflow transformation to the visible coordinates of the hidden-nucleon matrix: $(R, S) \to (\tilde{R}, \tilde{S})$. We use the Deep-Sets architecture again to enforce fermion anti-symmetry

$$(\tilde{\mathbf{r}}_i, \tilde{s}_i^z) = \rho_{\text{bf}}\Big(\mathbf{r}_i, s_i^z, \log\Big(\sum_j \exp(\phi_{\text{bf}}(\mathbf{r}_j, s_j^z)\Big)\Big)\,. \tag{7.2}$$

To further augment the expressivity, separate ρ_{bf} and ϕ_{bf} neural networks are used for each of the A visible orbitals.

We simulate infinite neutron matter using 14 particles in a box with periodic boundary conditions. Following Ref. [59], the latter are imposed by mapping the spatial coordinates onto periodic functions by

$$\boldsymbol{r}_i \to \left(\sin\left(\frac{2\pi \boldsymbol{r}_i}{L}\right), \cos\left(\frac{2\pi \boldsymbol{r}_i}{L}\right)\right) \tag{7.3}$$

which ensures the wave function is continuous and differentiable at the box boundary. Here L is the size of the simulation periodic box, and the sin and cos functions are applied element-wise to $\boldsymbol{r}_i$. Finite-size effects due to the tail corrections of two- and three-body potentials are accounted for by summing the contributions given by neighboring cells to the simulation box [60].

Evaluating the expectation values of quantum mechanical operators, including the Hamiltonian, requires carrying out multi-dimensional integration over the spatial and spin

coordinates of the neutrons. To this aim, we exploit Monte Carlo quadrature and sample R and S from $|\Psi_{HN}(R,S)|^2$ using the Metropolis-Hastings algorithm [61] — additional details can be found in the supplemental material of Ref. [41]. The best variational parameters defining the NQS are found by minimizing the system's energy, which we carry out using the R(oot)M(ean)S(quared)Prop(agation)-enhanced version of the stochastic-reconfiguration optimization method introduced in Ref. [43].

7.3 Results and Discussion

We first benchmark the expressivity of the hidden-nucleon NQS for periodic systems by comparing the energy per particle of infinite neutron matter against "conventional" variational Monte Carlo (VMC), and both constrained-path and AFDMC results. The variational wave function used in state-of-the-art neutron-matter studies, see for example [22, 36], contains a spin-independent Jastrow factor that multiplies a Slater determinant augmented by spin-dependent backflow correlations. The constrained-path approximation, commonly employed to alleviate the AFDMC fermion-sign problem [34], brings about a bias in the ground-state energy estimate [21, 36]. Exact results can be obtained by performing unconstrained propagations, but the statistical error grows exponentially with the imaginary time.

As shown in Fig. 7.1 for $\rho = 0.04$ fm^{-3}, after $\simeq 2000$ stochastic-reconfiguration steps, the NQS ansatz converges to the *virtually exact* unconstrained AFDMC energy, using a fraction of its computing time: about 100 hours on NVIDIA-A100 GPUs vs approximately 1.2 million hours on Intel-KNL CPUs. Notice that the constrained-path approximation violates the variational principle. In contrast, variational Monte Carlo calculations based on the NQS never yield energies below that of the Hamiltonian's ground state. Comparing

with the Hartree-Fock approximation, it appears that the hidden-nucleon ansatz captures the overwhelming majority of the correlation energy.

In Fig. 7.2, we compare the $\not\pi$EFT energies obtained with the NQS ansatz against AFDMC calculations of 14 particles with periodic-box boundary conditions, so that finite-size effects are the same in both approaches. The AFDMC takes as input the AV18+UIX Hamiltonian used in the celebrated APR equation of state [54], and the local, Δ-full chiral-EFT potentials that include tritium β-decay in the fitting procedure and do not make neutron-matter collapse, i.e. models NV2+3-Ia*/b*, and NV2+3-IIb* [21, 36]. Since for all the densities we consider, the AV18+UIX, NV2+3-Ia*/b*, and NV2+3-IIb* are in excellent agreement, they are collectively displayed by the "π-full" band, stressing that all these interactions explicitly retain pion-exchange terms.

The $\not\pi$EFT Hamiltonian "o" is in excellent agreement with the π-full models — both providing energies much below the non-interacting Fermi gas (not shown in the Figure). These minor differences are likely because model "o" yields a slightly larger nn scattering length than the experimental value and, therefore, more attraction in neutron matter. For benchmark purposes, we also consider the $\not\pi$EFT model "a" of Ref. [49], which provides a slightly stiffer equation of state than model "o". By checking the individual expectation value of the two and three-body potentials, we find that this behavior is primarily due to the three-body force contribution that is more repulsive in model "a" than model "o", which arises from a more bound ^{3}H when the two-body force alone is employed.

Once trained on the system's energy, the NQS can be used to accurately evaluate a variety of quantum-mechanical observables, such as the spin-singlet and triplet two-body distribution functions defined in Ref. [62]. Figure 7.3 shows these distributions at $\rho = 0.01$ fm^{-3} (panel a) and $\rho = 0.04$ fm^{-3} (panel b). The significant increase in the spin-singlet

channel compared to the non-interacting Fermi Gas indicates that the NQS wave function can capture the emergence of the $^1S_0$ neutron pairing, despite not being explicitly encoded in the ansatz. Consistent with the behavior of the pairing gap [37, 28], the enhancement is more prominent at $\rho = 0.01$ fm^{-3} than $\rho = 0.04$ fm^{-3}. On the other hand, at these densities, no pairing correlations are present in the spin-triplet channel.

7.4 Conclusion

In this work, we have put forward an NQS suitable to model the normal and superfluid phases of infinite neutron matter in a unified fashion. We improve the expressivity of the hidden-nucleon ansatz of Ref. [43] by adding state-dependent generalized backflow correlations, whose inclusion has proven beneficial in condensed-matter applications [46, 47]. Periodic-box boundary conditions are imposed by mapping the spatial coordinates of the neutrons onto periodic functions.

Combined with Monte Carlo techniques to sample the Hilbert space and the stochastic-reconfiguration algorithm to optimize the variational parameters, the NQS yields energies per particle of low-density neutron matter that are in excellent agreement with unconstrained AFDMC calculations at a fraction of the computational cost. In contrast, the computationally-inexpensive AFDMC constrained-path approximation brings about appreciable violations of the variational principle.

We have shown that $\not\pi$EFT yields a low-density neutron matter equation of state that is remarkably close to those obtained from AFDMC calculations that take as input highly-realistic phenomenological and chiral-EFT interactions [54, 21, 36]. This finding paves the way for more systematic comparisons between dilute neutron matter and Fermi gas around the unitary limit. In addition, it enables studies of phenomena relevant to understand the

inner crust and the outer core of neutron stars, such as pairing and superfluidity, using relatively simple models of nuclear dynamics.

Finally, we have analyzed the possible onset of Cooper pairing in the neutron medium. Specifically, the NQS two-body distribution functions corresponding to pairs of neutrons in the spin-singlet $^1S_0$ channel exhibit a clear enhancement at small inter-particle distances with respect to the non-interacting case, which is absent in the spin-triplet channel. Consistent with pairing-gap calculations [37, 28, 38], this behavior is more prominent at smaller densities. Note that this feature has not been encoded in the NQS; rather, it is a self-emerging quantum mechanical phenomenon.

As a future development, we plan on including more sophisticated interactions, including highly-realistic phenomenological ones, including AV18+UIX and the local, Δ-full chiral-EFT potentials of Ref. [63, 21, 36]. The flexibility of the NQS ansatz will also be tested in isospin-asymmetric nucleonic matter at low densities, where strong clustering is expected to occur [64].

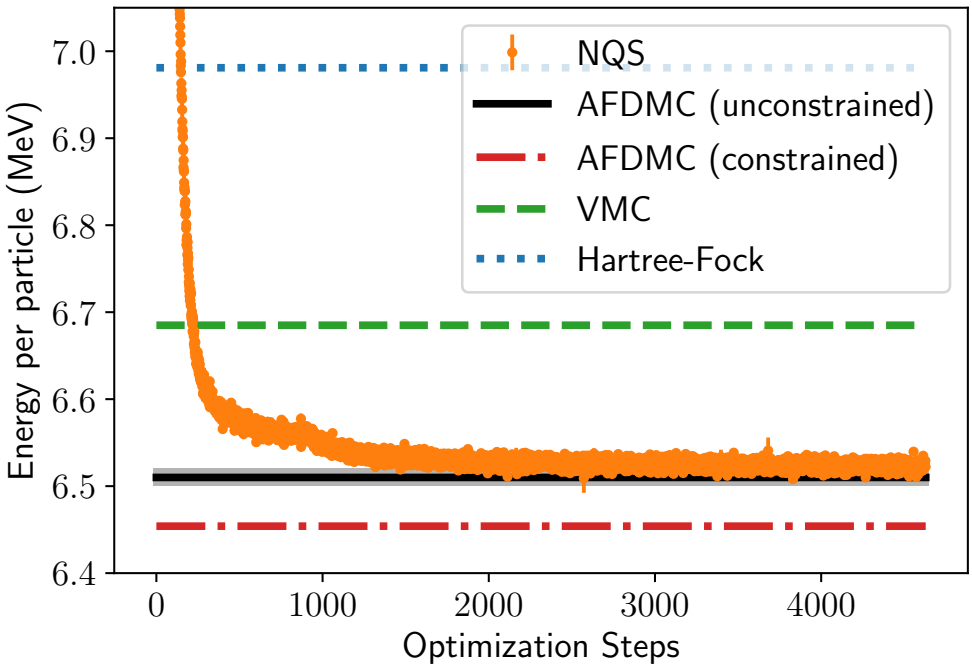

Figure 7.1: NQS training data in neutron matter at $\rho = 0.04$ fm^{-3} (data points) compared with Hartree-Fock (dotted line), conventional VMC (dashed line), constrained-path ADMC (dash-dotted line) and unconstrained-path ADMC results (solid line).

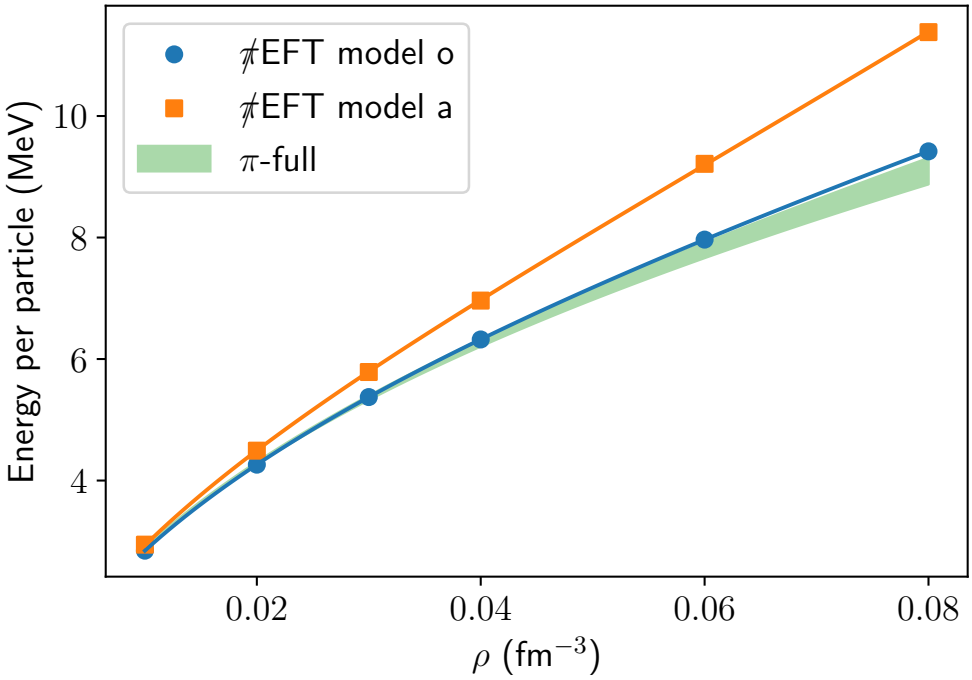

Figure 7.2: Low-density neutron-matter ≠EFT equation of state as obtained with the hidden-nucleon NQS for ≠EFT potential "o" (blue circles) and ≠EFT potential "a" (orange squares) compared with interactions which retain pion-exchange terms (green band). We see that the "o" potential is in excelent agreement with the π-full interactions while the "a" potential has a slightly stiffer equations of state due primarily to a more repulsive three-body force.

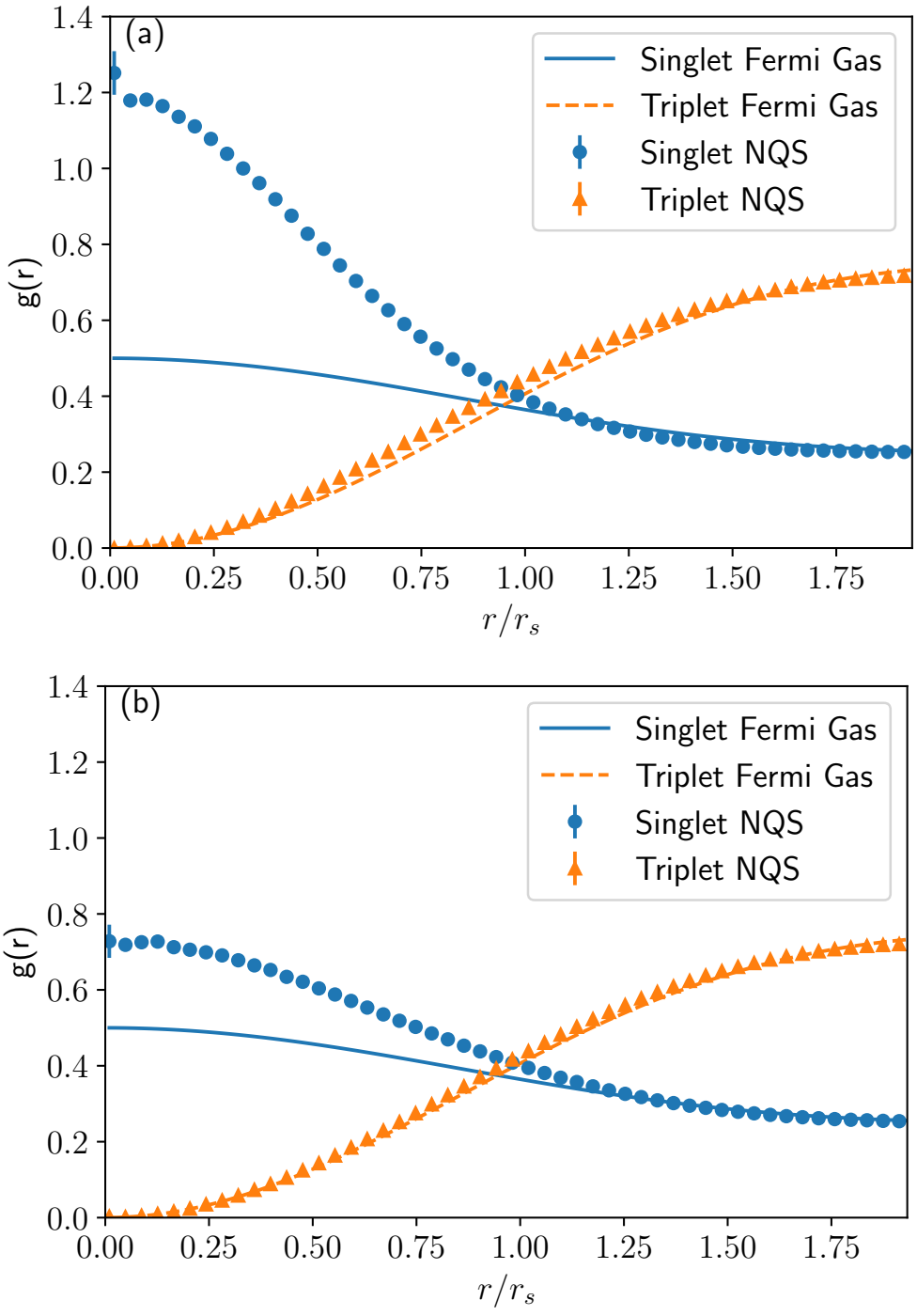

Figure 7.3: Spin-singlet and triplet two-body distribution functions at two different densities: $\rho = 0.01$ fm^{-3} (panel a) and $\rho = 0.04$ fm^{-3} (panel b). The NQS calculations (solid symbols) are compared with non-interacting Fermi Gas results (solid lines).

8 Homogeneous Electron Gas

The following article titled '**Message-Passing Neural Quantum States for the Homogeneous Electron Gas**' is currently under review [2].

We introduce a message-passing-neural-network-based wave function Ansatz to simulate extended, strongly interacting fermions in continuous space. Symmetry constraints, such as continuous translation symmetries, can be readily embedded in the model. We demonstrate its accuracy by simulating the ground state of the homogeneous electron gas in three spatial dimensions at different densities and system sizes. With orders of magnitude fewer parameters than state-of-the-art neural-network wave functions, we demonstrate better or comparable ground-state energies. Reducing the parameter complexity allows scaling to $N = 128$ electrons, previously inaccessible to neural-network wave functions in continuous space, enabling future work on finite-size extrapolations to the thermodynamic limit. We also show the Ansatz's capability of representing qualitative different phases of matter.

8.1 Introduction

Predicting emergent physical phenomena and system properties from the *ab-initio* description of the system's constituents is notoriously difficult [65, 66]. Fermionic systems, in particular, can exhibit strong correlations among the particles, leading to collective phenomena in the form of exotic phases of matter, e.g. superconductivity and superfluidity [67, 68]. In recent years, progress in numerical simulations of strongly correlated systems was triggered by the development of increasingly precise machine-learning approximation techniques. Most notably, artificial neural network (NN)

architectures, in combination with Variational Monte Carlo (VMC), have shown great promise to represent ground states of quantum spin systems, especially in more than one spatial dimension [1, 69, 70, 71, 72, 73, 74, 75]. Due to the universal approximation property of NNs, neural-network quantum states (NQS) can, in theory, accurately represent any quantum many-body state [76, 77]. NQS have been extended to fermionic degrees of freedom in a discrete basis [78, 79, 80], by incorporating the indistinguishability of quantum particles. More recently, advancements to ground and excited state searches for fermionic and bosonic continuous degrees of freedom with open [3, 81, 82] and periodic boundary conditions (PBCs) [83, 84, 85], have been introduced.

The flexibility of NQS, compared to more traditional models, allows representing multiple phases of matter and even different physical systems with a single Ansatz. To exemplify this point, we refer to NQS studies on the ground-state of molecular systems [3, 81], solutions to effective field theory Hamiltonians describing atomic nuclei [86, 87, 4], bulk studies of fermionic and bosonic extended systems [83, 88, 85, 89], as well as NQS simulations of low-density neutron matter found in neutron stars [15]. The downside of this flexibility, especially in continuous space, is that NQS typically need a significant amount of variational parameters to reach a given accuracy. This makes optimization challenging and costly, preventing the usage of refined optimization schemes, e.g. second order optimization procedures [90, 91]. As a result, the accessible system sizes are limited to a few tens of particles. However, studying larger system sizes is of utmost importance to estimate physical properties in the thermodynamic limit [92, 93, 94, 95]. To remedy the situation, novel NQS architectures must be developed that significantly reduce the parameter complexity while retaining high accuracy.

This work introduces a neural-network wave function suitable for simulating strongly

interacting fermionic quantum systems in continuous space with *one to two* orders of magnitude fewer parameters than current state-of-the-art NQS. The general form of the Ansatz is motivated by an analytical argument, relating the exact ground-state wave function to a many-body coordinate transformation of the electronic coordinates. It uses a permutation-equivariant message-passing architecture on a graph, inherently implementing the indistinguishability of same-species quantum particles [96]. As an application, we study the Homogeneous Electron Gas (HEG) in three spatial dimensions without explicitly breaking any of the fundamental symmetries of the system, such as translations and spin-inversion symmetry. This allows characterizing, from first principles, the onset of Wigner crystallization at low densities.

8.2 Exact Backflow Transformations

Throughout this work, we consider a non-relativistic Hamiltonian of identical particles with mass m in d spatial dimensions:

$$H = -\frac{\hbar^2}{2m}\sum_i^N \nabla_i^2 + V(\mathbf{X}), \tag{8.1}$$

where the potential and interaction energy, V, is assumed to be diagonal in position representation, defined by the particle coordinates $\mathbf{X} = (\boldsymbol{r}_1, ..., \boldsymbol{r}_N)$, $\boldsymbol{r}_i \in \mathbb{R}^d$. In the following, we derive an analytic functional form of the ground-state wave function and relate it to our variational Ansatz.

We use a suitable reference state $|\Phi_0\rangle$, as initial condition for the imaginary-time (τ) evolution induced by the Hamiltonian: $\Phi_\tau(\mathbf{X}) = \langle \mathbf{X}|e^{-\tau H}|\Phi_0\rangle$. The exact ground-state is obtained in the large imaginary-time limit: $\lim_{\tau\to\infty}\Phi_\tau(\mathbf{X}) \propto \Psi_0(\mathbf{X})$, provided $|\Phi_0\rangle$ is

non-orthogonal to the exact ground state, $|\langle\Psi_0|\Phi_0\rangle| > 0$. For fermions, non-orthogonality implies that the wave function must be antisymmetric w.r.t. the exchange of two particles i.e. $\Phi_0(\mathcal{P}_{ij}(\mathbf{X})) = -\Phi_0(\mathbf{X})$ ($\mathcal{P}_{ij}$ permutes particles i and j). Applying the mean-value theorem to the imaginary-time evolved state, yields:

$$\begin{aligned} \Phi_\tau(\mathbf{X}) &= \int_\Omega d\mathbf{X}' G_\tau(\mathbf{X},\mathbf{X}')\Phi_0(\mathbf{X}') && (8.2) \\ &= \mathrm{Vol}(\Omega) \times G_\tau\left(\mathbf{X}, \mathbf{Y}(\mathbf{X})\right) \Phi_0(\mathbf{Y}(\mathbf{X})), && (8.3) \end{aligned}$$

where Ω is the integration domain for the positional degrees of freedom, and $G_\tau(\mathbf{X},\mathbf{X}') = \langle\mathbf{X}|e^{-\tau H}|\mathbf{X}'\rangle$ is the matrix element of the imaginary-time propagator. In (8.3), we introduced the mean-value point $\mathbf{Y}(\mathbf{X}) = (\boldsymbol{y}_1(\mathbf{X}), ..., \boldsymbol{y}_N(\mathbf{X})) \in \Omega$, depending parametrically on the coordinates $\mathbf{X}$.

For non-relativistic Hamiltonians, Eq. (8.1), we have $G_\tau(\mathbf{X},\mathbf{X}') \geq 0$, for all $\mathbf{X},\mathbf{X}'$. Moreover, $G_\tau(\mathbf{X},\mathbf{X}')$ is invariant under the exchange of particle coordinates: $G_\tau(\mathcal{P}_{ij}(\mathbf{X}),\mathcal{P}_{ij}(\mathbf{X}')) = G_\tau(\mathbf{X},\mathbf{X}')$. In the fermionic case, the latter implies that $\mathbf{Y}(\mathbf{X})$ must be equivariant under particle exchange, $\mathbf{Y}(\mathcal{P}_{ij}(\mathbf{X})) = \mathcal{P}_{ij}(\mathbf{Y}(\mathbf{X}))$, to ensure antisymmetry of the total wave-function. Eq. (8.3) therefore yields the product between a permutation symmetric, positive semi-definite function $J(\mathbf{X}) = G_\tau(\mathbf{X},\mathbf{Y}(\mathbf{X})) \times \mathrm{Vol}(\Omega)$ and a reference state computed at modified coordinates $\mathbf{Y}(\mathbf{X})$:

$$\Phi_\tau(\mathbf{X}) = J(\mathbf{X}) \times \Phi_0(\mathbf{Y}(\mathbf{X})). \quad (8.4)$$

Identification of the mean-value point $Y(X)$ with a many-body coordinate transformation gives an alternative justification for the backflow transformations [97] of single-particle

coordinates. With a Slater determinant of given spin orbitals $\phi_\mu(\boldsymbol{r}_i)$ as initial state, $\Phi_0(\mathbf{X}) = \det \phi_\mu(\boldsymbol{r}_i)/\sqrt{N!}$, Eq. (8.4) is structurally related to the heuristic Jastrow-Backflow variational form [98, 99]. We remark that the symmetric contribution, $J(X)$, can be incorporated to the determinant:

$$\Phi_\tau(\mathbf{X}) = \mathcal{K} \times \det \varphi_\mu(\mathbf{y}_i(\mathbf{X})), \tag{8.5}$$

with $\varphi_\mu(\mathbf{y}_i(\mathbf{X})) = \phi_\mu(\mathbf{y}_i(\mathbf{X})) \times \sqrt[N]{J(\mathbf{X})}$, and $\mathcal{K}$ a normalization constant.

The functional form (8.4),(8.5) is exact, provided that the symmetric factor $J(\mathbf{X})$ and the mean-value coordinates $\mathbf{Y}(\mathbf{X})$ satisfy Eq. ((8.3)), and the reference state is not orthogonal to the exact ground state. An approximate but explicit form for the coordinate transformation $\mathbf{Y}(\mathbf{X})$ can be obtained by repeatedly applying the imaginary-time propagator to the reference state in the limit of small τ. This process gives rise to the iterative backflow transformation, as introduced in Ref. [100, 101].

8.3 Message-Passing Neural-Network Quantum States

Motivated by Eq. (8.5), we use single-particle orbitals, $\{\phi_\mu\}_{\mu=1}^N$, evaluated at many-body backflow coordinates, $\mathbf{Y}(\mathbf{X})$, to construct the variational Ansatz. The backflow transformation is parameterized with permutation-equivariant message-passing NNs (MPNN) [96], hence we name it Message-Passing Neural Quantum State (MP-NQS).

In the MPNN, an all-to-all connected graph, encoding effective particle positions (nodes) $\mathbf{x}_i^{(t)} = [\mathbf{x}_i^{(0)}, \mathbf{h}_i^{(t)}] \in \mathbb{R}^{D1}$ and their interactions (edges) $\mathbf{x}_{ij}^{(t)} = [\mathbf{x}_{ij}^{(0)}, \mathbf{h}_{ij}^{(t)}] \in \mathbb{R}^{D2}$, is updated iteratively (we have introduced a time-step $t \geq 0$, an initial graph ($\mathbf{x}_i^{(0)}$, $\mathbf{x}_{ij}^{(0)}$), successively updated auxiliary variables known as hidden states ($\mathbf{h}_i^{(t)} \in \mathbb{R}^{D1^h}$, $\mathbf{h}_{ij}^{(t)} \in \mathbb{R}^{D2^h}$),

suitably chosen *feature* dimensions $D_1^{(h)}$ and $D_2^{(h)}$, and we denoted concatenation with $[\cdot,\cdot]$). Construction of the initial graph $(\mathbf{x}_i^{(0)},\ \mathbf{x}_{ij}^{(0)})$ is system dependent and will be discussed in detail later.

The hidden states are updated using equivariant *messages*, obtained from an attention mechanism [102]. The messages are given by weighted transformations of the edges $\mathbf{x}_{ij}^{(t)}$: $\mathbf{m}_{ij}^{(t+1)} = \boldsymbol{\omega}_{ij}^{(t)}(\mathbf{x}_{ij}^{(t)}) \odot \boldsymbol{\phi}(\mathbf{x}_{ij}^{(t)})$, where $\odot$ represents element-wise multiplication along the feature dimension, and $\boldsymbol{\omega}_{ij}^{(t)} \in \mathbb{R}^{D2}$ are weight vectors. The weights are obtained using *query/key* matrices given by $\mathbf{Q}_{ij}^{(t)} = W_Q^{(t)} \cdot \mathbf{x}_{ij}^{(t)}$ and $\mathbf{K}_{ij}^{(t)} = W_K^{(t)} \cdot \mathbf{x}_{ij}^{(t)}$, with weight matrices $W_Q^{(t)}, W_K^{(t)} \in \mathbb{R}^{D2\times D2}$. Applying an element-wise GELU non-linearity [103] to the overlap between queries and keys along the *particle dimension* (as opposed to the feature dimension [102, 104]), results in permutation-equivariant weights

$$\boldsymbol{\omega}_{ij}^{(t)} = \mathrm{GELU}\left(\sum_l \mathbf{Q}_{il}^{(t)}\mathbf{K}_{lj}^{(t)}\right). \tag{8.6}$$

This attention mechanism compares environments of particles i and j, and effectively increases the order of correlations that can be embedded in a single iteration of the network. This is crucial to reduce the total number of network iterations (parameters) and capture many-body effects. The hidden states are updated using the current graph and messages:

$$\mathbf{h}_i^{(t+1)} = \boldsymbol{f}\left(\mathbf{x}_i^{(t)}, \sum_{j\neq i} \mathbf{m}_{ij}^{(t+1)}\right) \tag{8.7}$$

$$\mathbf{h}_{ij}^{(t+1)} = \boldsymbol{g}\left(\mathbf{x}_{ij}^{(t)}, \mathbf{m}_{ij}^{(t+1)}\right) \tag{8.8}$$

where the functions $\boldsymbol{\phi}$, $\boldsymbol{f}$, and $\boldsymbol{g}$ are parameterized by Multilayer Perceptrons (MLPs).

The updated graph has then the same strucutre as the former: $\mathbf{x}_i^{(t+1)} = [\mathbf{x}_i^{(0)}, \mathbf{h}_i^{(t+1)}]$, $\mathbf{x}_{ij}^{(t+1)} = [\mathbf{x}_{ij}^{(0)}, \mathbf{h}_{ij}^{(t+1)}]$. Inclusion of the initial inputs, referred to as a "skip connection" in ML literature [105], mitigates the vanishing gradient problem and allows a more efficient capture of correlations.

The final backflow coordinates are constructed as $\boldsymbol{y}_i(\mathbf{X}) = \boldsymbol{r}_i + \delta\boldsymbol{r}_i(\mathbf{X})$, where the displacements, $\delta\boldsymbol{r}_i(\mathbf{X})$, are obtained via a linear transformation of the final node states to d dimensions, i.e. $\delta\boldsymbol{r}_i(\mathbf{X}) = W \cdot \mathbf{x}_i^{(T)}$ with $W \in \mathbb{C}^{d \times D_1}$. The complex-valued backflow transformation allows changing the degree of localization, determined by the chosen single-particle orbitals, and representing complex-valued wave functions in general.

Following (8.5), we further augment the orbitals with a permutation-invariant factor J, yielding:

$$\Psi(\mathbf{X}) = \det \varphi_\mu(\mathbf{y}_i(\mathbf{X})), \tag{8.9}$$

with $\varphi_\mu(\boldsymbol{y}_i) = \exp[J(\mathbf{Y}, \mu)] \times \phi_\mu(\boldsymbol{y}_i)$.

8.4 Hamiltonian

We now study the case of the homogeneous electron gas (HEG) in $d = 3$ spatial dimensions, a prototypical model for the electronic structure in solids. It includes Coulomb interactions among the solids' electrons while treating its positively charged ions as uniform, static, positive background [106]. Despite this simplification, the HEG exhibits different phases of matter and captures properties of real solids, particularly of Alkali metals. The Hamiltonian

(in units of Hartree), for a system of N electrons with uniform density $n = \frac{N}{V}$, is given by:

$$H = -\frac{1}{2r_s^2}\sum_{i}^{N}\nabla_i^2 + \frac{1}{r_s}\sum_{i<j}^{N}\frac{1}{\|\boldsymbol{r}_i - \boldsymbol{r}_j\|} + \text{const.} \tag{8.10}$$

where we introduced the Wigner-Seitz radius $r_s = \sqrt[3]{3/(4\pi n)}$, and a constant arising from the electron-background interaction [98]. The conditionally convergent series of pairwise Coulomb interactions is evaluated using the Ewald summation technique, as is standard for extended systems in QMC [107, 108, 109]. We'll assume a fixed spin-polarization with $N = N_\uparrow + N_\downarrow$, where $N_{\uparrow/\downarrow}$ denotes the number of up/down spins. $s_i \in \{\uparrow, \downarrow\}$ denotes the i-th electron's spin. We equip the cubic simulation cell of side length L with periodic boundary conditions (PBCs) in all spatial directions to access the bulk of the system.

As in (8.5), we use a single Slater determinant as a reference state, $\Phi_0(\mathbf{X})$. For the translation-invariant HEG, plane-wave orbitals are a natural and physically-motivated choice:

$$\phi_{\boldsymbol{k}}(\boldsymbol{r}) = \exp\left(i\boldsymbol{k}\cdot\boldsymbol{r}\right) \tag{8.11}$$

with $\boldsymbol{k} = \frac{2\pi}{L}\boldsymbol{n}$, where $\boldsymbol{n} \in \mathbb{Z}^d$. To account for the spin s of a particle located at $\mathbf{r}$, we use spin-orbitals $\phi_\mu(\boldsymbol{r}, s) = \phi_{\boldsymbol{k}_\mu}(\boldsymbol{r})\delta_{s_\mu,s}$, where each spin-orbital is characterized by the quantum numbers $\mu = (\boldsymbol{k}_\mu, s_\mu)$. These orbitals allow modeling of translation invariant systems at fixed total momentum $\boldsymbol{k}_{\text{tot}} = \sum_{i=1}^{N}\boldsymbol{k}_i$. Furthermore, the determinant factorizes into a product of determinants of up and down spin orbitals.

We further specialize the MP-NQS to the HEG by defining initial feature vectors. Respecting the spin inversion and translation symmetries of the HEG requires us to ignore single-particle positions and spins. We, therefore, initialize the nodes to a learnable

embedding vector $\mathbf{e} \in \mathbb{R}^{D_1}$, that does not depend on the particle index i. For the edge features, we use the translation invariant particle-distances $\boldsymbol{r}_{ij} = \boldsymbol{r}_i - \boldsymbol{r}_j$ and their norm. Same- and opposite-spin pairs are distinguished using products of the form $s_i \cdot s_j = \pm 1$ in the edge features. Overall, we obtain the following initial features:

$$\mathbf{x}_i^{(0)} = \mathbf{e}, \qquad \mathbf{x}_{ij}^{(0)} = \left(\mathbf{r}_{ij}, \|\mathbf{r}_{ij}\|, s_i \cdot s_j\right). \tag{8.12}$$

Notice that this choice preserves the spin quantum number of each particle.

The PBCs of the simulation box are incorporated by mapping the components of vectors $\boldsymbol{r} \in \mathbb{R}^d$ (where $\boldsymbol{r} = \boldsymbol{r}_i$ or $\boldsymbol{r} = \boldsymbol{r}_{ij}$) to a Fourier basis $\boldsymbol{r} \mapsto \left[\sin(\frac{2\pi}{L}\boldsymbol{r}), \cos(\frac{2\pi}{L}\boldsymbol{r})\right] \in \mathbb{R}^{2d}$, and the norm of the distance between two particles, $\|\boldsymbol{r}_{ij}\|$, to a periodic surrogate $\|\boldsymbol{r}_{ij}\| \mapsto \|\sin(\frac{\pi}{L}\boldsymbol{r}_{ij})\|$, as in Ref. [83]. In sum, our Ansatz is translation- and spin-inversion- invariant and possesses fixed total momentum, $\boldsymbol{k}_{\text{tot}}$. Its number of parameters is system-size independent (here $\sim$ 19000) and, using Stochastic Reconfiguration (SR) [91], only $\mathcal{O}(10^3)$ optimization steps are needed to reach convergence. A comparison to other NQS approaches is given in the Supplemental Material.

8.5 Results

We study the fully spin-polarized and unpolarized HEG in different density regimes $r_s \in [1, 200]$ and up to system sizes of $N = 128$ electrons. We benchmark our ground-state energies against a variety of methods, including transcorrelation augmented full configuration interaction method (FCI), distinguishable clusters with doubles (DCD) method [94] for large densities, and *Diffusion Monte Carlo* (DMC) with backflow (BF-DMC) [110, 111] for small densities. We also compare to state-of-the-art NQS

architectures – FermiNet [85] and WAPNet [84] – available for small system sizes $N \in \{14, 19\}$. The effect of our backflow transformation on the nodal surface is studied by comparing it to fixed-node DMC (FN-DMC) results (see Supplemental Material). We use an energy of 1.5 mHa per particle (chemical accuracy) to assess the significance of energy differences between the different methods. An overview of our results and benchmarks for the various system sizes and densities is provided in the Supplemental Material.

8.5.1 Energy Benchmarks for Small Systems

For $N = 14$, FCI provides the lowest available ground-state energy for the HEG for $r_s \leq 5$. The energy difference between the MP-NQS and FCI results is lower than 1.5 mHa per particle. State-of-the-art NQS architectures perform comparably to the MP-NQS: The unrestricted FermiNet performs slightly better ($\mathcal{O}\left(10^{-5}\right)$ Ha/N) than both MP-NQS and WAPNet for $r_s \leq 2$, while the MP-NQS and WAPNet improve over this version of FermiNet for $r_s = 5$. The restricted FermiNet yields worse ground-state energies than the MP-NQS over all probed densities [85] (see Fig. 8.1). For $r_s \geq 5$ we compare to results obtained with WAPNet and DCD. We find slightly better performance than WAPNet for all of the reported densities. Furthermore, both neural-network-based methods outperform the DCD method. Nevertheless, all results lie within a range of 1.5 mHa per particle. A similar pattern is seen for $N = 19$: the MP-NQS obtains slightly higher energies than WAPNet for large densities ($r_s \leq 5$) and marginally lower ones at smaller densities ($r_s \geq 5$). The differences are lower than 1.5 mHa per particle.

8.5.2 Energy Benchmarks for Large Systems

For $N = 54$ particles, we compare to FCI at high density $r_s = 1$, where our method achieves the same accuracy. The difference between our results and FCI decreases with

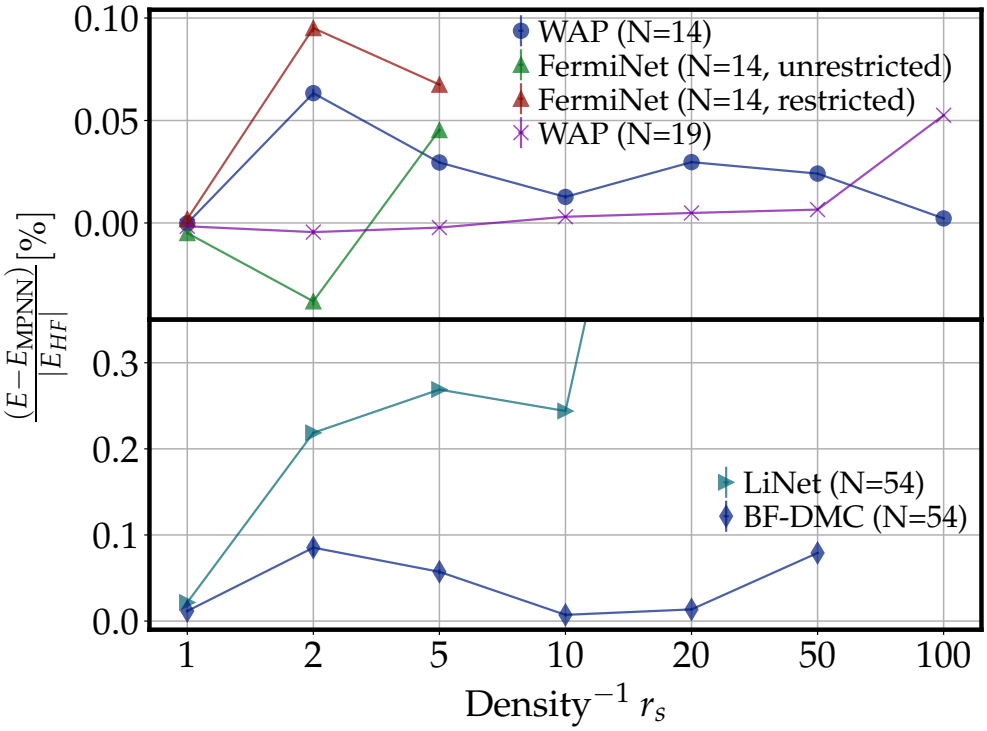

Figure 8.1: Energy differences between ground-state energies, obtained with other methods and with the MP-NQS, in units of the thermodynamic Hartree Fock energy, for various densities, polarizations, and system sizes. (Top) $N = 14, 19$, (Bottom): $N = 54$ particles. Error bars are too small to be visible for most densities. The corresponding numerical data can be found in the Supplemental Material.

increasing system size, yielding statistically indistinguishable results for $N = 54$. We obtain significantly better ground-state energies than BF-DMC, especially at high densities. This is in stark contrast to the (FermiNet-based) architecture of [88] (dubbed LiNet in the following), which does not improve upon BF-DMC energies over the whole density regime (see Figure 8.1, bottom). Comparison to DCD data shows that the MP-NQS can consistently improve the ground-state energies for all examined densities, with energy differences exceeding 1.5 mHa per particle. This indicates a deterioration of DCD's accuracy with increasing system size, while our method appears to maintain its precision independent of system size. As expected, we reach higher accuracy compared to FN-DMC due to the optimization of the nodal surface. The difference is ~ 1 mHa per particle at the

largest density. Still, it decreases with decreasing density because the nodal surface does not contribute as much to the ground-state energy in this regime.

A liquid-crystal phase transition is expected for the HEG, as a function of the density n. The dominating kinetic energy in Eq. (8.10) ($\sim 1/r_s^2$) for large n leads to the well-known Fermi liquid behavior. For small n the potential energy ($\sim 1/r_s$) dominates and enforces a crystalline structure among the electrons, known as Wigner crystal. The Wigner crystal, expected to be of BCC type [112], is translation invariant, and the resulting crystal structure is called a *floating* crystal. A homogeneous single-particle density distribution and a crystalline two-body radial distribution function characterize the latter. Detection of the translation invariant transition has not been realized up to now, to the best of our knowledge. In previous QMC simulations of the Wigner crystal phase (e.g., in the works of Refs. [113, 95, 85]), the transition is investigated by comparison of energies obtained with a crystalline and liquid variational Ansatz, explicitly breaking translation symmetry. Consequently, these works find a pinned BCC lattice. Crystallization is further favored by Gaussian orbitals centered around BCC lattice sites, a primitive BCC simulation cell, or both. We use the MP-NQS as unbiased, translation invariant variational Ansatz to investigate the transition in a conventional BCC cell, i.e., a simple cubic simulation cell. We study a system size of $N = 128$ particles at $r_s = 110, 200$. We display the radial distribution functions for different densities in Figure 8.2. We compare results from a liquid FN-DMC and BF-DMC calculation to results obtained with the MP-NQS at the predicted transition density around $r_s = 110$ [113]. We observe remarkable agreement between the three correlators, implying that the MP-NQS favors the fluid behavior at this density. For the lower density of $r_s = 200$, we observe increasing oscillations up to larger distances, departing from the liquid FN-DMC result. This shows the capability of the MP-NQS to describe qualitatively different phases.

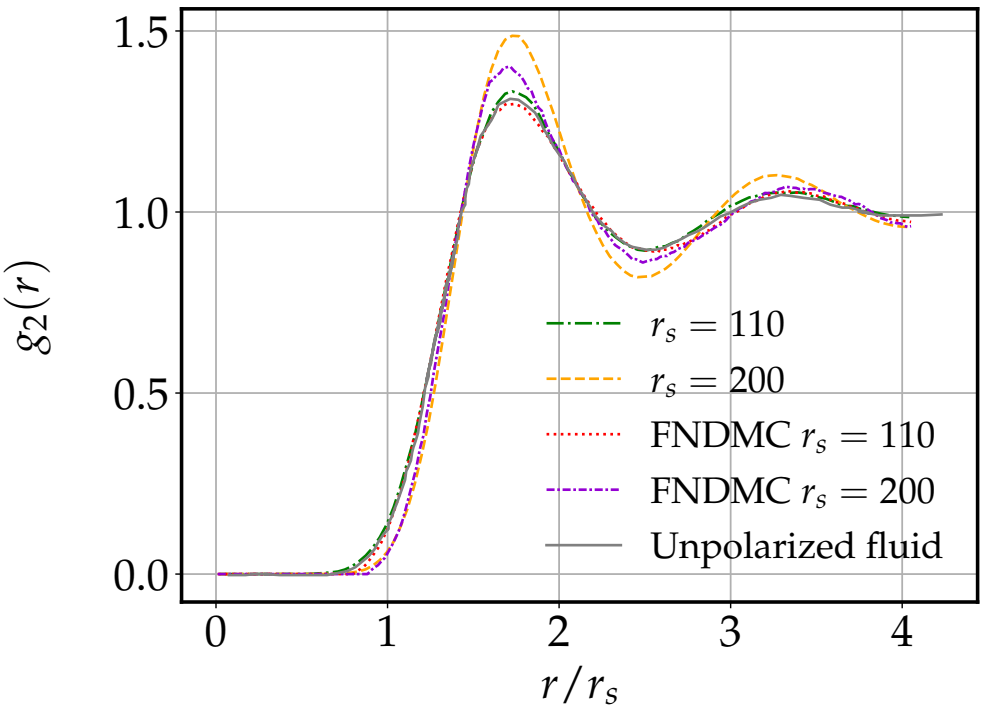

Figure 8.2: Spin-averaged radial distribution function for the homogeneous electron gas with $N = 128$ electrons at low densities ($r_s = 110, 200$). Error bars are smaller than the symbols. The unpolarized fluid is obtained from [113] for $r_s = 110$.

8.6 Conclusions

We have introduced MP-NQS, a novel NQS architecture, leveraging MPNNs to build highly expressive backflow coordinates. We demonstrate its power on the HEG system, reducing the number of parameters by orders of magnitudes compared to state-of-the-art NQS in continuous space while reaching at par or better accuracy. We also show improvement upon state-of-the-art BF-DMC results on large systems. The favorable scaling allows us to accurately simulate large periodic electronic systems, previously inaccessible to state-of-the-art NQS models. We increase the available system sizes from $N = 27$ and $N = 54$ electrons in periodic systems [88, 85, 84] to $N = 128$ electrons in this work. We hence open the door to extrapolation methods to the thermodynamic limit for extended systems.

N	r_s	MP-NQS	WAP [84]	FermiNet [85]	FCI*/DCD** [94]
14	1	0.568967(6)	0.568965(1)	0.568904(1)	0.56861(1)*
	2	−0.008391(1)	−0.0083310(3)	−0.008427(1)	−0.00868(2)*
	5	−0.0798544(4)	−0.0798360(1)	−0.079821(1)	−0.08002(2)*
	10	−0.0552126(6)	−0.05520380(3)	N/A	−0.05509**
	20	−0.0324553(2)	−0.0324434(1)	N/A	−0.03201**
	50	−0.01462631(6)	−0.01462211(4)	N/A	−0.01384**
	100	−0.00773018(3)	−0.007729980(2)	N/A	N/A

Table 8.1: Total energy per particle in Hartree for unpolarized system of $N = 14$ particles. WAPNet and FermiNet are alternative NQS architectures optimized via VMC. We include FCI and DCD results as benchmarks from quantum chemistry.

N	r_s	MP-NQS	LiNet [88]	FN-DMC	BF-DMC [110, 111]
54	1	0.52973(1)	0.530019(1)	0.53094(2)	0.52989(4)
	2	−0.014046(8)	−0.013840(1)	−0.01326(2)	−0.013966(2)
	5	−0.079090(2)	−0.0788354(2)	−0.07867(1)	−0.079036(3)
	10	−0.054448(1)	−0.0542785(1)	−0.054269(8)	−0.054443(2)
	20	−0.0320524(5)	−0.0316886(1)	−0.031976(8)	−0.032047(2)
	50	−0.0145015(1)	N/A	−0.01387(2)	−0.0144877(1)
	100	−0.0076793(1)	N/A	−0.007674(3)	N/A

Table 8.2: Total energy per particle in Hartree for the unpolarized system of $N = 54$ particles.

The simulation of the HEG phase transition using a translation invariant Ansatz in an unbiased simulation cell shows the MP-NQS' capability to represent different phases of matter. We reproduce the liquid phase of the HEG up to around $r_s = 110$, while observing pronounced density fluctuations, potentially compatible with Wigner crystallization, at around $r_s = 200$. We leave it to further, more specialized, investigation to study the nature of the floating phase found at small densities, potentially resolving the tension with existing predictions [113, 95] (based on translation-symmetry breaking wave function states), on the location and nature of the phase transition.

In addition to the numerical results, we introduced an analytical argument, justifying commonly adopted backflow transformations. Our argument shows that a backflow transformation over a reference state is sufficient to obtain the exact ground-state wave

function. It will be of particular interest to characterize the geometrical properties of these transformations and understand in what cases neural-network parameterizations can efficiently describe them.

9 Ultra-cold Fermi Gases

The following article titled '**Neural-network quantum states for ultra-cold Fermi gases**' is currently under review [5].

Ultra-cold Fermi gases exhibit a rich array of quantum mechanical properties, including the transition from a fermionic superfluid BCS state to a bosonic superfluid BEC state, which can be precisely probed experimentally. However, accurately describing these properties poses significant theoretical challenges due to strong pairing correlations and the non-perturbative nature of particle interactions. Here, we introduce a Pfaffian-Jastrow neural-network quantum state with a message-passing architecture to encode pairing and other quantum correlations efficiently. Our novel approach surpasses existing Slater-Jastrow frameworks and outperforms state-of-the-art diffusion Monte Carlo methods, as evidenced by lower ground-state energies. We observe the emergence of strong pairing correlations by analyzing the opposite-spin pair distribution functions. Additionally, we demonstrate that transfer learning enhances the training of neural-network wave functions, facilitating the exploration of the BCS-BEC crossover region near unitarity. Our findings highlight the potential of neural-network quantum states as a promising strategy for investigating ultra-cold Fermi gases.

9.1 Introduction

The study of ultra-cold Fermi gases has received considerable experimental and theoretical attention in recent years due to their unique properties and potential applications in fields ranging from condensed matter physics to astrophysics. These systems can be created and

manipulated in the laboratory with high precision, providing a versatile platform for investigating a wide variety of phenomena. By tuning the s-wave scattering length a via external magnetic fields near a Feshbach resonance, one can smoothly crossover from a fermionic superfluid BCS state ($a < 0$) of long-range Cooper pairs to a bosonic superfluid BEC state ($a > 0$) of tightly-bound, repulsive dimers. Given their diluteness, the behavior of these systems is mainly governed by a and the effective range of the potential r_e, with natural units provided by the Fermi momentum k_F and the Fermi gas energy per particle in the thermodynamic limit $E_{FG} = \frac{3}{5}\frac{\hbar^2}{2m}k_F^2$ (see Ref. [114] and references therein).

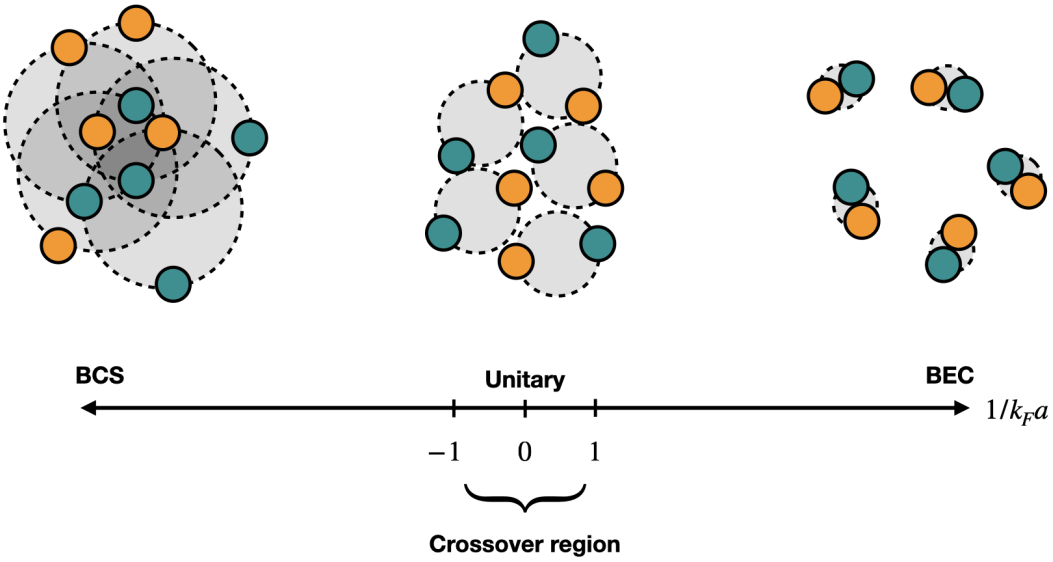

Figure 9.1: A cartoon of the BCS-BEC crossover. Moving from left to right, the attractive interaction between opposite-spin fermions increases. However, in the BEC regime, the attraction binds the pairs so tightly that they behave as weakly repulsive bosons. The region between the weakly attractive Cooper pairs and the weakly repulsive dimers is known as the unitary limit.

The region between the BCS and BEC states, known as the "unitary limit," is particularly interesting as a diverges and r_e approaches zero. The unitary Fermi gas (UFG) is a strongly-

interacting system that exhibits surprisingly stable superfluid behavior. Studying the BCS-BEC crossover near the unitary limit can reveal critical aspects of the underlying mechanism behind superfluidity in fermionic matter. The UFG is also universal, meaning its properties are independent of the details of the two-body potential. This universality allows for robust comparisons and predictions between seemingly disparate quantum systems. For instance, the UFG is relevant for neutron stars, as they provide a means to study superfluid low-density neutron matter [115, 116], whose properties are crucial for the phenomenology of glitches [32] and the cooling of these stars via neutrino emission [30, 31, 117].

The onset of strong pairing correlations and the non-perturbative nature of the interaction makes the theoretical study of these systems particularly challenging for quantum many-body methods. Among them, quantum Monte Carlo (QMC) has proven to be exceptionally efficient in calculating various properties with high accuracy, including the energy [118], pairing gap [119], and other quantities related to the so-called contact parameter [120]. Diffusion Monte Carlo (DMC), in particular, is an accurate tool for calculating the properties of quantum many-body systems [121]. The fixed-node approximation typically employed in DMC calculations to control the fermion-sign problem provides a rigorous upper bound to the ground-state energy that agrees well with other methods and experiments [122, 123]. Moreover, unlike the Auxiliary Field Quantum Monte Carlo method, DMC can handle broad classes of local interactions, which provides exact results that are sign-problem free but is limited only to unpolarized systems with a purely attractive interaction [118]. However, the fixed-node approximation limits the accuracy of DMC energies, which induces a residual dependence on the starting variational wave function. The latter has a critical role in DMC calculations of expectation values of operators that do not commute with the Hamiltonian, such as spatial and momentum distributions. The analytical form of the variational ansatz

is usually tailored to specific problems of interest and biased by the physical intuition of the researchers.

In this work, we overcome these limitations by performing variational Monte Carlo (VMC) calculations of ultra-cold Fermi gases with neural-network quantum states (NQS) [1] that incorporate only the most essential symmetries and boundary conditions. After their initial application to quantum-chemistry problems [46, 124], continuous-space NQS have been successfully employed to study quantum many-body systems in the presence of spatial periodicities, such as interacting quantum gases of bosons [125], the homogeneous electron gas [126, 127], and dilute neutron matter [15]. Recent works have also used NQS to solve the nuclear Schrödinger equation in both real space[40, 128, 129, 130, 131] and the occupation number formalism [132]. When dealing with fermions, the antisymmetry is usually enforced using generalized Slater determinants, the expressivity of which can be augmented with either backflow transformations [133] or by adding "hidden" degrees of freedom [56].

Strong pairing correlations in fermionic systems motivate adopting an antisymmetrized wave function constructed from pairing orbitals rather than single-particle orbitals. One such construction, often called the geminal wave function [134, 135], considers determinants of spin-singlet pairs, while other more general wave functions based on the Pfaffian [136, 6, 7, 137], consider both singlet and triplet contributions. Pfaffian wave functions combined with neural-network Jastrow correlators [138] have successfully modeled lattice fermions, even revealing the existence of a quantum spin liquid phase in the J1-J2 models on two-dimensional lattices [139].

We propose a novel NQS that extends the conventional Pfaffian-Jastrow [6] ansatz by incorporating neural backflow transformations into a fully trainable pairing orbital. The

backflow transformations are generated by a message-passing architecture recently introduced to model the homogeneous electron gas [2]. In addition to being a significant departure from generalized Slater determinants, our Pfaffian-Jastrow NQS naturally encodes pairing in the singlet and triplet channels, without stipulating a particular form for the pairing orbital. In view of this, it is broadly applicable to other strongly-interacting systems with the same symmetries and boundary conditions. We demonstrate the representative power of our NQS by computing ground-state properties of ultra-cold Fermi gases in the BCS-BEC crossover. Our Pfaffian-Jastrow NQS outperforms Slater-Jastrow NQS by a large margin, even when generalized backflow transformations are included in the latter. Most notably, we find lower energies than those obtained with state-of-the-art DMC methods, which start from highly-accurate BCS-like trial wave functions.

The rest of the paper is organized as follows. In Section **??**, we introduce the Hamiltonian used to model ultra-cold atomic gases near the unitary limit and the many-body techniques used to solve the Schrödinger equation. In Section 9.6, we compare the Pfaffian-Jastrow NQS with other NQS ansätze and state-of-the-art DMC results. Finally, in Section 9.7, we draw our conclusion and provide future perspectives of this work.

9.2 Hamiltonian

As customary in QMC approaches, we simulate the infinite system using a finite number of fermions N in a cubic simulation cell with side length L, equipped with periodic boundary conditions (PBCs) in all $d = 3$ spatial dimensions. We use $\boldsymbol{r}_i \in \mathbb{R}^d$ and $s_i \in \{\uparrow, \downarrow\}$ to denote the positions and spin projections on the z-axis of the i-th particle, and the length L can be determined from the uniform density of the system $N/L^3 = k_F^3/(3\pi^2)$. The dynamics of the

unpolarized gas is governed by the non-relativistic Hamiltonian

$$H = -\frac{\hbar^2}{2m}\sum_i^N \nabla_i^2 + \sum_{ij}^N v_{ij}\,, \tag{9.1}$$

where the attractive two-body interaction

$$v_{ij} = (\delta_{s_i,s_j} - 1)v_0 \frac{2\hbar^2}{m}\frac{\mu^2}{\cosh^2(\mu r_{ij})}\,, \tag{9.2}$$

acts only between opposite-spin pairs, making the interaction mainly in s-wave for small values of r_e. In the above equations, ∇_i^2 is the Laplacian with respect to $\boldsymbol{r}_i$ and $r_{ij} = \|\boldsymbol{r}_i - \boldsymbol{r}_j\|$ is the Euclidean distance between particles i and j. The Pöschl-Teller interaction potential of Eq. (9.2) provides an analytic solution of the two-body problem and has been employed in several previous QMC calculations [140, 141, 142, 120]. The parameters v_0 and μ tune the scattering length a and effective range r_e, respectively. In the unitary limit $|a| \to \infty$, the zero-energy ground state between two particles corresponds to $v_0 = 1$ and $r_e = 2/\mu$. In order to analyze the crossover between the BCS and BEC phases, we will use different combinations of v_0 and μ that correspond to the same effective range. In addition, we will consider various values of μ with fixed $v_0 = 1$ to extrapolate the zero effective range behavior at unitarity.

9.3 Neural-Network Quantum States

We solve the Schrödinger equation associated with the Hamiltonian of Eq. (9.1) using two different families of NQS. All ansätze have the general form

$$\Psi(X) = e^{J(X)}\Phi(X), \tag{9.3}$$

where the Jastrow correlator $J(X)$ is symmetric under particle exchange and $\Phi(X)$ is antisymmetric. Here, we have introduced $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, with $\boldsymbol{x}_i = (\boldsymbol{r}_i, s_i)$, to represent the set of all single-particle positions and spins compactly.

In addition to the antisymmetry of fermionic wave functions, the periodic boundary conditions, and the translational symmetry (which will be discussed in Sec. **??**), we also enforce the discrete parity and time-reversal symmetries as prescribed in Ref. [130]. More specifically, we carry out the VMC calculations for the unpolarized gas using $\Psi^{PT}(R, S)$ given by

$$\Psi^{P}(R, S) = \Psi(R, S) + \Psi(-R, S), \tag{9.4}$$

$$\Psi^{PT}(R, S) = \Psi^{P}(R, S) + (-1)^{n}\Psi^{P}(R, -S), \tag{9.5}$$

where $n = N/2$ and we have used the notation $R = \{\boldsymbol{r}_1, \ldots, \boldsymbol{r}_N\}$ and $S = \{s_1, \ldots, s_N\}$ for the set of all positions and spins, respectively. Enforcing these symmetries has been shown to accelerate the convergence of ground-state energies for both atomic nuclei [130] and dilute neutron matter [15].

9.3.1 Pfaffian-Jastrow-Backflow

The antisymmetric part of the wave function employed in QMC studies of ultra-cold Fermi gases is typically constructed as an antisymmetrized product of BCS spin-singlet pairs [143, 140, 115, 141, 144]. It goes by a variety of names, such as the geminal wave function [134, 135], the singlet pairing wave function [6], and the (number-projected) BCS wave function [144], just to name a few. Although geminal wave functions have demonstrated significant improvements over single-determinant wave functions of

single-particle orbitals, the energy gains are typically smaller for partially spin-polarized systems [135], as contributions from the spin-triplet channel are missing. This naturally leads to the singlet-triplet-unpaired (STU) Pfaffian wave function [136, 6], in which the pairing orbitals are explicitly decomposed into singlet and triplet channels. Then, the STU ansatz is expressed as the Pfaffian of a block matrix, with the singlet, triplet, and unpaired contributions partitioned into separate blocks. When the triplet blocks are zero, the STU wave function reduces to the geminal wave function.

Both the geminal and the STU wave functions rely on fixing the spin ordering of the interacting fermions. Consequently, they are not amenable to potentials that exchange spin, such as those used to model the interaction among nucleons [145]. In neutron-matter calculations, for instance, the pairing orbital for the Pfaffian wave function can be taken as a product of a radial part and a spin-singlet part [7, 137]. The spin-triplet pairing has so far been neglected in neutron-matter calculations, but they can be treated similarly without requiring spin ordering.

To address the limitations of the works mentioned above, we take the most general form of the Pfaffian wave function [136, 6] as the antisymmetric part of our ansatz

$$\Phi_{PJ}(X) = \mathrm{pf} \begin{bmatrix} 0 & \phi(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & \phi(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \phi(\boldsymbol{x}_2, \boldsymbol{x}_1) & 0 & \cdots & \phi(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\boldsymbol{x}_N, \boldsymbol{x}_1) & \phi(\boldsymbol{x}_N, \boldsymbol{x}_2) & \cdots & 0 \end{bmatrix}, \tag{9.6}$$

where we assume the unpolarized case for this initial investigation. We do not keep the spins fixed, nor do we mandate a specific form for the pairing orbital $\phi(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Instead, we capitalize on the universal approximation property of feed-forward neural networks

(FNN) [146] by defining the pairing orbital as

$$\phi(\boldsymbol{x}_i, \boldsymbol{x}_j) = \nu(\boldsymbol{x}_i, \boldsymbol{x}_j) - \nu(\boldsymbol{x}_j, \boldsymbol{x}_i), \tag{9.7}$$

where ν is a dense FNN. The above expression ensures that the Pfaffian is mathematically well-defined, as the matrix is skew-symmetric by construction $\phi(\boldsymbol{x}_i, \boldsymbol{x}_j) = -\phi(\boldsymbol{x}_j, \boldsymbol{x}_i)$. Since ν takes all the degrees of freedom of a given pair of particles as input, including the spins, our pairing orbital has the capacity to discover the spin-singlet and spin-triplet correlations on its own.

This design leaves our Pfaffian-Jastrow (PJ) ansatz agnostic to any particular form of the interaction and systematically improvable by simply increasing the size of ν. The input dimension of ν only depends on the spatial dimension d and not the total number of particles N, leading to an exceptionally scalable ansatz. Given the generality of our formulation, the Pfaffian ansatz calculation cannot be reduced to a determinant of singlet pairing orbitals as in the geminal wave function. Thus, the efficient computation of the Pfaffian is crucial to the scalability of our approach. To this aim, we implement the Pfaffian computation according to Ref. [13].

We further improve the nodal structure of our PJ ansatz through backflow (BF) transformations [97]. To our knowledge, this is the first time neural BF transformations have been used in a Pfaffian wave function, although they have demonstrated their superiority over traditional BF transformations within the Slater-Jastrow formalism in numerous applications [133, 46, 124]. We replace the original single-particle coordinates $\boldsymbol{x}_i$ by new ones $\tilde{\boldsymbol{x}}_i(X)$, such that correlations generated by the presence of all particles are incorporated into the pairing orbital. To ensure that the Pfaffian remains antisymmetric,

the backflow transformation must be permutation equivariant with respect to the original $\boldsymbol{x}_i$, i.e. $\tilde{\boldsymbol{x}}_i$ depends on $\boldsymbol{x}_i$ and is invariant with respect to the set $\{\boldsymbol{x}_j\}_{j\neq i}$. In Sec. **??**, we discuss in detail how the backflow correlations are encoded via a permutation-equivariant message-passing neural network. All calculations labeled as PJ-BF assume that we apply the transformation $\nu(\boldsymbol{x}_i, \boldsymbol{x}_j) \to \nu(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j)$ to the FNN in Eq. (9.7).

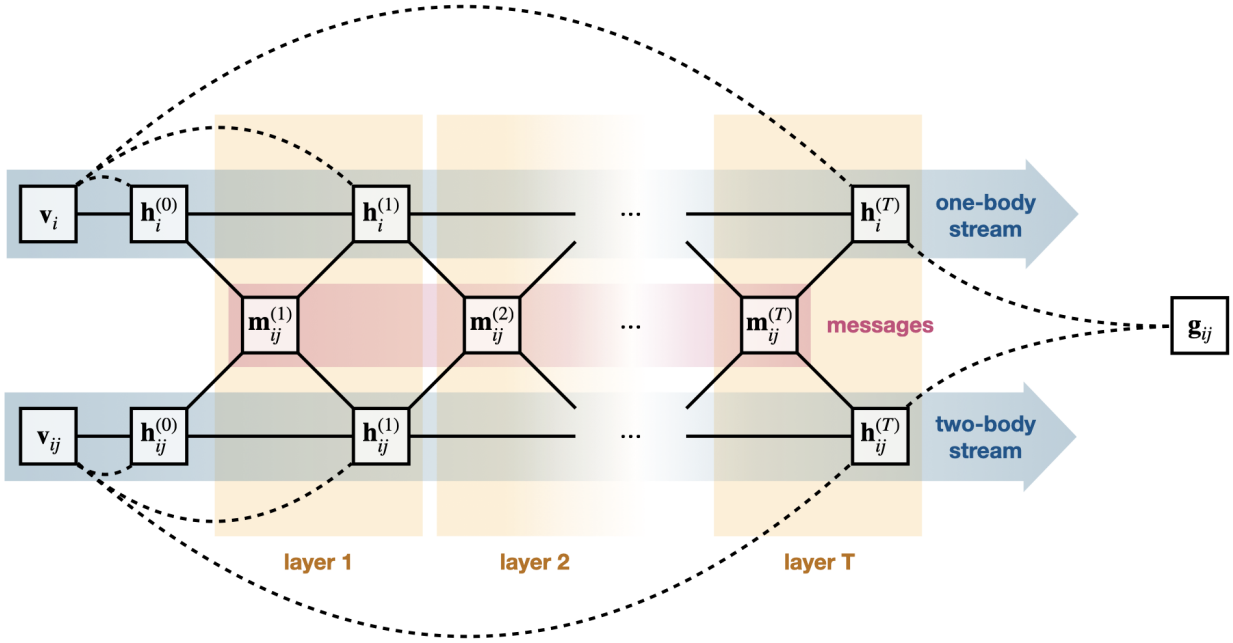

Figure 9.2: Schematic representation of a message-passing neural network with T iterations. Dashed lines represent the concatenation operations, while solid lines represent the parameterized transformations (linear transformations and nonlinear feedforward neural networks). Messages, highlighted in pink, mediate the exchange of information between the one- and two-body streams, in blue. A yellow box indicates a single iteration of the network.

9.3.2 Fixed-Node Slater-Jastrow

For comparison, we will report results obtained using a Slater-Jastrow (SJ) ansatz, which amounts to taking the antisymmetric part of the wave function to be a Slater determinant

of single-particle states

$$\Phi_{SJ}(X) = \det \begin{bmatrix} \phi_1(\boldsymbol{x}_1) & \phi_1(\boldsymbol{x}_2) & \cdots & \phi_1(\boldsymbol{x}_N) \\ \phi_2(\boldsymbol{x}_1) & \phi_2(\boldsymbol{x}_2) & \cdots & \phi_2(\boldsymbol{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_N(\boldsymbol{x}_1) & \phi_N(\boldsymbol{x}_2) & \cdots & \phi_N(\boldsymbol{x}_N) \end{bmatrix}. \tag{9.8}$$

In the fixed-node approximation, the single-particle states are the products of spin eigenstates with definite spin projection on the z-axis s_α and plane wave (PW) orbitals with discrete momenta $\boldsymbol{k}_\alpha = 2\pi \boldsymbol{n}_\alpha / L$, $\boldsymbol{n}_\alpha \in \mathbb{Z}^d$,

$$\phi_\alpha(\boldsymbol{x}_i) = \mathrm{e}^{i\boldsymbol{k}_\alpha \cdot \boldsymbol{r}_i} \chi_\alpha(s_i)\,, \tag{9.9}$$

where $\chi_\alpha(s_i) = \delta_{s_\alpha, s_i}$. Here, $\alpha = (\boldsymbol{k}_\alpha, s_\alpha)$ denotes the quantum numbers characterizing the state. We will label Slater-Jastrow NQS calculations using above plane wave orbitals as SJ-PW.

9.3.3 Slater-Jastrow-Backflow

As in the Pfaffian case, we improve the nodal structure of the above Slater determinant using backflow transformations generated by the message-passing neural network discussed in Sec. **??**. We modify the spatial coordinates of Eq. (9.9) as

$$\boldsymbol{r}_i \to \boldsymbol{r}_i + \boldsymbol{u}_i(X)\,, \tag{9.10}$$

where the complex backflow displacement $\boldsymbol{u}_i(X) \in \mathbb{C}^d$ allows for changes in both the phases and amplitudes of the spatial part of the single-particle states. We also map the single-

particle spinors onto the Bloch sphere as

$$\chi_\alpha(\tilde{\boldsymbol{x}}_i) = \cos\left(\frac{\theta_i(X)}{2}\right)\delta_{s_\alpha,s_i} + \sin\left(\frac{\theta_i(X)}{2}\right)(1-\delta_{s_\alpha,s_i}), \tag{9.11}$$

where $\theta_i(X) \in \mathbb{R}$ is the polar angle on the sphere. Both $\boldsymbol{u}_i(X)$ and $\theta_i(X)$ are permutation-equivariant functions of the original coordinates $\boldsymbol{x}_i$, the functional form of which will be discussed in Sec. 9.3.4.

To motivate the form of the backflow transformation in Eqs. (9.10) and (9.11), let us first revisit the original plane wave orbitals in Eq. (9.9). We simulate our system in the basis

$$|\boldsymbol{x}_i\rangle = |\boldsymbol{r}_i\rangle \otimes |s_i\rangle, \tag{9.12}$$

where $|\boldsymbol{r}_i\rangle$ are eigenstates of the position operator, with $\boldsymbol{r}_i \in \mathbb{R}^d$, and $|s_i\rangle$ are eigenspinors of the S_z operator, with $s_i \in \{\uparrow,\downarrow\}$. In the fixed-node approximation, we take the single-particle states to be products of momentum eigenstates with definite wave vector $\boldsymbol{k}_\alpha = 2\pi\boldsymbol{n}_\alpha/L$, $\boldsymbol{n}_\alpha \in \mathbb{Z}^d$, and eigenspinors with definite spin projection s_α,

$$|\phi_\alpha\rangle = |\boldsymbol{k}_\alpha\rangle \otimes |s_\alpha\rangle. \tag{9.13}$$

Omitting overall normalization constants, the probability amplitude of measuring particle i in state α is

$$\phi_\alpha(\boldsymbol{x}_i) = \langle \boldsymbol{x}_i|\phi_\alpha\rangle = \langle \boldsymbol{r}_i|\boldsymbol{k}_\alpha\rangle\langle s_i|s_\alpha\rangle = e^{i\boldsymbol{k}_\alpha\cdot\boldsymbol{r}_i}\delta_{s_\alpha,s_i}, \tag{9.14}$$

which we call the plane wave orbitals.

Now, let us transform to a new basis with modified position eigenstates and a superposition of eigenspinors

$$|\tilde{\boldsymbol{x}}_i\rangle = |\tilde{\boldsymbol{r}}_i\rangle \otimes |\chi_i\rangle. \tag{9.15}$$

While permutation equivariance is the sole essential property required for the backflow transformation $|\boldsymbol{x}_i\rangle \mapsto |\tilde{\boldsymbol{x}}_i\rangle$ to preserve the antisymmetry of the fermionic wave function, an additional property is desirable for computational convenience. Specifically, when the transformation depends on certain parameters, we aim to have $|\tilde{\boldsymbol{x}}_i\rangle = |\boldsymbol{x}_i\rangle$ when the parameters are identically zero. Then, nonzero parameters signify deviations from the original plane wave orbitals, such that less training is required compared to completely trainable orbitals.

An appropriate spatial transformation is trivial. We simply define new parameters $\boldsymbol{u}_i \in \mathbb{C}^d$, called the backflow displacement, and shift the coordinates as $\boldsymbol{r}'_i = \boldsymbol{r}_i + \boldsymbol{u}_i$. The backflow displacement is complex, allowing for changes in both the phases and amplitudes of the original plane wave orbitals.

For the spin part of the transformation, we look to spinors on the Bloch sphere for inspiration,

$$|\chi_i\rangle = \cos\left(\frac{\theta_i}{2}\right)|s_i\rangle + \sin\left(\frac{\theta_i}{2}\right)\sigma_i^x|s_i\rangle. \tag{9.16}$$

In the above, we have introduced another backflow variable $\theta_i \in \mathbb{R}$ akin to the polar angle of a Bloch spinor, and we have excluded the relative phase in favor of a completely real-valued wave function. We also write the superposition in terms of $|s_i\rangle$ and the Pauli X-operator σ_i^x, which flips the spin of the i-th particle, rather than $|\uparrow\rangle$ and $|\downarrow\rangle$. This way, it is obvious

$\langle\chi_i\vert\chi_j\rangle$	$\theta_i = \theta_j$	$\theta_i - \theta_j = \pm\pi$
$s_i = s_j$	1	0
$s_i \neq s_j$	0	± 1

Table 9.1: The limiting cases of the overlap between two neural backflow spinors inspired by spinors on the Bloch sphere.

that $|\chi_i\rangle = |s_i\rangle$ when $\theta_i = 0$, as desired. The overlap of two spinors is given by

$$\begin{aligned}
\langle\chi_i|\chi_j\rangle &= \cos\left(\frac{\theta_i}{2}\right)\cos\left(\frac{\theta_j}{2}\right)\langle s_i|s_j\rangle \\
&\quad + \sin\left(\frac{\theta_i}{2}\right)\cos\left(\frac{\theta_j}{2}\right)\langle s_i|\sigma_i^{x\dagger}|s_j\rangle \\
&\quad + \cos\left(\frac{\theta_i}{2}\right)\sin\left(\frac{\theta_j}{2}\right)\langle s_i|\sigma_j^{x}|s_j\rangle \\
&\quad + \sin\left(\frac{\theta_i}{2}\right)\sin\left(\frac{\theta_j}{2}\right)\langle s_i|\sigma_i^{x\dagger}\sigma_j^{x}|s_j\rangle \\
&= \left[\cos\left(\frac{\chi_i}{2}\right)\cos\left(\frac{\chi_j}{2}\right) + \sin\left(\frac{\chi_i}{2}\right)\sin\left(\frac{\chi_j}{2}\right)\right]\delta_{s_i,s_j} \\
&\quad \left[\sin\left(\frac{\theta_i}{2}\right)\cos\left(\frac{\theta_j}{2}\right) - \cos\left(\frac{\theta_i}{2}\right)\sin\left(\frac{\theta_j}{2}\right)\right](1-\delta_{s_i,s_j}) \\
&= \cos\left(\frac{\theta_i-\theta_j}{2}\right)\delta_{s_i,s_j} + \sin\left(\frac{\theta_i-\theta_j}{2}\right)(1-\delta_{s_i,s_j}),
\end{aligned} \tag{9.17}$$

where the limiting cases are summarized in the following table.

Finally, we can compute the backflow orbitals with the transformed degrees of freedom

$$\begin{aligned}
\phi_\alpha(\tilde{\boldsymbol{x}}_i) &= \langle\tilde{\boldsymbol{x}}_i|\phi_\alpha\rangle \\
&= \langle\tilde{\boldsymbol{r}}_i|\boldsymbol{k}_\alpha\rangle\langle\chi_i|s_\alpha\rangle \\
&= e^{i\boldsymbol{k}_\alpha\cdot\tilde{\boldsymbol{r}}_i}\left(\cos\left(\frac{\theta_i}{2}\right)\langle s_i|s_\alpha\rangle + \sin\left(\frac{\theta_i}{2}\right)\langle s_i|\sigma_i^{x\dagger}|s_\alpha\rangle\right) \\
&= e^{i\boldsymbol{k}_\alpha\cdot(\boldsymbol{r}_i+\boldsymbol{u}_i)}\left(\cos\left(\frac{\theta_i}{2}\right)\delta_{s_\alpha,s_i} + \sin\left(\frac{\theta_i}{2}\right)(1-\delta_{s_\alpha,s_i})\right).
\end{aligned} \tag{9.18}$$

In Eqs. (9.10) and (9.11), we use the notation $\boldsymbol{u}_i(X)$ and $\theta_i(X)$ to emphasize that the

backflow “parameters" we define here are not variational parameters, but a function of all other particles. More specifically, they are permutation-equivariant functions of the original $\boldsymbol{x}_i$, whose functional forms depend on the outputs of the permutation-equivariant message-passing neural network (MPNN) described in Sec. 9.3.4.

In addition to omitting the relative phase in Eq. (9.16) so that the spinors remain real, we map the spatial components of the plane wave and backflow orbitals to the equivalent real ones. For the unpolarized system of fermions, the latter mapping is

$$\left\{ \begin{array}{c} e^{i\boldsymbol{k}_\alpha \cdot (\boldsymbol{r}_i + \boldsymbol{u}_i)} \\ e^{-i\boldsymbol{k}_\alpha \cdot (\boldsymbol{r}_i + \boldsymbol{u}_i)} \end{array} \right\} \mapsto \left\{ \begin{array}{c} e^{\boldsymbol{k}_\alpha \cdot \mathrm{Im}(\boldsymbol{u}_i)} \cos\left(\boldsymbol{k}_\alpha \cdot (\boldsymbol{r}_i + \mathrm{Re}(\boldsymbol{u}_i))\right) \\ e^{\boldsymbol{k}_\alpha \cdot \mathrm{Im}(\boldsymbol{u}_i)} \sin\left(\boldsymbol{k}_\alpha \cdot (\boldsymbol{r}_i + \mathrm{Re}(\boldsymbol{u}_i))\right) \end{array} \right\}. \tag{9.19}$$

Remember that the spatial inputs to the MPNN are L-periodic, which guarantees that $\boldsymbol{u}_i$ is as well. Therefore it is less redundant to compute

$$\left\{ \begin{array}{c} e^{\mathrm{Im}(\boldsymbol{u}_i)} \cos\left(\boldsymbol{k}_\alpha \cdot \boldsymbol{r}_i + \mathrm{Re}(\boldsymbol{u}_i)\right) \\ e^{\mathrm{Im}(\boldsymbol{u}_i)} \sin\left(\boldsymbol{k}_\alpha \cdot \boldsymbol{r}_i + \mathrm{Re}(\boldsymbol{u}_i)\right) \end{array} \right\}, \tag{9.20}$$

instead of the right-hand side of Eq. (9.19). Then the modified phase $\mathrm{Re}(\boldsymbol{u}_i)$ and amplitude $e^{\mathrm{Im}(\boldsymbol{u}_i)}$ still have the correct periodicity, but they do not reduce to the trivial case when $\boldsymbol{k}_\alpha = \boldsymbol{0}$. In the end, this last step is a small detail and does not affect the final result.

9.3.4 Message-Passing Neural Network

Implementing the aforementioned neural-network quantum states is possible using X as direct inputs to the appropriate FNNs and Deep-Sets [12]. Still, it is advantageous to devise new inputs that already capture a large portion of the correlations. As in Ref. [2], we employ a permutation-equivariant message-passing neural network (MPNN) to

iteratively build correlations into new one-body and two-body features from the original "visible" features. The visible features are chosen to be

$$\boldsymbol{v}_i = (s_i)\,, \tag{9.21}$$

$$\boldsymbol{v}_{ij} = \big(\boldsymbol{r}_{ij}, \|\boldsymbol{r}_{ij}\|, s_{ij}\big)\,, \tag{9.22}$$

with the separation vectors $\boldsymbol{r}_{ij} = \boldsymbol{r}_i - \boldsymbol{r}_j$ and distances $\|\boldsymbol{r}_{ij}\| = r_{ij}$ replaced by their L-periodic surrogates

$$\boldsymbol{r}_{ij} \mapsto \big(\cos(2\pi\boldsymbol{r}_{ij}/L), \sin(2\pi\boldsymbol{r}_{ij}/L)\big)\,, \tag{9.23}$$

$$\|\boldsymbol{r}_{ij}\| \mapsto \|\sin(\pi\boldsymbol{r}_{ij}/L)\|, \tag{9.24}$$

and the quantity $s_{ij} \equiv 2\delta_{s_i,s_j} - 1$ assigned a value of $+1$ for aligned spins and -1 for anti-aligned spins. Note that we have excluded explicit dependence on the particle positions $\boldsymbol{r}_i$ in the visible one-body features, thereby enforcing translational invariance in the new features. Linear transformations are applied to and concatenated with each feature to obtain the initial hidden features

$$\boldsymbol{h}_i^{(0)} = (\boldsymbol{v}_i, A\boldsymbol{v}_i), \tag{9.25}$$

$$\boldsymbol{h}_{ij}^{(0)} = (\boldsymbol{v}_{ij}, B\boldsymbol{v}_{ij}). \tag{9.26}$$

The main purpose of the linear transformations is to preprocess the input data. Still, they also help simplify the implementation by keeping the dimension of the hidden features $\boldsymbol{h}_i^{(t)}$, $\boldsymbol{h}_{ij}^{(t)}$ constant for all t. In each iteration, $t = 1, \ldots, T$ of the MPNN, information is exchanged

between the one- and two-body streams through a so-called "message"

$$\boldsymbol{m}_{ij}^{(t)} = \boldsymbol{M}_t \left(\boldsymbol{h}_i^{(t-1)},\ \boldsymbol{h}_j^{(t-1)},\ \boldsymbol{h}_{ij}^{(t-1)} \right). \tag{9.27}$$

For a given particle i, relevant messages are collected and pooled together to destroy the ordering with respect to all other particles $j \neq i$,

$$\boldsymbol{m}_i^{(t)} = \boldsymbol{pool} \left(\{ \boldsymbol{m}_{ij}^{(t)} \mid j \neq i \} \right). \tag{9.28}$$

The pooling operation ***pool*** collapses the order of the elements in the set it acts upon and produces a vector with the same dimension as an individual element. Throughout this work, we use logsumexp-pooling, the smooth variation of max-pooling.

The pairwise messages $\boldsymbol{m}_{ij}^{(t)}$ and the implied particle messages $\boldsymbol{m}_i^{(t)}$ are then used to update the hidden features

$$\boldsymbol{h}_i^{(t)} = \left(\boldsymbol{v}_i,\ \boldsymbol{F}_t \left(\boldsymbol{h}_i^{(t-1)},\ \boldsymbol{m}_i^{(t)} \right) \right), \tag{9.29}$$

$$\boldsymbol{h}_{ij}^{(t)} = \left(\boldsymbol{v}_{ij},\ \boldsymbol{G}_t \left(\boldsymbol{h}_{ij}^{(t-1)},\ \boldsymbol{m}_{ij}^{(t)} \right) \right). \tag{9.30}$$

The functions $\boldsymbol{M}_t$, $\boldsymbol{F}_t$, and $\boldsymbol{G}_t$ are all unique FNNs with the same output dimension as the linear preprocessors A and B. By incorporating concatenated skip connections to the visible features, we guarantee that the signal originating from the raw data remains discernible even as the MPNN depth T increases. Finally, we combine the resulting outputs $\boldsymbol{h}_i^{(T)}$ and $\boldsymbol{h}_{ij}^{(T)}$ into pairwise feature vectors

$$\boldsymbol{g}_{ij} = \left(\boldsymbol{h}_i^{(T)}, \boldsymbol{h}_j^{(T)}, \boldsymbol{h}_{ij}^{(T)} \right) \tag{9.31}$$

to feed into subsequent networks. The flow of information through the MPNN can be visualized in Fig. 9.2. Notice how the hidden features in a given layer depend on the hidden features of the previous layer *and* the original visible features.

For all our NQS, we use a Jastrow correlator based on a Deep-Set [12] to enforce permutation invariance over the set of all pairwise features

$$J(X) = \rho\Big(\boldsymbol{pool}\Big(\{\boldsymbol{\zeta}(\boldsymbol{g}_{ij}) \mid i \neq j\}\Big)\Big). \tag{9.32}$$

Here, ρ and $\boldsymbol{\zeta}$ are FNNs, and the pooling operation is the same as in Eq. (9.28). While many Jastrow functions are typically designed to satisfy Kato's cusp condition [147] for specific systems, we take a different approach and allow our neural networks to learn the cusp fully. The short-range behavior of the ground state is particularly important for the UFG, so leaving our NQS completely unbiased serves as an important test for evaluating the overall capabilities of NQS.

The Slater-Jastrow ansatz with plane wave orbitals (SJ-PW) does not require any additional neural networks beyond ρ and $\boldsymbol{\zeta}$, so it establishes a baseline for the number of trainable parameters in this work. On the other hand, the backflow variables $\boldsymbol{u}_i$ and θ_i for the Slater-Jastrow ansatz with backflow orbitals (SJ-BF) are the outputs of another Deep-Set

$$(\mathrm{Re}(\boldsymbol{u}_i), \mathrm{Im}(\boldsymbol{u}_i), \theta_i) = \boldsymbol{\rho}_{bf}\Big(\boldsymbol{pool}\Big(\{\boldsymbol{\zeta}_{bf}(\boldsymbol{g}_{ij}) \mid j \neq i\}\Big)\Big), \tag{9.33}$$

which is permutation invariant with respect to all $j \neq i$ by construction. The size of $\boldsymbol{\rho}_{bf}$ and $\boldsymbol{\zeta}_{bf}$ determines the number of extra variational parameters present in the SJ-BF ansatz compared to the SJ-PW ansatz. For the PJ-BF ansatz, the pairing orbital ν in Eq. (9.7) simply takes $\boldsymbol{g}_{ij}$ as input in place of $(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Therefore, the number of additional variational

parameters in the PJ-BF ansatz relative to the SJ-PW ansatz is determined by the size of ν.

All of the feedforward neural networks mentioned throughout this section have at least two hidden layers with 16 nodes each. The activation function is GELU [148] and the weights/biases are initialized with glorot normal/zeros unless pretrained parameters are used.

It is worth highlighting that the individual feedforward neural networks within our NQS are solely dependent on the spatial dimension d and not the system size N. Therefore, even though this study focuses on benchmarking the $N = 14$ case, the trained NQS can be used as starting points for larger even-N, unpolarized systems without requiring any modifications to the network structure. This is an example of transfer learning, a powerful strategy that involves applying knowledge gained from solving one problem to another, often more challenging problem.

9.4 Variational Monte Carlo and Training

We train our NQS by minimizing the energy

$$E(\boldsymbol{p}) \equiv \frac{\langle \Psi(\boldsymbol{p}) | H | \Psi(\boldsymbol{p}) \rangle}{\langle \Psi(\boldsymbol{p}) | \Psi(\boldsymbol{p}) \rangle} \tag{9.34}$$

with respect to the variational parameters $\boldsymbol{p}$. To compute the energy and its gradient $\nabla_{\boldsymbol{p}} E$ using Monte Carlo integration, we sample positions R and spins S from $|\Psi(R, S)|^2$ in a way that preserves periodicity and total spin projection on the z-axis, as in Refs. [125, 130]. Since the ordering of the spins is not fixed, our ansätze can be immediately applied to any continuous-space Hamiltonian that exchange spin, such as Ref. [149].

A sophisticated optimization technique is critical for achieving an ansatz that is both

compact and expressive. In this work, we employ the stochastic reconfiguration [9] (SR) algorithm with regularization based on the RMSprop method, introduced in Ref. [130]. The parameters are updated as

$$\boldsymbol{p} \leftarrow \boldsymbol{p} - \eta G^{-1} \nabla_{\boldsymbol{p}} E, \tag{9.35}$$

where η is a constant learning rate and G is the quantum geometric tensor [10].

Due to the strong and short-range nature of the interaction in Eq. (9.2), it is likely for the optimization process to get trapped in a local minimum when initialized with random parameters. To avoid this problem, we use transfer learning by pretraining the NQS on a softer interaction ($\mu = 5$) before proceeding to harder ones ($\mu = 10, 20, 40$). Not only does this approach improve the final converged energy, but the efficiency of the optimization process overall. The training for lower values of μ can handle a more aggressive learning rate δ and fewer samples. As a general guideline, we reduce δ by a factor of 10 and double the number of samples each time the value of μ is doubled. The number of optimization steps required for training ranges from $\mathcal{O}(10^3)$ to $\mathcal{O}(10^4)$, depending on whether the NQS were pretrained or initialized with random parameters.

9.5 Diffusion Monte Carlo

The fixed-node DMC calculations are performed as described in Ref. [150]. The initial state is prepared using VMC methods with a variational wave function with the same general form as Eq. (9.3). Note that, within the fixed-node approximation, DMC provides a strict upperbound to the energy of the system. While DMC is a precise method, its accuracy relies on the choice of nodal surface and the quality of the preceding VMC calculation. The

symmetric Jastrow factor is given by

$$J(X) = \sum_{ii'}^{n} u(r_{ii'}), \tag{9.36}$$

$$u(r) = K \tanh(\mu_J r) \cosh(\gamma r)/r \,, \tag{9.37}$$

where $n = N/2$ and the unprimed and primed indicies denote the spin-up and spin-down particles, respectively. The parameters K and γ are adjusted so that $u(d) = 0$ and $u'(d) = 0$, and μ_J and d are variational parameters. Considering that the s-wave channel dominates the interaction, the antisymmetric part is given by the number-projected BCS wave function

$$\Phi_{BCS}(X) = \det \begin{bmatrix} \phi(\boldsymbol{r}_{11'}) & \phi(\boldsymbol{r}_{12'}) & \cdots & \phi(\boldsymbol{r}_{1n'}) \\ \phi(\boldsymbol{r}_{21'}) & \phi(\boldsymbol{r}_{22'}) & \cdots & \phi(\boldsymbol{r}_{2n'}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\boldsymbol{r}_{n1'}) & \phi(\boldsymbol{r}_{n2'}) & \cdots & \phi(\boldsymbol{r}_{nn'}) \end{bmatrix}, \tag{9.38}$$

with the pairing orbitals

$$\phi(\boldsymbol{r}) = \tilde{\beta}(r) + \sum_i a(k_i^2) e^{i\boldsymbol{k}_i \cdot \boldsymbol{r}} \,, \tag{9.39}$$

$$\tilde{\beta}(r) = \beta(r) + \beta(L - r) - 2\beta(L/2) \,, \tag{9.40}$$

$$\beta(r) = (1 + cbr)\ (1 - e^{-dbr}) \frac{e^{-br}}{dbr} \ . \tag{9.41}$$

The parameters $a(k_i^2)$, b and d are obtained by minimizing the energy, and c is chosen so that the function β has zero slope at the origin. If we instead let $\beta = 0$ and restrict the sum in Eq. (9.39) to momentum states filled up to k_F, the antisymmetric part is equivalent to

the Slater determinant with single-particle plane waves as in Eqs. (9.8) and (9.9). Since this approach does not involve pairing, we will refer to the related DMC results as DMC-PW. Conversely, the approach that accounts for pairing will be identified as DMC-BCS.

It should be emphasized that the BCS wave function of Eq. (9.38) is a special case of the generalized Pfaffian of Eq. (9.6). In fact, it can be easily shown [6] that by only retaining the spin-singlet blocks, the calculation of the Pfaffian reduces to the determinant of spin-singlet block.

9.6 Results

9.6.1 Energy

We first compare the performance of the various neural-network quantum states outlined in Sec. 9.3 as the message-passing neural network (MPNN) depth T is varied. As shown in Fig. 9.3, the final converged energies per particle for the Slater-Jastrow ansatz with plane wave orbitals (SJ-PW) decreases monotonically towards the corresponding DMC-PW benchmark, with remarkable agreement at $T = 5$. This behavior echoes the findings of Ref. [151], and demonstrates the impact of the MPNN on the flexibility of our Jastrow. Incorporating backflow correlations into the Slater-Jastrow ansatz (SJ-BF) significantly improves results compared to the fixed-node approach with PW, but more than half of the discrepancy between the two DMC energies remains. Due to the observed weak dependence on T, it is unlikely that further increasing T would yield a substantial improvement in energy. The SJ-BF ansatz may be able to achieve energies more similar to the DMC-BCS benchmark by increasing the width of the feedforward neural networks. Still, the associated computational expenses are expected to be prohibitively high. Therefore, we turn our attention to the Pfaffian-Jastrow-Backflow (PJ-BF) ansatz. Even

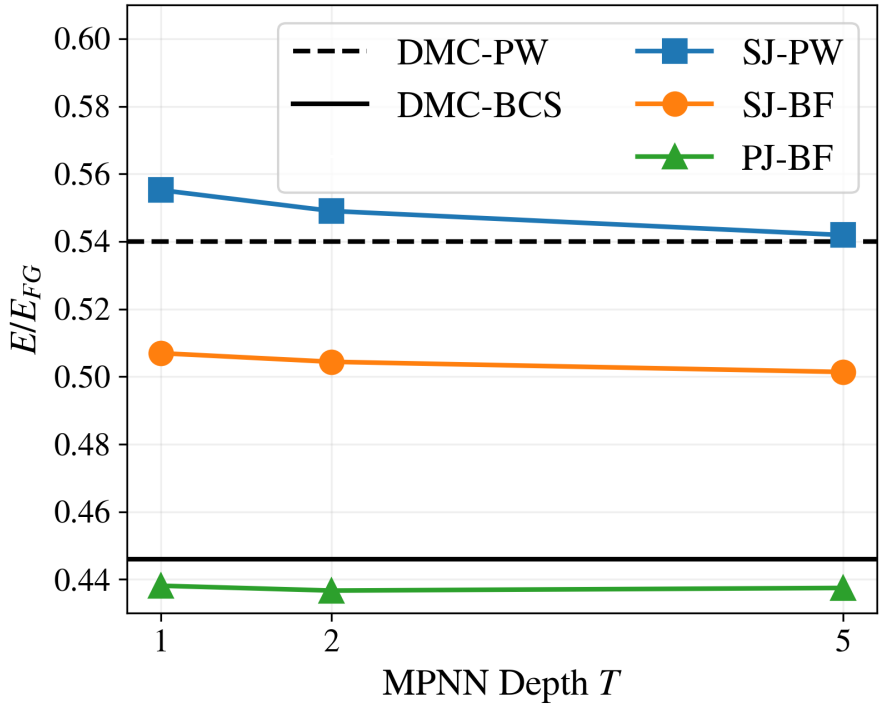

Figure 9.3: Ground-state energies per particle as a function of the MPNN depth T for the SJ-PW (blue squares), SJ-BF (orange circles), and PJ-BF (green triangles) ansätze. The interaction parameters are set to $v_0 = 1$ and $\mu = 5$, corresponding to an effective range of $r_e k_F = 0.4$. The DMC benchmark energies with and without pairing are displayed as solid and dashed lines, respectively.

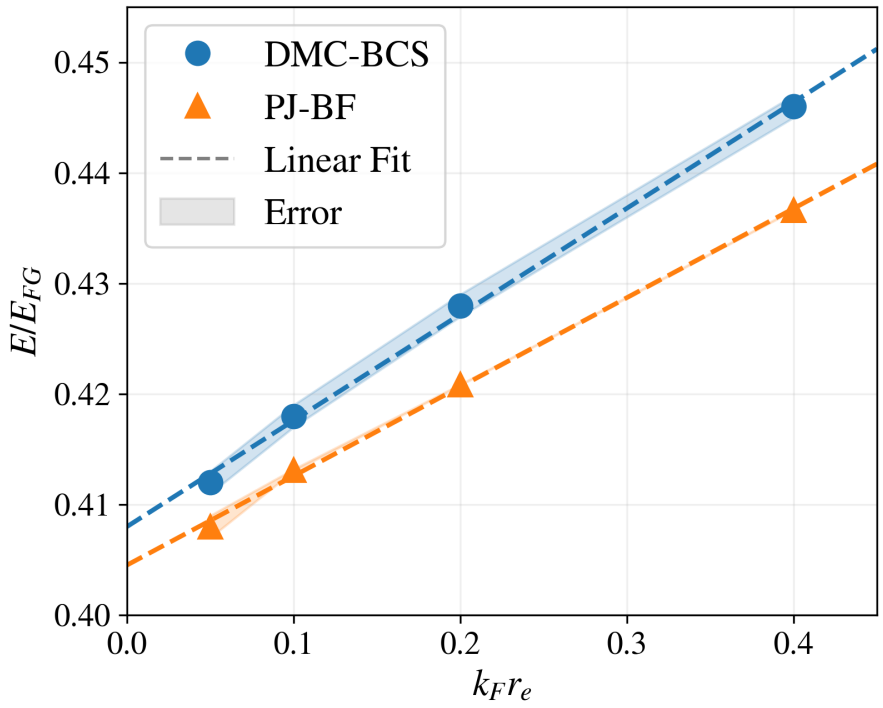

Figure 9.4: Ground-state energies per particle as a function of the effective range. The DMC-BCS benchmark energies (blue circles) and the Pfaffian-Jastrow with backflow (PJ-BF) energies (orange triangles) are extrapolated to zero effective range using linear fits (dashed lines). The shaded regions are the error bands for the DMC-BCS and PJ-BF energies.

with a single MPNN layer, the PJ-BF ansatz easily outperforms DMC-BCS while also possessing fewer parameters (∼5600 v.s. ∼6200) than the single-layer SJ-BF ansatz. The overall dependence on the MPNN depth is weak, with $T = 2$ giving slightly lower energy and variance than $T = 5$. For the remainder of our analysis, we will use the PJ-BF ansatz with $T = 2$, which contains about 8500 variational parameters.

As the unitary limit is characterized by a vanishing effective range, we study how the ground-state energy responds to changing $k_F r_e$ in Fig. 9.4. The PJ-BF ansatz gives energies ∼1-2% lower than DMC-BCS as the effective range is decreased from $k_F r_e = 0.4$ to $k_F r_e = 0.1$. At $k_F r_e = 0.05$, our energy falls below the range of the DMC-BCS error band, suggesting

μ	$k_F r_e$	DMC-BCS	PJ-BF
5	0.4	0.446(1)	0.4366(3)
10	0.2	0.428(1)	0.4208(3)
20	0.1	0.418(1)	0.4131(8)
40	0.05	0.412(1)	0.408(1)
∞	0.0	0.408(1)*	0.405(1)*

Table 9.2: Energy per particle for various values of μ and the corresponding values of r_e. The values with asterisks (*) are extrapolations from the linear fits shown in Fig. 9.4. The parameter $v_0 = 1$ is fixed.

our approach is likely to maintain its superior performance as r_e is decreased further. To estimate the energy at zero effective range, we also perform simple linear fits — See Table 9.2 for the extrapolated values. Note that our results have been obtained by simulating a system of $N = 14$ particles for benchmark purposes. In order to obtain energies closer to the thermodynamic limit, further simulations with more particles will be needed [122, 152].

9.6.2 Pair Distribution Functions

In Fig. 9.5, we show the opposite-spin pair distribution functions at unitarity for $\mu = 5$, 10, and 20. Notice how the peaks of the distributions at $k_F r = 0$ grow roughly quadratically with μ, demonstrating the presence of strong pairing correlations as we approach the unitary limit $\mu \to \infty$. Clearly, the short-range character of the distributions are important to capture at unitarity, as they begin to converge around $k_F r \gtrsim 0.4$.

Fig. 9.6 presents a complementary set of opposite-spin pair distribution functions in the crossover region with fixed effective range of $k_F r_e = 0.2$. When we lean towards the BCS phase $1/ak_F = -0.5$, the long-range tail of the density is enhanced compared to the unitary case $1/ak_F = 0$. On the other hand, the tail is diminished in the BEC phase $1/ak_F = -0.5$, suggesting the initiation of dimer formation. The differences in the peaks of the distributions are not as dramatic as in Fig. 9.5, but they are consistent with the expected behavior in the

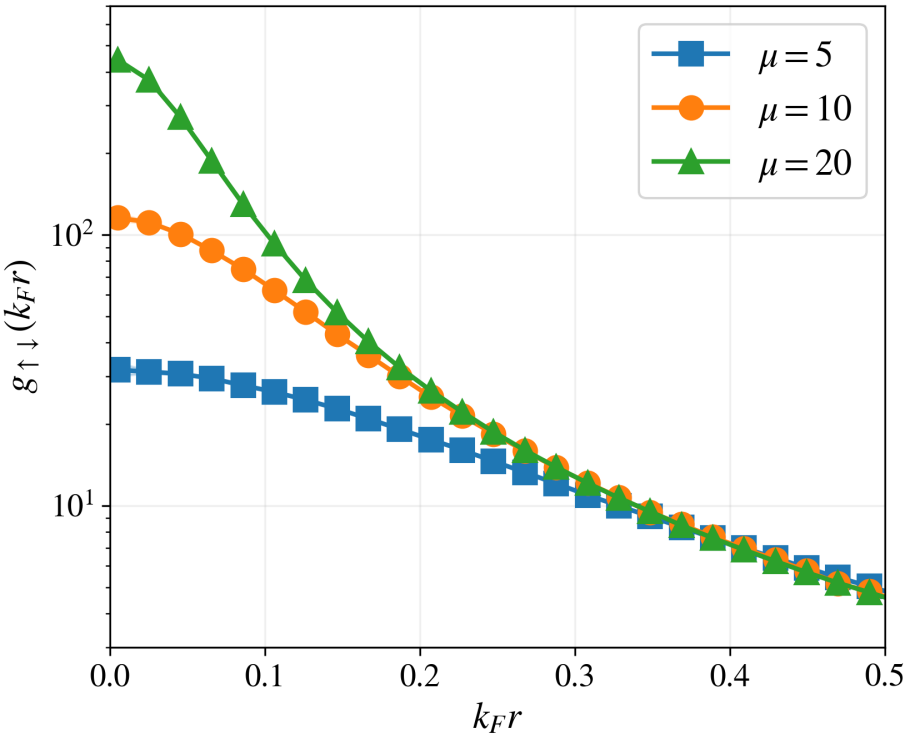

Figure 9.5: Opposite-spin pair densities as a function of small $k_F r$ at unitarity ($v_0 = 1$) and $\mu = 5$ (blue squares), $\mu = 10$ (orange circles), and $\mu = 20$ (green triangles).

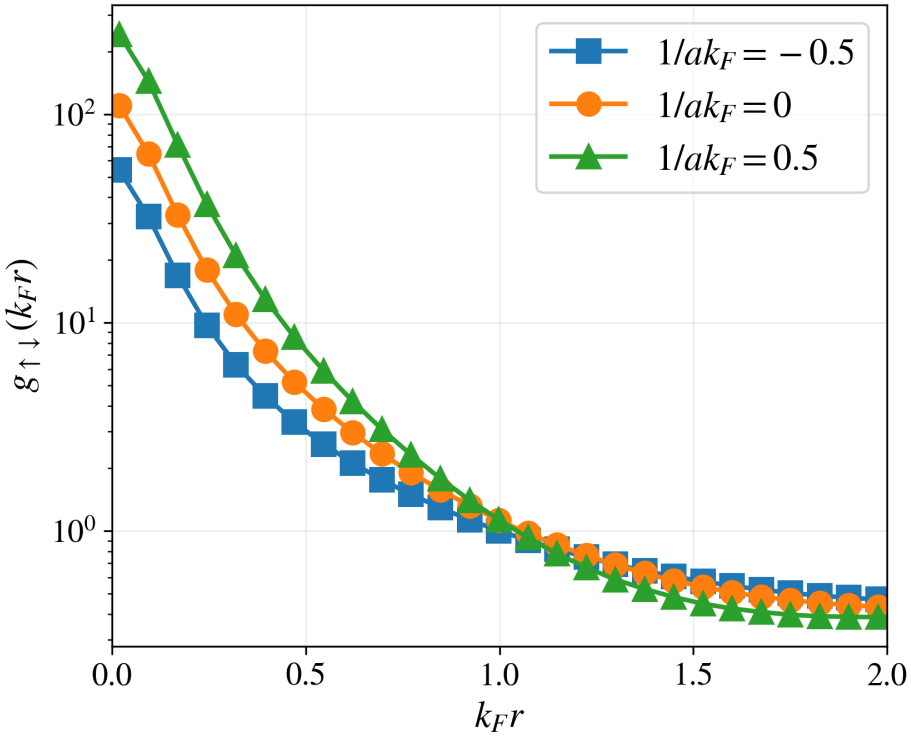

Figure 9.6: Opposite-spin pair densities in the crossover region for the BCS phase $1/ak_F = -0.5$ (blue squares), unitarity $1/ak_F = 0$ (orange circles), and BEC phase $1/ak_F = 0.5$ (green triangles). The effective range of all cases are fixed $k_F r_e = 0.2$. See Table 9.4 for the corresponding values of v_0 and μ.

$1/ak_F$	v_0	μ	DMC-BCS	PJ-BF
-1	0.879214	11.06247	0.801(1)	0.7930(2)
-0.5	0.933216	10.55715	0.705(1)	0.6937(3)
-0.2	0.971423	10.23012	0.578(1)	0.5671(3)
-0.1	0.985366	10.11637	0.510(1)	0.5014(4)
0	1.0	10.0	0.428(1)	0.4208(3)
0.1	1.015388	9.880801	0.328(1)	0.3218(3)
0.2	1.031602	9.758564	0.208(1)	0.2017(3)
0.5	1.086081	9.371025	-0.319(1)	-0.3244(4)
1	1.204354	8.632898	-2.053(1)	-2.0566(6)

Table 9.3: Energies per particle and interaction parameters for the two-body potential in Eq. (9.2) giving different scattering lengths with the same effective range $k_F r_e = 0.2$.

BCS and BEC regimes near unitarity.

Finally, we explore the BCS-BEC crossover region for a fixed effective range $k_F r_e = 0.2$ in Fig. 9.7. See Table 9.4 for the values of the interaction parameters v_0 and μ, as well as the corresponding DMC-BCS benchmarks and the PJ-BF ansatz results. The cases closer to unitarity were used to pretrain the cases further away. In the BCS regime, our PJ-BF ansatz consistently yields energies $\sim 0.01 E_{FG}$ lower than those obtained from DMC-BCS, albeit with slightly inferior performance in the BEC regime. We attribute this effect to the need for increased flexibility in capturing the short-range behavior of pairs in the BEC regime. Simply increasing the size of feedforward neural network ν that defines the pairing orbital should alleviate this discrepancy. In any case, the PJ-BF ansatz gives lower energies than DMC-BCS for all scattering lengths tested.

9.6.3 Pairing Gap

The pairing gap can be evaluated using

$$\Delta(N) = E(N) - \frac{1}{2}\Big(E(N+1) + E(N-1)\Big), \tag{9.42}$$

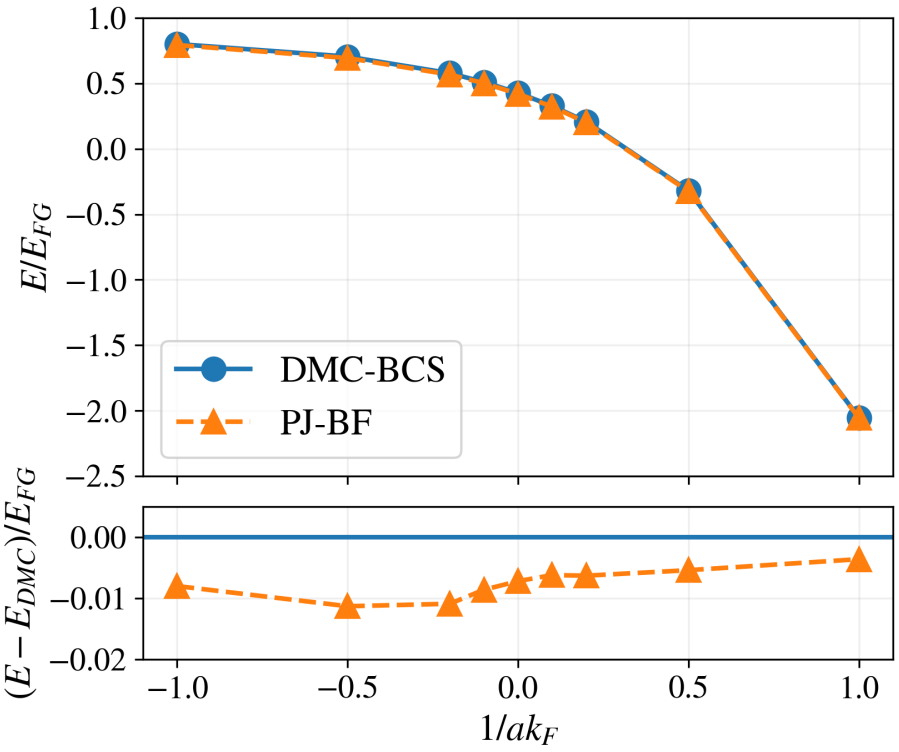

Figure 9.7: Upper panel: Energy per particle in the BCS-BEC crossover region as a function of the scattering length a for a fixed effective range $k_F r_e = 0.2$. Lower panel: Difference between Pfaffian-Jastrow with backflow (PJ-BF) and DMC-BCS benchmark energies. See Table 9.4 for the corresponding values of v_0 and μ.

N	DMC-BCS	PJ-BF
14	0.428(1)	0.4208(3)
15	0.4900(2)	0.4766(5)
15	0.5357(2)*	0.5209(5)*
16	0.4240(2)	0.4177(4)

Table 9.4: Energies per particle for different numbers of particles, with $k_F r_e = 0.2$. Values with asterisks (*) indicate translation-invariant calculations.

where N is taken to be odd and $E(N)$ denotes the *total* energy for N particles. Since our construction of the Pfaffian-Jastrow-Backflow ansatz remains independent of the system size N (assuming N is even), the trained ansatz for $N = 14$ serves as the initial state for the $N = 16$ calculation. To accommodate odd-N cases, we incorporated an additional neural network to represent the unpaired single-particle orbital. Furthermore, for the $N = 15$ calculation, the pairing orbital from the $N = 16$ calculation was utilized as the starting point.

In Table 9.5, we present two different evaluations of the pairing gap, corresponding to the DMC-BCS results using a single-particle orbital with and without translation invariance. To enforce translation invariance in our odd-N PJ-BF ansatz, we simply take the one-body output of the message-passing neural network $\boldsymbol{h}_i^{(T)}$ as input to the single-particle pairing orbital. To break translation invariance, we concatenate the positions $\left(\boldsymbol{r}_i, \boldsymbol{h}_i^{(T)}\right)$ with the MPNN output, keeping in mind the spins have already been included in the latter. Due to our lower energies for $N = 15$ compared to DMC-BCS, we predict a pairing gap that is $\sim 0.10 E_{FG}$ smaller in the non-translation invariant case and $\sim 0.12 E_{FG}$ smaller in the translation invariant case.

9.7 Conclusions and perspectives

In this study, we propose a novel neural-network quantum state based on the Pfaffian-Jastrow (PJ) framework that utilizes a message-passing neural network (MPNN) to encode pairing

$\Delta(15)/E_{FG}$	DMC-BCS	PJ-BF
Non-translation invariant	0.9620	0.8618
Translation invariant	1.6475	1.5263

Table 9.5: Energies per particle for different numbers of particles, with $k_F r_e = 0.2$. Values with asterisks (*) indicate translation-invariant calculations.

and backflow (BF) correlations. We evaluate its performance against comparable Slater-Jastrow (SJ) ansätze with identical MPNN architectures. Our results indicate that increasing the depth of the MPNN systematically improves the performance of the SJ ansätze, but backflow correlations within the single-particle picture are still insufficient in capturing all pairing correlations. However, we demonstrate that a simple and compact PJ-BF ansatz surpasses the DMC-BCS benchmark with ease.

Transfer learning has proven to be an essential tool in this work. It enables the realization of the unitary limit in a controlled manner, mitigating the risk of becoming trapped in local minima. It also allows for the efficient exploration of regions beyond unitarity, unlocking new avenues for studying the BCS-BEC crossover. Transfer learning will remain a crucial part of our training procedure as we move to larger systems. All unpolarized systems can be treated with a single architecture, while the $N \pm 1$ systems can be treated by introducing one additional FNN to represent the unpaired single-particle orbital. This modification is straightforward to implement, making the calculation of the gap in the thermodynamic limit more accessible and enabling further advancements in our work.

Besides calculating the gap for larger systems, our next steps include a direct comparison with the STU Pfaffian wave function of Ref. [6]. We also plan to perform a more careful extrapolation to the $r_e \to 0$ limit since we have used relatively large values of $k_F r_e$ for this initial investigation. However, more hyperparameter tuning will be needed, especially about the width of the hidden layers, since the smaller values of r_e will require more flexibility.

Our Pfaffian-Jastrow-Backflow NQS displays immense potential in the study of ultracold Fermi gases. Unlike conventional methods, our PJ-BF ansatz is not subject to biases arising from physical intuition or a lack thereof, as it does not require specifying a particular form for the pairing orbitals. For this reason, it can be readily applied to other strongly-correlated systems, including molecules and other strongly-correlated quantum systems. In stark contrast to the commonly used geminal wave function, our ansatz does not rely on ordering the spin of the interacting fermions, and it is therefore amenable to Hamiltonians that exchange spin, such as those modeling nuclear dynamics. In this regard, we anticipate calculations of atomic nuclei and low-density isospin-asymmetric nucleonic matter and carry out detailed investigations on the nature of nuclear pairing [153].

When the stochastic reconfiguration algorithm and transfer learning techniques are combined with the enforcement of translational, parity, and time-reversal symmetries, highly non-perturbative correlations can be encoded in a small number of parameters by modern standards. This approach will pave the way for future developments in the study of many-body systems, as it offers a powerful tool for encoding correlations in a compact and computationally feasible manner.

Note Added: A work very recently appeared in pre-print [154] introduces neural backflow transformations in a geminal wave function and studies the unitary Fermi gas. We leave systematic comparisons between the two approaches to future works while already observing that the Pfaffian wave function is a strict generalization of the geminal wave function [155, 6].

10 Conclusions and Perspectives

The emergence of neural-network quantum states has ushered in a transformative era in our exploration of complex quantum many-body systems. These novel models, which draw inspiration from the interdisciplinary marriage of machine learning and quantum physics, demonstrate an unprecedented capacity to capture nonlinear correlations in the many-body wave function while maintaining flexibility in adapting to a variety of Hamiltonians. By omitting the cusp condition and constraining only the most essential symmetries and boundary conditions, we prove that neural-network quantum states have the representational power to reproduce state-of-the-art results without imposing biased physical intuition.

Before neural-network quantum states, the diffusion Monte Carlo (DMC) method unequivocally outperformed the variational Monte Carlo (VMC) method, with the former treated as the gold standard. The difference in performance between the two methods can be attributed to imperfections in the parameterization of the trial wave function in the variational Monte Carlo calculation. However, the diffusion Monte Carlo method, while extremely accurate, still relies on the fixed-node approximation to handle the fermion sign problem. With the adoption of neural-network quantum states, we find that the VMC method often surpasses the DMC method, implying that the errors in the DMC calculation are due to the fixed-node approximation.

In addition to avoiding the fermion sign problem, a neural-network quantum state trained with the VMC algorithm produces a closed form expression for the ground state wave function. This is in contrast to the DMC method, which produces a final distribution

of Markov chain Monte Carlo walkers rather than an analytical formula. Though the variational parameters of the neural-network quantum state are harder to interpret compared to traditional VMC methods, they can be used to reproduce the wave function and extract observables.

The universal approximation property of feedforward neural networks endows them with an impressive capacity to handle a wide range of Hamiltonians. This generality comes at a cost, as neural-network quantum states often require many orders of magnitude more variational parameters than conventional trial wave functions. Throughout this work, we prioritize a compact ansatz by improving upon other aspects of the calculation, such as the optimization scheme and sampling algorithm, in addition to increasing the flexibility of the wave function through imposing discrete symmetries, attention mechanisms, and message-passing neural networks. As a result, our neural-network quantum states utilize substantially fewer parameters than related networks applied to similar challenges.

Gazing ahead, this young field will continue to develop as cutting-edge neural-network architectures are adapted into neural-network quantum states. The most compact designs will prevail as the most scalable solutions, allowing us to tackle larger systems than ever before. This is particularly significant for infinite matter, as finite-size errors can hide crucial information about phase transitions, in which long-range correlations dominate.

Besides addressing stationary problems, neural-network quantum states offer the potential to describe the dynamic evolution of quantum many-body systems in real time. While this feat has been achieved for spin systems, its application to continuous-space systems remains unexplored. In the context of the unitary Fermi gas (as discussed in Sec. 9), investigating real-time dynamics could provide irrefutable evidence of superfluidity, manifested through the formation of quantum vortices. Moreover, neural-network quantum

states hold promise for applications to the finite-temperature variational Monte Carlo method. This avenue may allow for the prediction of the critical temperature of superfluidity for the unitary Fermi gas—a parameter speculated to be among the highest across all known systems. The implications are far-reaching, resonating with the pursuit of room-temperature superconductivity, which is arises from the same pairing mechanism. As neural-network quantum states continue to unravel the complex behaviors of quantum systems, they open doors to a more comprehensive understanding of phenomena that have the potential to revolutionize materials science and technology.

All of the periodic systems discussed in this work (neutron matter, electron gas, ultra-cold Fermi gases) are relevant to the description of neutron stars. Adding asymmetric nuclear matter into the list of systems would further broaden the scope of our understanding and enable a more comprehensive exploration of the intricate physics governing these astrophysical phenomena. The ground state wave function could reveal the formation of nuclear clusters in their various possible shapes depending on the density and proton fraction. The real-time evolution of asymmetric nuclear matter could shed light on the nuclear reactions that can occur in neutron stars, like the fusion of alpha particles.

In conclusion, neural-network quantum states have achieved impressive milestones in their relatively short lifespan. Looking forward, their potential remains substantial and encouraging. Yet, there's still much ground to cover, inviting ongoing research and exploration in this evolving field.

BIBLIOGRAPHY

[1] Giuseppe Carleo and Matthias Troyer. "Solving the quantum many-body problem with artificial neural networks". *Science*. 2017.

[2] Gabriel Pescia et al. "Message-Passing Neural Quantum States for the Homogeneous Electron Gas". arXiv: 2305.07240 (quant-ph). 2023.

[3] David Pfau et al. "Ab initio solution of the many-electron Schrödinger equation with deep neural networks". *Phys. Rev. Res.* 2020.

[4] Alessandro Lovato et al. "Hidden-nucleons neural-network quantum states for the nuclear many-body problem". *Phys. Rev. Res.* 2022.

[5] Jane Kim et al. "Neural-network quantum states for ultra-cold Fermi gases". arXiv: 2305.08831 (cond-mat.quant-gas). 2023.

[6] M. Bajdich et al. "Pfaffian pairing and backflow wavefunctions for electronic structure quantum Monte Carlo methods". *Phys. Rev. B*. 2008.

[7] S. Gandolfi et al. "Equation of state of low-density neutron matter, and the $^1S_0$ pairing gap". *Phys. Rev. C*. 2009.

[8] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015.

[9] Sandro Sorella. "Wave function optimization in the variational Monte Carlo method". *Phys. Rev. B*. 2005.

[10] James Stokes et al. "Quantum Natural Gradient". *Quantum*. 2020.

[11] M. P. Kuchera et al. "Machine Learning Methods for Track Classification in the AT-TPC". *Nucl. Instrum. Meth. Phys. Res. A*. 2019.

[12] Manzil Zaheer et al. "Deep sets". *Advances in neural information processing systems*. 2017.

[13] M. Wimmer. "Algorithm 923: Efficient Numerical Computation of the Pfaffian for Dense and Banded Skew-Symmetric Matrices". *ACM Trans. Math. Softw.* 2012.

[14] Hiroki Saito. "Method to Solve Quantum Few-Body Problems with Artificial Neural Networks". *J. Phys. Soc. Jpn.* 2018.

[15] Bryce Fore et al. "Dilute neutron star matter from neural-network quantum states". *Phys. Rev. Res.* 2023.

[16] B. P. Abbott et al. "GW170817: Observation of Gravitational Waves from a Binary Neutron Star Inspiral". *Phys. Rev. Lett.* 2017.

[17] B. P. Abbott et al. "Multi-messenger Observations of a Binary Neutron Star Merger". *Astrophys. J.* 2017.

[18] Andrea Sabatucci and Omar Benhar. "Tidal Deformation of Neutron Stars from Microscopic Models of Nuclear Dynamics". *Phys. Rev. C.* 2020.

[19] Peter Senger. "Probing Dense Nuclear Matter in the Laboratory: Experiments at FAIR and NICA". *Universe.* 2021.

[20] C. Drischler et al. "Neutron matter from chiral two- and three-nucleon calculations up to N^3LO". *Phys. Rev. C.* 2016.

[21] M. Piarulli et al. "Benchmark calculations of pure neutron matter with realistic nucleon-nucleon interactions". *Phys. Rev. C.* 2020.

[22] D. Lonardoni et al. "Nuclear and neutron-star matter from local chiral interactions". *Phys. Rev. Res.* 2020.

[23] W. G. Jiang et al. "Accurate bulk properties of nuclei from $A = 2$ to ∞ from potentials with Δ isobars". *Phys. Rev. C.* 2020.

[24] Francesca Sammarruca and Randy Millerson. "Overview of symmetric nuclear matter properties from chiral interactions up to fourth order of the chiral expansion". *Phys. Rev. C.* 2021.

[25] H. Heiselberg and M. Hjorth-Jensen. "Phases of dense matter in neutron stars". *Phys. Rep.* 2000.

[26] Andrea Sabatucci et al. "Sensitivity of neutron star observations to three-nucleon forces". *Phys. Rev. D.* 2022.

[27] Armen Sedrakian, John W Clark, and Mark Alford. "Pairing in Fermionic Systems". 2006.

[28] Omar Benhar and Giulia De Rosi. "Superfluid Gap in Neutron Matter from a Microscopic Effective Interaction". *J. Low Temp. Phys.* 2017.

[29] D. J. Dean and M. Hjorth-Jensen. "Pairing in nuclear systems: from neutron stars to finite nuclei". *Rev. Mod. Phys.* 2003.

[30] Dima G. Yakovlev and C. J. Pethick. "Neutron star cooling". *Ann. Rev. Astron. Astrophys.* 2004.

[31] Dany Page et al. “Rapid Cooling of the Neutron Star in Cassiopeia A Triggered by Neutron Superfluidity in Dense Matter”. *Phys. Rev. Lett.* 2011.

[32] C. Monrozeau, J. Margueron, and N. Sandulescu. “Nuclear superfluidity and cooling time of neutron-star crust”. *Phys. Rev. C.* 2007.

[33] Frédéric Nowacki, Alexandre Obertelli, and Alfredo Poves. “The neutron-rich edge of the nuclear landscape: Experiment and theory.” *Prog. Part. Nucl. Phys.* 2021.

[34] J. Carlson et al. “Quantum Monte Carlo methods for nuclear physics”. *Rev. Mod. Phys.* 2015.

[35] K. E. Schmidt and S. Fantoni. “A quantum Monte Carlo method for nucleon systems”. *Phys. Lett. B.* 1999.

[36] A. Lovato et al. “Benchmark calculations of infinite neutron matter with realistic two- and three-nucleon potentials”. *Phys. Rev. C.* 2022.

[37] S. Gandolfi et al. “Equation of state of superfluid neutron matter and the calculation of S(0)-1 pairing gap”. *Phys. Rev. Lett.* 2008.

[38] Stefano Gandolfi et al. “The 1S0 Pairing Gap in Neutron Matter”. *Condens. Mat.* 2022.

[39] M. Bajdich et al. “Pfaffian pairing wave functions in electronic structure quantum Monte Carlo”. *Phys. Rev. Lett.* 2006.

[40] J. W. T. Keeble and A. Rios. “Machine learning the deuteron”. *Phys. Lett. B.* 2020.

[41] Corey Adams et al. “Variational Monte Carlo Calculations of A$\leq$4 Nuclei with an Artificial Neural-Network Correlator Ansatz”. *Phys. Rev. Lett.* 2021.

[42] Alex Gnech et al. “Nuclei with up to $\boldsymbol{A = 6}$ nucleons with artificial neural network wave functions”. *Few Body Syst.* 2022.

[43] A. Lovato et al. “Hidden-nucleons neural-network quantum states for the nuclear many-body problem”. 2022.

[44] Y. L. Yang and P. W. Zhao. “A consistent description of the relativistic effects and three-body interactions in atomic nuclei”. *Phys. Lett. B.* 2022.

[45] Mauro Rigo et al. “Solving the nuclear pairing model with neural network quantum states”. 2022.

[46] Jan Hermann, Zeno Schätzle, and Frank Noé. “Deep-neural-network solution of the electronic Schrödinger equation”. *Nature Chemistry.* 2020.

[47] David Pfau et al. “Ab initio solution of the many-electron Schrödinger equation with deep neural networks”. *Phys. Rev. Res.* 2020.

[48] L. Brualla et al. “Spin orbit induced backflow in neutron matter with auxiliary field diffusion Monte Carlo”. *Phys. Rev. C.* 2003.

[49] R. Schiavilla et al. “Two- and three-nucleon contact interactions and ground-state energies of light- and medium-mass nuclei”. *Phys. Rev. C.* 2021.

[50] Sebastian König et al. “Nuclear Physics Around the Unitarity Limit”. *Phys. Rev. Lett.* 2017.

[51] A. Kievsky et al. “Correlations imposed by the unitary limit between few-nucleon systems, nuclear matter and neutron stars”. *Phys. Rev. Lett.* 2018.

[52] Robert B. Wiringa, V. G. J. Stoks, and R. Schiavilla. “An Accurate nucleon-nucleon potential with charge independence breaking”. *Phys. Rev. C.* 1995.

[53] B. S. Pudliner et al. “Quantum Monte Carlo calculations of A <= 6 nuclei”. *Phys. Rev. Lett.* 1995.

[54] A. Akmal, V. R. Pandharipande, and D. G. Ravenhall. “The Equation of state of nucleon matter and neutron star structure”. *Phys. Rev. C.* 1998.

[55] R. Machleidt and D.R. Entem. “Chiral effective field theory and nuclear forces”. *Phys. Rep.* 2011.

[56] Javier Robledo Moreno et al. “Fermionic wave functions from neural-network constrained hidden states”. *Proceedings of the National Academy of Sciences of the United States of America.* 2022.

[57] Edward Wagstaff et al. “On the Limitations of Representing Functions on Sets”. *arXiv e-prints.* 2019.

[58] Di Luo and Bryan K. Clark. “Backflow Transformations via Neural Networks for Quantum Many-Body Wave Functions”. *Phys. Rev. Lett.* 2019.

[59] Gabriel Pescia et al. “Neural-network quantum states for periodic systems in continuous space”. *Phys. Rev. Res.* 2022.

[60] A. Sarsa et al. “Neutron matter at zero temperature with auxiliary field diffusion Monte Carlo”. *Phys. Rev.* 2003.

[61] Nicholas Metropolis et al. “Equation of State Calculations by Fast Computing Machines”. *J. Chem. Phys.* 1953.

[62] S. Gandolfi et al. “Equation of state of low-density neutron matter, and the 1S0 pairing gap”. *Phys. Rev. C - Nuclear Physics*. 2009.

[63] M. Piarulli et al. “Light-nuclei spectra from chiral dynamics”. *Phys. Rev. Lett.* 2018.

[64] John W. Negele and D. Vautherin. “Neutron star matter at subnuclear densities”. *Nucl. Phys. A*. 1973.

[65] Brian M Austin, Dmitry Yu Zubarev, and William A Lester Jr. “Quantum Monte Carlo and related approaches”. *Chemical reviews*. 2012.

[66] J Carlson et al. “Quantum Monte Carlo methods for nuclear physics”. *Reviews of Modern Physics*. 2015.

[67] Anthony J. Leggett. “A theoretical description of the new phases of liquid ^{3}He”. *Rev. Mod. Phys.* 1975.

[68] Michael Tinkham. “Introduction to superconductivity”. 2004.

[69] Juan Carrasquilla. “Machine learning for quantum matter”. *Advances in Physics: X*. 2020.

[70] Kenny Choo et al. “Symmetries and Many-Body Excitations with Neural-Network Quantum States”. *Phys. Rev. Lett.* 2018.

[71] Francesco Ferrari, Federico Becca, and Juan Carrasquilla. “Neural Gutzwiller-projected variational wave functions”. *Phys. Rev. B*. 2019.

[72] Mohamed Hibat-Allah et al. “Recurrent neural network wave functions”. *Phys. Rev. Res.* 2020.

[73] Marin Bukov, Markus Schmitt, and Maxime Dupont. “Learning the ground state of a non-stoquastic quantum Hamiltonian in a rugged neural network landscape”. *SciPost Phys.* 2021.

[74] Yusuke Nomura and Masatoshi Imada. “Dirac-Type Nodal Spin Liquid Revealed by Refined Quantum Many-Body Solver Using Neural-Network Wave Function, Correlation Ratio, and Level Spectroscopy”. *Phys. Rev. X*. 2021.

[75] Nikita Astrakhantsev et al. “Broken-Symmetry Ground States of the Heisenberg Model on the Pyrochlore Lattice”. *Phys. Rev. X*. 2021.

[76] Dong-Ling Deng, Xiaopeng Li, and S Das Sarma. “Quantum entanglement in neural network states”. *Phys. Rev. X*. 2017.

[77] Yoav Levine et al. “Quantum Entanglement in Deep Learning Architectures”. *Phys. Rev. Lett.* 2019.

[78] Di Luo and Bryan K. Clark. "Backflow Transformations via Neural Networks for Quantum Many-Body Wave Functions". *Phys. Rev. Lett.* 2019.

[79] Kenny Choo, Antonio Mezzacapo, and Giuseppe Carleo. "Fermionic neural-network states for ab-initio electronic structure". *Nature communications*. 2020.

[80] Javier Robledo Moreno et al. "Fermionic wave functions from neural-network constrained hidden states". *Proceedings of the National Academy of Sciences*. 2022.

[81] Jan Hermann, Zeno Schätzle, and Frank Noé. "Deep-neural-network solution of the electronic Schrödinger equation". *Nature Chemistry*. 2020.

[82] MT Entwistle et al. "Electronic excited states in deep variational Monte Carlo". *Nature Communications*. 2023.

[83] Gabriel Pescia et al. "Neural-network quantum states for periodic systems in continuous space". *Phys. Rev. Res.* 2022.

[84] Max Wilson et al. "Wave function Ansatz (but Periodic) Networks and the Homogeneous Electron Gas". *arXiv preprint arXiv:2202.04622*. 2022.

[85] Gino Cassella et al. "Discovering Quantum Phase Transitions with Fermionic Neural Networks". *Phys. Rev. Lett.* 2023.

[86] Corey Adams et al. "Variational Monte Carlo Calculations of $A \leq 4$ Nuclei with an Artificial Neural-Network Correlator Ansatz". *Phys. Rev. Lett.* 2021.

[87] Alex Gnech et al. "Nuclei with Up to \ {A= 6}$A = 6$ Nucleons with Artificial Neural Network Wave Functions". *Few-Body Systems*. 2022.

[88] Xiang Li, Zhe Li, and Ji Chen. "Ab initio calculation of real solids via neural network ansatz". *Nature Communications*. 2022.

[89] Hao Xie, Linfeng Zhang, and Lei Wang. "$m^\ast$ of two-dimensional electron gas: a neural canonical transformation study". 2022.

[90] Julien Toulouse and C. J. Umrigar. "Optimization of quantum Monte Carlo wave functions by energy minimization". *J. Chem. Phys.* 2007.

[91] Sandro Sorella. "Green Function Monte Carlo with Stochastic Reconfiguration". *Phys. Rev. Lett.* 1998.

[92] C Lin, FH Zong, and David M Ceperley. "Twist-averaged boundary conditions in continuum quantum Monte Carlo algorithms". *Phys. Rev. E*. 2001.

[93] ND Drummond and RJ Needs. "Diffusion quantum Monte Carlo calculation of the quasiparticle effective mass of the two-dimensional homogeneous electron gas". *Phys. Rev. B.* 2013.

[94] Ke Liao et al. "Towards efficient and accurate ab initio solutions to periodic systems via transcorrelation and coupled cluster theory". *Phys. Rev. Research.* 2021.

[95] Sam Azadi, ND Drummond, and SM Vinko. "Correlation energy of the paramagnetic electron gas at the thermodynamic limit". *arXiv preprint arXiv:2209.10227.* 2022.

[96] Justin Gilmer et al. "Neural message passing for quantum chemistry". *International conference on machine learning.* 2017.

[97] R. P. Feynman and Michael Cohen. "Energy Spectrum of the Excitations in Liquid Helium". *Phys. Rev.* 1956.

[98] Yongkyung Kwon, D. M. Ceperley, and Richard M. Martin. "Effects of backflow correlation in the three-dimensional electron gas: Quantum Monte Carlo study". *Phys. Rev. B.* 1998.

[99] Yongkyung Kwon, DM Ceperley, and Richard M Martin. "Effects of three-body and backflow correlations in the two-dimensional electron gas". *Phys. Rev. B.* 1993.

[100] Michele Taddei et al. "Iterative backflow renormalization procedure for many-body ground-state wave functions of strongly interacting normal Fermi liquids". *Phys. Rev. B.* 2015.

[101] Michele Ruggeri, Saverio Moroni, and Markus Holzmann. "Nonlinear Network Description for Many-Body Quantum Systems in Continuous Space". *Phys. Rev. Lett.* 2018.

[102] Ashish Vaswani et al. "Attention is all you need". *Advances in neural information processing systems.* 2017.

[103] Dan Hendrycks and Kevin Gimpel. "Gaussian error linear units (gelus)". *arXiv preprint arXiv:1606.08415.* 2016.

[104] Ingrid von Glehn, James S Spencer, and David Pfau. "A Self-Attention Ansatz for Ab-initio Quantum Chemistry". *arXiv preprint arXiv:2211.13672.* 2022.

[105] Kaiming He et al. "Deep residual learning for image recognition". *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016.

[106] D Pines. "Elementary Excitations In Solids, edited by JD Jackson and D. Pines". 1963.

[107] Paul P Ewald. "Die Berechnung optischer und elektrostatischer Gitterpotentiale". *Annalen der physik*. 1921.

[108] Louisa M. Fraser et al. "Finite-size effects and Coulomb interactions in quantum Monte Carlo calculations for homogeneous systems with periodic boundary conditions". *Phys. Rev. B*. 1996.

[109] Abdulnour Y Toukmaji and John A Board Jr. "Ewald summation techniques in perspective: a survey". *Computer physics communications*. 1996.

[110] P. LPópez Ríos et al. "Inhomogeneous backflow transformations in quantum Monte Carlo calculations". *Phys. Rev. E*. 2006.

[111] Sam Azadi and N. D. Drummond. "Low-density phase diagram of the three-dimensional electron gas". *Phys. Rev. B*. 2022.

[112] E. Wigner. "On the Interaction of Electrons in Metals". *Phys. Rev.* 1934.

[113] ND Drummond et al. "Diffusion quantum Monte Carlo study of three-dimensional Wigner crystals". *Phys. Rev. B*. 2004.

[114] S Gandolfi. "Quantum Monte Carlo study of strongly interacting Fermi gases". *Journal of Physics: Conference Series*. 2014.

[115] Alexandros Gezerlis and J. Carlson. "Strongly paired fermions: Cold atoms and neutron matter". *Phys. Rev. C*. 2008.

[116] Stefano Gandolfi, Alexandros Gezerlis, and J. Carlson. "Neutron Matter from Low to High Density". *Annual Review of Nuclear and Particle Science*. 2015.

[117] Wynn C. G. Ho et al. "Tests of the nuclear equation of state and superfluid and superconducting gaps using the Cassiopeia A neutron star". *Phys. Rev. C*. 2015.

[118] J. Carlson et al. "Auxiliary-field quantum Monte Carlo method for strongly paired fermions". *Phys. Rev. A*. 2011.

[119] J. Carlson and Sanjay Reddy. "Superfluid Pairing Gap in Strong Coupling". *Phys. Rev. Lett.* 2008.

[120] S. Gandolfi, K. E. Schmidt, and J. Carlson. "BEC-BCS crossover and universal relations in unitary Fermi gases". *Phys. Rev. A*. 2011.

[121] W. M. C. Foulkes et al. "Quantum Monte Carlo simulations of solids". *Rev. Mod. Phys.* 2001.

[122] Michael McNeil Forbes, Stefano Gandolfi, and Alexandros Gezerlis. "Resonantly Interacting Fermions in a Box". *Phys. Rev. Lett.* 2011.

[123] M. J. H. Ku et al. “Revealing the Superfluid Lambda Transition in the Universal Thermodynamics of a Unitary Fermi Gas”. *Science*. 2012.

[124] David Pfau et al. “Ab initio solution of the many-electron Schrödinger equation with deep neural networks”. *Phys. Rev. Res.* 2020.

[125] Gabriel Pescia et al. “Neural-network quantum states for periodic systems in continuous space”. *Phys. Rev. Res.* 2022.

[126] Max Wilson et al. “Wave function Ansatz (but Periodic) Networks and the Homogeneous Electron Gas”. *arXiv e-prints*. 2022.

[127] Gino Cassella et al. “Discovering Quantum Phase Transitions with Fermionic Neural Networks”. *Phys. Rev. Lett.* 2023.

[128] C. Adams et al. “Variational Monte Carlo calculations of $A \leq 4$ nuclei with an artificial neural-network correlator ansatz”. *Phys. Rev. Lett.* 2021.

[129] Alex Gnech et al. “Nuclei with up to $A = 6$ nucleons with artificial neural network wave functions”. *Few-Body Systems*. 2021.

[130] Alessandro Lovato et al. “Hidden-nucleons neural-network quantum states for the nuclear many-body problem”. *Phys. Rev. Res.* 2022.

[131] Y. L. Yang and P. W. Zhao. “A consistent description of the relativistic effects and three-body interactions in atomic nuclei”. *Phys. Lett. B*. 2022.

[132] Mauro Rigo et al. “Solving the nuclear pairing model with neural network quantum states”. *Phys. Rev. E*. 2023.

[133] Di Luo and Bryan K. Clark. “Backflow Transformations via Neural Networks for Quantum Many-Body Wave Functions”. *Phys. Rev. Lett.* 2019.

[134] Michele Casula and Sandro Sorella. “Geminal wave functions with Jastrow correlation: A first application to atoms”. *J. Chem. Phys.* 2003.

[135] Michele Casula, Claudio Attaccalite, and Sandro Sorella. “Correlated geminal wave function for molecules: An efficient resonating valence bond approach”. *J. Comput. Phys.* 2004.

[136] M. Bajdich et al. “Pfaffian Pairing Wave Functions in Electronic-Structure Quantum MonteCarlo Simulations”. *Phys. Rev. Lett.* 2006.

[137] Stefano Gandolfi et al. “Atomic nuclei from quantum Monte Carlo calculations with chiral EFT interactions”. *Front. in Phys.* 2020.

[138] Yusuke Nomura et al. "Restricted Boltzmann machine learning for solving strongly correlated quantum systems". *Phys. Rev. B*. 2017.

[139] Yusuke Nomura and Masatoshi Imada. "Dirac-type nodal spin liquid revealed by refined quantum many-body solver using neural-network wave function, correlation ratio, and level spectroscopy". *Phys. Rev. X*. 2021.

[140] S. Y. Chang et al. "Quantum Monte Carlo studies of superfluid Fermi gases". *Phys. Rev. A*. 2004.

[141] Alexandros Gezerlis et al. "Heavy-Light Fermion Mixtures at Unitarity". *Phys. Rev. Lett.* 2009.

[142] Andrew J. Morris, P. López Ríos, and R. J. Needs. "Ultracold atoms at unitarity within quantum Monte Carlo methods". *Phys. Rev. A*. 2010.

[143] J. Carlson et al. "Superfluid Fermi Gases with Large Scattering Length". *Phys. Rev. Lett.* 2003.

[144] Alexander Galea et al. "Diffusion Monte Carlo study of strongly interacting two-dimensional Fermi gases". *Phys. Rev. A*. 2016.

[145] Maria Piarulli and Ingo Tews. "Local Nucleon-Nucleon and Three-Nucleon Interactions Within Chiral Effective Field Theory". *Front. in Phys.* 2020.

[146] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". *Neural Networks*. 1991.

[147] Tosio Kato. "On the eigenfunctions of many-particle systems in quantum mechanics". *Communications on Pure and Applied Mathematics*. 1957.

[148] Dan Hendrycks and Kevin Gimpel. "Gaussian error linear units (gelus)". *arXiv preprint arXiv:1606.08415*. 2016.

[149] R. Schiavilla et al. "Two- and three-nucleon contact interactions and ground-state energies of light- and medium-mass nuclei". *Phys. Rev. C*. 2021.

[150] Renato Pessoa et al. "Contact interaction in a unitary ultracold Fermi gas". *Phys. Rev. A*. 2015.

[151] Z. Schätzle, J. Hermann, and F. Noé. "Convergence to the fixed-node limit in deep variational Monte Carlo". *J. Chem. Phys.* 2021.

[152] Michael McNeil Forbes, Stefano Gandolfi, and Alexandros Gezerlis. "Effective-range dependence of resonantly interacting fermions". *Phys. Rev. A*. 2012.

[153] D. J. Dean and M. Hjorth-Jensen. “Pairing in nuclear systems: From neutron stars to finite nuclei”. *Rev. Mod. Phys.* 2003.

[154] Wan Tong Lou et al. “Neural Wave Functions for Superfluids”. arXiv: 2305.06989 (cond-mat.quant-gas). 2023.

[155] Claudio Genovese et al. “General Correlated Geminal Ansatz for Electronic Structure Calculations: Exploiting Pfaffians in Place of Determinants”. *Journal of Chemical Theory and Computation*. 2020.