Neural Network Antisymmetries for Quantum Many-Body Simulation

by

Jeffmin Lin

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Mathematics

in the

Graduate Division

of the

University of California, Berkeley

Spring 2022

Neural Network Antisymmetries for Quantum Many-Body Simulation

Abstract

Neural Network Antisymmetries for Quantum Many-Body Simulation

by

Jeffmin Lin

Doctor of Philosophy in Mathematics

University of California, Berkeley

Professor Lin Lin, Chair

This work is concerned with the accurate numerical simulation of the many-electron problem, which involves the modeling of the electron wavefunction, from which all of the properties of a chemical or condensed matter system can, in principle, be computed. This problem poses a number of challenges, including the effective parametrization of the wavefunction space. The combination of neural networks and quantum Monte Carlo methods has arisen as a promising path forward for highly accurate electronic structure calculations. Previous proposals have combined equivariant neural network layers with a final antisymmetric layer in order to satisfy the antisymmetry requirements of the electronic wavefunction. However, to date it is unclear if one can represent arbitrary antisymmetric functions of physical interest, and it is difficult to precisely measure the expressiveness of the antisymmetric layer.

In the first chapter of this dissertation, we begin by introducing the electronic structure problem and the variational nature of finding the lowest energy wavefunction, or ground state. We describe Metropolis Monte Carlo sampling techniques, as well as a simplifying reduction to the number of degrees of freedom. We conclude the first chapter with a brief discussion of some optimization techniques used to address the variational ground state problem once a trial parametrization has been established.

In the next chapter, we then introduce the form of some modern neural-network based trial wavefunctions. We attempt to investigate the expressiveness of the antisymmetric layers by proposing explicitly antisymmetrized universal neural network layers. This approach has a computational cost which increases factorially with respect to the system size, but we are nonetheless able to apply it to small systems to better understand how the structure of the antisymmetric layer affects its performance. We first introduce a generic antisymmetric (GA) neural network layer, which we use to replace the entire antisymmetric layer of the highly accurate ansatz known as the FermiNet. We also consider a factorized antisymmetric (FA) layer which more directly generalizes the FermiNet by replacing the products of determinants with products of antisymmetrized neural networks.

We next investigate the numerical performance of these explicitly antisymmetrized ansatzes. We demonstrate that the FermiNet-GA architecture can yield effectively the exact ground state energy for small atoms and molecules. We find, interestingly, that the resulting FermiNet-FA architecture does not outperform the FermiNet. This strongly suggests that the sum of products of antisymmetries is a key limiting aspect of the FermiNet architecture. To explore this further, we also investigate a slight modification of the FermiNet, called the full determinant mode, which replaces each product of determinants with a single combined determinant. We find that the full single-determinant FermiNet closes a large part of the gap between the standard single-determinant FermiNet and FermiNet-GA on small atomic and molecular problems. Surprisingly, on the nitrogen molecule at a dissociating bond length of 4.0 Bohr, the full single-determinant FermiNet can significantly outperform the largest standard FermiNet calculation with 64 determinants, yielding an energy within 0.4 kcal/mol of the best available computational benchmark.

In the final chapter, we introduce the VMCNet repository, which was used to implement the numerical experiments previously described. VMCNet is intended to be a flexible, general purpose VMC framework which interfaces natively with the JAX library for rapid prototyping, performance benefits due to just-in-time XLA compilation, and easy dispatch to multiple GPU systems. We describe both the Python API, intended for more complex use-cases that require customization of finer details of the VMC loop, and the command-line interface, which provides a more streamlined and encapsulated way to run variational Monte Carlo experiments, including those described in this dissertation.

Contents

Acknowledgments

I give the deepest thanks to my doctoral advisor, Lin Lin, for his continued support through the ups and downs of my graduate studies. Any of his students can confirm that he is a limitless source of scientific energy and ideas. Without his advice and advocacy I would not have made it past even my third year. I am also grateful for the many enjoyable and insightful conversations with other students and postdocs who studied and toiled on these or similar problems with me, perhaps foremost Gil Goldshlager in the summer and fall of 2021, but also including (but not limited to) Jiefu Zhang, Xiaojie Wu, Qinyi Zhu, Jiahao Yao, and Fabian Faulstich.

I cannot thank my parents Chi and Yishun enough for their ceaseless toil to provide me taller shoulders to stand on. I am thankful for my brother Christopher's love, including his endless willingness to be a sounding board for whatever ideas happen to be stuck in my head. Finally, I am forever indebted to Ming-Ming Tran for her companionship and patience in my life each day. She makes me feel special.

Thank you all for being a part of my journey!

Preface

The highly accurate simulation of many-electron quantum physics is an enterprise with far-reaching impact in the physical sciences. Efficient numerical simulation of the quantum many-body problem has the potential to provide both quantitative and qualitative information on entire families of chemical and condensed matter systems which can be otherwise prohibitively expensive or even completely impractical to obtain by experiment. This dissertation focuses on the accurate modeling of electron behavior in molecular systems.

A fundamental challenge in modeling the behavior of electrons in the many-body Schrödinger equation is that the electronic wavefunction must be antisymmetric with respect to particle exchange. When the number of electrons grows, effective parametrization of the space of such wavefunctions becomes difficult. Deep learning techniques have recently impacted *ab initio* quantum chemistry by providing a new approach to the problem of tractable parameterization of high dimensional function spaces in quantum many-body problems. Over the past few years, a growing number of works [9, 45, 14, 43, 41, 23, 66, 28, 47, 13, 55] have demonstrated the use of neural networks in wavefunction approximation, with an increasing amount of importance placed on building symmetry constraints into models. In particular, several works [41, 23, 28, 47, 55] have recently applied neural networks to model antisymmetric wavefunctions.

In Chapter 1, we provide an abridged introduction to the parts of the many-electron problem which are salient to the variational Monte Carlo (VMC) [19, 22, 61, 5] technique. In particular, we discuss the variational nature of the ground state problem. We provide some simple pedagogical examples to give the reader a concrete foothold, and we revisit one of these examples in Chapter 4 during our overview of VMCNet [39], a general purpose VMC framework built on the JAX library [6]. We discuss the Monte Carlo and optimization techniques used to make the variational problem tractable.

In Chapter 2, we discuss the trial wavefunctions that we explore. We begin by describing the simplest ansatz for representing antisymmetric electronic wavefunctions, known as a Slater determinant. The optimization of this ansatz is the core of the Hartree-Fock (HF) method [59]. Conventionally, the representation power of the Slater determinant has been improved by including multiplicative Jastrow factors and transforming the particle coordinates via a so-called backflow transformation [18, 60], resulting in the Slater–Jastrow–backflow ansatz. While the Hartree Fock problem can be solved efficiently for a wide range of systems of interest using matrix diagonalization methods, the Slater–Jastrow and the Slater–Jastrow–backflow ansatzes are significantly more complicated and can in practice only be optimized using quantum Monte Carlo (QMC) techniques. For strongly correlated quantum systems, and even weakly correlated quantum systems when high accuracy is required, a linear combination of either a large number of Slater determinants[1] or a number of Slater–Jastrow–backflow ansatzes is needed to yield a sufficiently low, and therefore accurate, energy estimate.

Recently, these considerations have led to an active interest in leveraging neural networks to improve the construction of the backflow [41, 28, 47], the antisymmetry [23, 47], and the Jastrow factor [28] of these ansatzes. PauliNet [28] uses relatively small permutation equivariant neural networks for the backflow and invariant neural networks for the Jastrow factor. The backflow transformation in PauliNet is applied multiplicatively to the Hartree-Fock orbitals before the determinant layer is applied. The FermiNet work [47, 53], revealed around the same time as PauliNet, uses a more sophisticated equivariant backflow transformation with many more parameters. Another interesting and surprising feature of the FermiNet is that it eschews the Jastrow factor entirely. For a given system, the FermiNet often achieves lower energies than PauliNet [53]. Our discussion in Chapter 2 will focus on the FermiNet architecture.

While there has been some progress [24, 52, 33, 29, 3] in analyzing the expressiveness of the permutation equivariant mappings used in the backflow construction [67], the understanding of the effectiveness of the antisymmetric neural network layers remains limited [24, 29, 34]. Interestingly, Refs. [41, 47, 29] propose that a single FermiNet determinant could in theory achieve a universal representation of antisymmetric functions. However, these constructions are based on either a sorting process [41, 47] or an equivariant mapping that essentially encodes the entire wavefunction [29]. Both constructions yield discontinuous feature mappings when the ambient space dimension is larger than 1 or the number of particles is greater than 2, as opposed to the continuous neural network layers used in all works in the literature so far. Furthermore, in practice, the success of both PauliNet and FermiNet depends crucially on the quality of the permutation equivariant backflow. Therefore

[1]When these determinants are constructed from so-called excitations of a single reference determinant, this is called the configuration interaction (CI) method.

when the VMC energy is higher than the exact ground state energy, it is difficult to pin down the source of the error.

To address this issue, we consider wavefunction ansatzes which replace parts of the antisymmetric layer of FermiNet with explicitly antisymmetrized universal neural networks. The obvious drawback of this approach is that its computational cost increases factorially with respect to the number of electrons, and it can therefore only be applied to very small atoms and molecules. However, the use of explicit antisymmetrization can still allow us to better understand how the structure of the antisymmetric layer affects the overall performance.

We first consider a generic antisymmetric (GA) layer, which replaces the entire sum of products of determinants structure in FermiNet with an explicitly antisymmetrized feed forward neural network. When combined with the FermiNet backflow, the resulting FermiNet-GA architecture can achieve a universal representation of antisymmetric functions by construction.

In Chapter 3, we investigate the empirical performance of these neural network ansatzes. We find that the FermiNet-GA structure is empirically a highly expressive ansatz. For all systems studied, the error of the correlation energy is less than 1%, and is well below chemical accuracy (1 kcal/mol $\approx 1.6 \times 10^{-3}$ a.u.). On the other hand, at least from a practical perspective, we find that the so-called single-determinant FermiNet, which is in fact computed as a product of two determinants, is not expressive enough to represent electronic wavefunctions of interest.

To investigate further, we replace the product of determinants in the single-determinant FermiNet with a product of explicitly antisymmetrized feed forward neural networks, yielding the factorized antisymmetric neural network layer of rank 1 (FA-1). We find that FA-1 is not able to outperform single-determinant FermiNet, which suggests that the ineffectiveness of the single-determinant FermiNet is closely related to its product structure. To explore this further, we define a factorized antisymmetric neural network layer of rank K (FA-K), which generalizes the structure of K-determinant FermiNet. We find that FermiNet-FA-K does not outperform K-determinant FermiNet, which indicates that the sum-of-products structure is a key limiting feature of both architectures.

These results suggest that removing this sum-of-products structure may be a promising avenue towards developing an efficient antisymmetric layer that is more expressive than the original FermiNet architecture. We thus study a variant of the FermiNet called full determinant mode which replaces the products of determinants used in the FermiNet with single combined determinants. The full determinant construction is implemented in the JAX branch of the FermiNet repository [53], and is also mentioned in passing in the original FermiNet paper [47], but to our knowledge its performance has not been reported in the literature. We specifically investigate the full single-determinant FermiNet, which replaces the product of two determinants used in single-determinant FermiNet with a single combined determinant. Our numerical results show that full single-determinant FermiNet can close a large part of the gap between standard single-determinant FermiNet and the true ground state energy on small atomic and molecular problems. We further evaluate the performance of full single-determinant FermiNet on the nitrogen molecule at a dissociating bond length of 4.0 Bohr, a challenging strongly correlated system where the standard FermiNet architecture is not able to yield accurate results even with 64 determinants. To our great surprise, we find that the full single-determinant FermiNet can outperform the standard 64-determinant FermiNet on this system, and the error of the energy can be as small as 0.4 kcal/mol compared to the best available computational benchmark.

An important limitation of our numerical work is that for the multiple-determinant FermiNet, especially when pretraining is not used to initialize our training runs, we observed some run-to-run variance, where separate optimization runs would seemingly converge to differing final energy levels. In these cases we present the lowest energy attained, which represents the lowest energy wavefunction which we were able to obtain and reproduce from a small number of independent runs. While we unfortunately have only a very preliminary understanding of the optimization landscape for any given neural network architecture, a good characterization of the effects of statistical uncertainty due to the optimization is important for a careful application of quantum Monte Carlo techniques in any study of energy differences between different molecular geometries. If this uncertainty is affected by the geometry of the system in consideration, then the resulting physical predictions may be negatively affected.

In Chapter 4, we describe the repository underlying all of the numerical results discussed in this dissertation. We walk the reader through some of the structure and capabilities of the VMCNet [39] repository for enabling complex variational Monte Carlo calculations. We provide some involved examples of custom model construction and the flexibility of the VMCNet Python API for prototyping a full VMC calculation. We additionally demonstrate that results obtained using VMCNet are comparable to those published by [47, 53] using the FermiNet repository.

The reader should note that much of this dissertation, especially this Preface, the later sections of Chapter 1, most of Chapters 2 and 3, and the benchmarking section of Chapter 4, are closely derived in content and organization from a joint work with Gil Goldshlager and Lin Lin [38].

Chapter 1

Variational Monte Carlo

In this chapter, we introduce the many-body electronic structure problem and discuss some basic pedagogical examples. We explore how Monte Carlo techniques are used to estimate values of the high dimensional integrals that arise from the calculation of expectation of observable quantities, and derive the reduction of the full electronic wavefunction problem to an equivalent problem involving a spatial wavefunction. We conclude with a brief discussion of optimization techniques used to find the ground state.

Put together with a parametrization of the electronic wavefunction space, these techniques are used to find the ground state in a method known as **variational Monte Carlo** (VMC) (see Fig. 1.1). A number of excellent references, should the reader desire a more traditional and in-depth discussion of the topic, are available at [19, 22, 61, 5].

1.1 The Schrödinger equation

In non-relativistic quantum mechanics, one studies the **time-dependent Schrödinger equation**. For a particular choice of unit length and a symmetric[1] operator $\hat{H}$, this is given by the partial differential equation

$$i\frac{\partial \Psi}{\partial t}(\mathbf{X}, t) = \hat{H}\Psi(\mathbf{X}, t). \tag{1.1}$$

Here $\mathbf{X}$ is generally a mixture of continuous and discrete degrees of freedom, such as spatial degrees of freedom (e.g. a copy of $\mathbb{R}^3$ for each particle being modeled) and

[1]Here the term symmetric is being used in the typical functional analysis sense, i.e. $(\Phi, \hat{H}\Psi) = (\hat{H}\Phi, \Psi)$. Physicists use the term **Hermitian**, which encompasses both symmetric and self-adjoint (and essentially self-adjoint) operators, and aligns with the finite dimensional terminology.

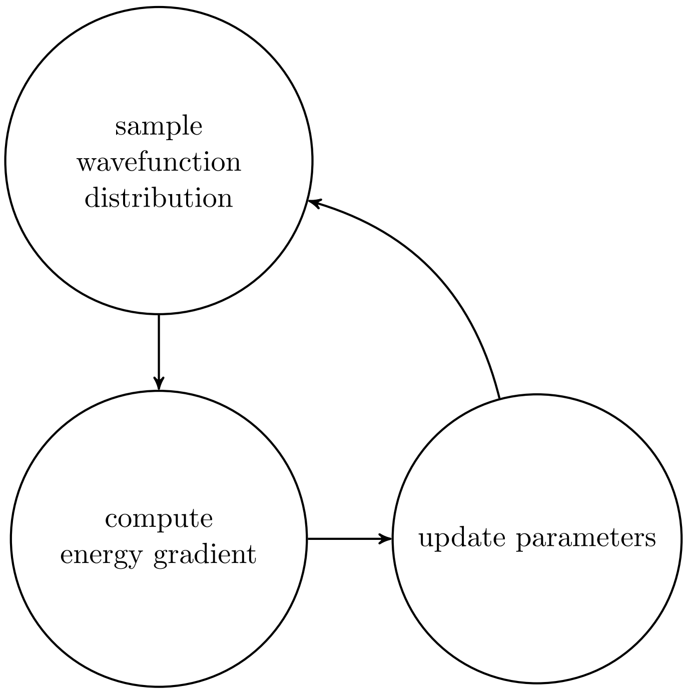


Figure 1.1: A simplified overview of the variational Monte Carlo method.

spin degrees of freedom (e.g. $\{\uparrow, \downarrow\}$ for electron spin). The object Ψ is a member of a Hilbert space with inner product

$$(\Phi, \Psi) = \int \Phi(\mathbf{X})^* \Psi(\mathbf{X}) \, d\mathbf{X}. \tag{1.2}$$

The regularity properties of admissible Ψ are generally determined by the choice of $\hat{H}$, known as the **Hamiltonian**. Quantum mechanics postulates that such a vector Ψ, known as a **wavefunction**, is to be interpreted as the (sometimes unnormalized[2]) square root of a probability density $p = |\Psi|^2 / \|\Psi\|^2$.

When the operator $\hat{H}$ is constant in time, then the solutions to the time-dependent Schrödinger equation are determined by the solutions to the **time-independent Schrödinger equation**, which is the eigenvalue problem

$$\hat{H}\Psi(\mathbf{X}) = E\Psi(\mathbf{X}), \tag{1.3}$$

[2]The physical states themselves are postulated to be **rays**, or members of a projective Hilbert space arising from the equivalence classes of a separable complex Hilbert space which contains the objects Ψ. Topological separability gives the mathematical convenience of a countable basis.

for eigenvalue (**energy**) E. The solutions to this equation give the stationary states of the time-dependent Schrödinger equation. The state coresponding to the lowest eigenvalue E is known as the **ground state**, and the other states are known as the **excited states**. Understanding the properties of these eigenstates is key to understanding chemical and physical properties of a quantum system.

To extend the above definition of energy to all wavefunctions Ψ, the above eigenvalue problem can be understood variationally in terms of the Rayleigh quotient:

$$E(\Psi) := \frac{(\Psi, \hat{H}\Psi)}{\|\Psi\|^2}. \tag{1.4}$$

In terms of the Rayleigh quotient $E(\Psi)$, the ground state is the variational minimizer among the admissible non-zero wavefunctions. The variational formulation allows us to understand the problem of finding the ground state as risk minimization (in the language of statistical machine learning). Importantly, $E(\Psi)$ is always an upper bound to the exact ground state energy.

More generally, given a symmetric operator $\hat{O}$ on the Hilbert space of wavefunctions, one can define the expectation $\langle \hat{O} \rangle$ as

$$\langle \hat{O} \rangle = \frac{(\Psi, \hat{O}\Psi)}{\|\Psi\|^2}. \tag{1.5}$$

If a $\{\Psi_0, \Psi_1, \ldots\}$ is an orthonormal eigenbasis corresponding to a discrete spectrum $\{E_0, E_1, \ldots\}$ of $\hat{O}$, then a normalized wavefunction

$$\Psi = \frac{1}{\sqrt{\sum_i |c_i|^2}} \sum_i c_i \Psi_i \tag{1.6}$$

defines a probability mass function $p(E_i) = |c_i|^2 / \sum_i |c_i|^2$. In this sense $\langle \hat{O} \rangle$ can be understood as a usual statistical expectation:

$$\langle \hat{O} \rangle = \sum_i E_i \, p(E_i). \tag{1.7}$$

In Section 1.3 we will discuss how in some cases $\langle \hat{O} \rangle$ can be understood as an expectation of a local observable over a continuous probability density.

Single-particle physics

A useful prototypical problem is to consider a single quantum particle in space[3] with coordinates $\mathbf{r} \in \mathbb{R}^d$ and corresponding wavefunction $\Psi(\mathbf{r})$, subject to the following

[3]Usually, the dimension d of interest to condensed-matter physicists or chemists is 1, 2, or 3.

Hamiltonian (for a particular choice of units):

$$\hat{H}\Psi(\mathbf{r}) = -\frac{1}{2}\Delta_{\mathbf{r}}\Psi(\mathbf{r}) + V(\mathbf{r})\Psi(\mathbf{r}), \tag{1.8}$$

where $V(\mathbf{r})$ is a potential field.

Example 1. A classic introductory example to single-particle quantum physics is that of the $d = 1$ quantum harmonic oscillator, described by the potential

$$V(r) = \frac{1}{2}\omega^2 r^2 \tag{1.9}$$

for some constant ω. By using a standard analytic method like the Frobenius method or a spectral decomposition, one can find that the eigenfunctions of $\hat{H}$ are given by

$$\Psi_n(r) \propto e^{-\frac{1}{2}\omega r^2} H_n(\sqrt{\omega} r), \tag{1.10}$$

where H_n is the Hermite polynomial

$$H_n(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n}\left(e^{-x^2}\right). \tag{1.11}$$

The corresponding energies are given by

$$E_n = \omega\left(n + \frac{1}{2}\right). \tag{1.12}$$

These eigenfunctions are depicted in Fig. 1.2.

Example 2. For $d > 1$, another interesting single-particle system of note is that of the hydrogen-like atom. The potential is given by the Coulomb potential

$$V(\mathbf{r}) = -\frac{Z}{|\mathbf{r}|}, \tag{1.13}$$

where $Z > 0$ is a constant and $|\mathbf{r}|$ denotes the euclidean distance of $\mathbf{r}$ from the origin. For this problem, the ground state wavefunction is given by

$$\Psi_0(\mathbf{r}) \propto \exp\left(-\frac{2Z|\mathbf{r}|}{d-1}\right). \tag{1.14}$$

One can compute the ground state energy to be

$$E_0 = -2\left(\frac{Z}{d-1}\right)^2. \tag{1.15}$$

In the special case of $d = 3$ (the hydrogen atom), this gives the ground state wavefunction $\Psi_0(\mathbf{r}) \propto \exp(-Z|\mathbf{r}|)$ with energy $E_0 = -Z^2/2$.

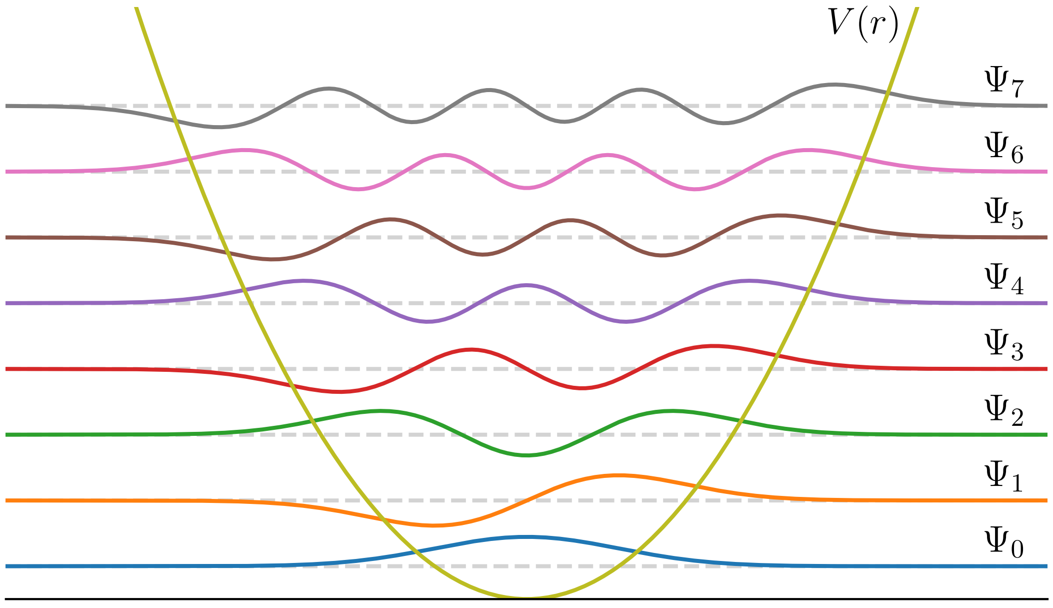


Figure 1.2: Eigenfunctions of the quantum harmonic oscillator (Ψ_n has been plotted at height E_n).

Non-interacting identical particles

When a quantum system has more than one particle, there are often non-spatial degrees of freedom which are relevant in order to describe the behavior of the system (often discrete rather than continuous, e.g. electron spin). Thus, for example, the many-body wavefunction may have inputs $\mathbf{X} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_N)$, where $\mathbf{x}_i \equiv (\mathbf{r}_i, \sigma_i)$ for $\sigma_i \in \{\uparrow, \downarrow\}$. However, the kinetic energy due to the particles only depends on the derivatives with respect to the spatial coordinates $\mathbf{R} \equiv (\mathbf{r}_1, \ldots, \mathbf{r}_N)$.

If these particles are subject to identical potentials, one can construct a system of independent identical particles by simply adding up single-particle terms of the form Eq. (1.8), resulting in a Hamiltonian such as

$$\hat{H}\Psi(\mathbf{X}) = -\frac{1}{2}\Delta_{\mathbf{R}}\Psi(\mathbf{X}) + \sum_{i=1}^{N} V(\mathbf{r}_i)\Psi(\mathbf{X}). \tag{1.16}$$

Without any additional constraints (e.g. symmetry requirements) on the wavefunction space, the resulting ground state is simply the N-fold product of the single-particle ground state. However, as we will discuss in the next section, the electronic wavefunction does not take this form due to the presence of antisymmetry constraints.

1.2 Ab-initio quantum chemistry

In chemistry, a typical model used to study the (zero-temperature) bonding behavior of molecules of interest is to fix the nuclei as points which provide a classical Coulomb potential as a term in the Hamiltonian. In this regime, known as the **Born-Oppenheimer approximation**, one is interested in the many-body electronic wavefunction, or the **electronic structure** of the system. This approximation can then be used to compute the **potential energy surface** of a system, which is the ground state energy surface as a function of the nuclear positions.

To define the electronic structure problem, it remains to specify the form of the operator $\hat{H}$. The electronic wavefunctions of a fixed number of electrons N are functions of electron position and spin, $\mathbf{x}_i = (\mathbf{r}_i, \sigma_i) \in \mathbb{R}^3 \times \{\uparrow, \downarrow\}$ for electron index $1 \leq i \leq N$. Many-body electron wavefunctions are postulated to be **antisymmetric**[4] with respect to the electron particle index. This means that for a wavefunction of N electrons, specified by $\Psi(\mathbf{x}_1, \ldots, \mathbf{x}_N) \in \mathbb{C}$, we require that

$$\Psi(\mathbf{x}_{\pi(1)}, \ldots, \mathbf{x}_{\pi(N)}) = \operatorname{sgn}(\pi)\Psi(\mathbf{x}_1, \ldots, \mathbf{x}_N), \tag{1.17}$$

where π is a member of the permutation group on N elements (S_N). The function $\operatorname{sgn}(\pi) \in \{1, -1\}$ is the **sign**, **signature**, or **parity** of π, given by the alternating character or sign representation of the symmetric group. The representation sgn partitions S_N into even ($\operatorname{sgn}(\pi) = 1$) and odd ($\operatorname{sgn}(\pi) = -1$) permutations.

The electronic Hamiltonian under consideration is an operator on the antisymmetric states with the form

$$\hat{H} = \hat{T} + \hat{V}, \tag{1.18}$$

where $\hat{T}$ is a kinetic part and $\hat{V}$ is a potential part. The exact form of the kinetic and potential part depend on the application. In ab-initio chemical applications the systems of interest are often described by a Hamiltonian whose kinetic part is simply proportional to the Laplacian operator, and whose potential part is determined by a classical local Coulomb potential field given by point charges which are fixed at the locations of atomic nuclei, known as the **Born-Oppenheimer approximation**. If the atomic nuclei are at $\{\mathbf{R}_I^{(a)}\}$ and the electron positions are at $\{\mathbf{r}_i\}$, then (in **atomic units**) the Born-Oppenheimer Hamiltonian is given by:

$$H = \hat{T} + \hat{V}_{\text{ei}} + \hat{V}_{\text{ee}} + E_{\text{II}}, \tag{1.19}$$

[4]More accurately, only particles with symmetric (**bosonic**) and antisymmetric (**fermionic**) wavefunctions have been experimentally observed, and the statistical behavior of electrons is consistent with that of the antisymmetric wavefunctions.

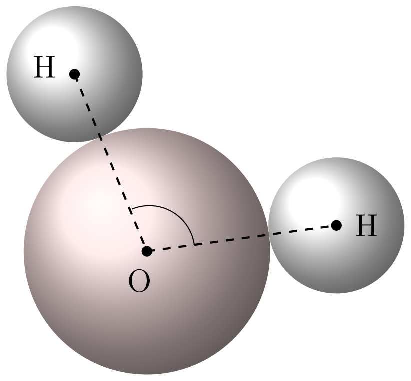


Figure 1.3: A diagram of a water molecule. There are three atomic nuclei, each modeled as a point charge. By varying the bond angle and bond length and computing the resulting ground state energy, one can find a lowest energy configuration.

where the Hamiltonian is partitioned into the kinetic, electron-ion potential, electron-electron, and ion-ion interaction, respectively:

$$\begin{aligned} \hat{T} &= \sum_{i=1}^{N} -\frac{1}{2}\Delta_{\mathbf{r}_i}, \quad \hat{V}_{\text{ei}} = -\sum_{i=1}^{N}\sum_{I=1}^{M} \frac{Z_I}{\left|\mathbf{r}_i - \mathbf{R}_I^{(a)}\right|}, \\ \hat{V}_{\text{ee}} &= \sum_{i<j}^{N} \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|}, \quad E_{\text{II}} = \sum_{I<J}^{M} \frac{Z_I Z_J}{\left|\mathbf{R}_I^{(a)} - \mathbf{R}_J^{(a)}\right|}. \end{aligned} \tag{1.20}$$

For a given atomic configuration, the ion-ion interaction simply adds a constant shift to the Hamiltonian.

Example 3. Consider the example of a water molecule (Fig. 1.3). Under the Born-Oppenheimer approximation, this corresponds to $N = 10$ electrons subject to the potentials created by the charges $Z_1 = Z_2 = 1$ and $Z_3 = 8$. We furthermore consider only configurations in which the two hydrogen nuclei are equidistant from the oxygen nucleus, i.e.

$$\begin{aligned} \text{bond length} &= \left|\mathbf{R}_1^{(a)} - \mathbf{R}_3^{(a)}\right| = \left|\mathbf{R}_2^{(a)} - \mathbf{R}_3^{(a)}\right|, \text{ and} \\ \text{bond angle} &= \arccos\left(\frac{\left(\mathbf{R}_1^{(a)} - \mathbf{R}_3^{(a)}\right)\cdot\left(\mathbf{R}_2^{(a)} - \mathbf{R}_3^{(a)}\right)}{\left|\mathbf{R}_1^{(a)} - \mathbf{R}_3^{(a)}\right|\left|\mathbf{R}_2^{(a)} - \mathbf{R}_3^{(a)}\right|}\right). \end{aligned} \tag{1.21}$$

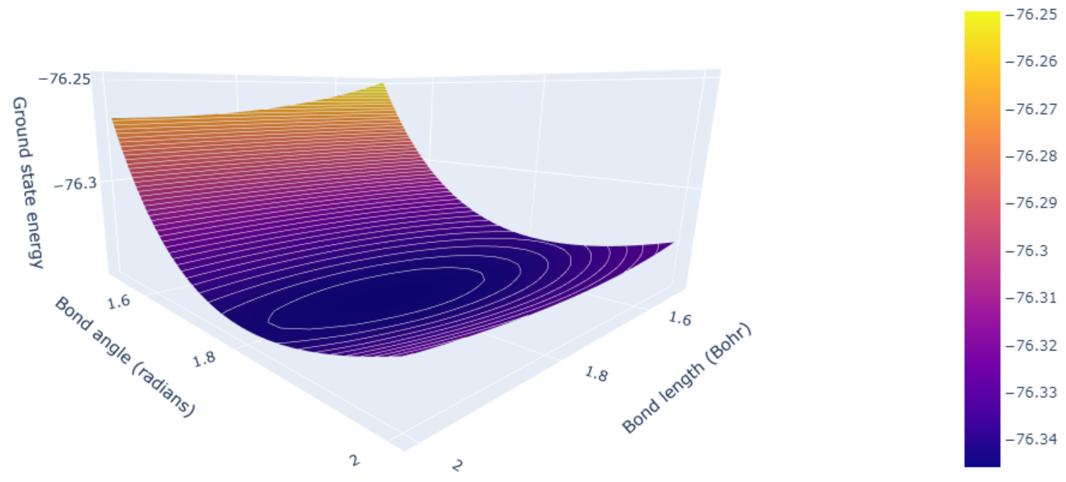


Figure 1.4: Estimated potential energy surface of a water molecule with varying bond angle and length.

Given a way to compute accurate ground state energies of this molecule for various configurations, one can compute a potential energy surface as a function of the bond angle and bond length. Using coupled-cluster methods[5], we can estimate the potential energy surface as in Fig. 1.4.

1.3 Monte Carlo integration

Directly computing the Rayleigh quotient in Eq. (1.4) for the Born-Oppenheimer $\hat{H}$ requires computing two expectations which are integrals over the positions in $\mathbb{R}^{3N}$ and sums over the permutations of the spin configuration. As N grows, the growing dimensionality of these expectations renders practical use of any traditional quadrature method essentially impossible. Instead, one can estimate $\langle \hat{O} \rangle$ by appealing to Monte Carlo integration techniques. This is done by defining a notion of a **local observable** for which the expected value under the distribution $p(\mathbf{X}) \propto |\Psi(\mathbf{X})|^2$ is

[5]In this case, we used the CCSD(T) method via the PySCF package in the cc-pVTZ orbital basis.

precisely $\langle \hat{O} \rangle$. Indeed, let the local observable be given by

$$\hat{O}_L(\mathbf{X}) \equiv \frac{\hat{O}\Psi(\mathbf{X})}{\Psi(\mathbf{X})}. \tag{1.22}$$

Then

$$\int \hat{O}_L(\mathbf{X}) p(\mathbf{X})\, d\mathbf{X} = \int \hat{O}_L(\mathbf{X}) \frac{|\Psi(\mathbf{X})|^2}{\|\Psi(\mathbf{X})\|^2}\, d\mathbf{X} = \langle \hat{O} \rangle. \tag{1.23}$$

If we draw a set of $n \gg 1$ independent and identically distributed samples ξ from $p(\mathbf{X})$, then following the law of large numbers, the integral in Eq. (1.23) can be estimated as

$$\langle \hat{O} \rangle \approx \frac{1}{n} \sum_{\mathbf{X} \in \xi} \hat{O}_L(\mathbf{X}). \tag{1.24}$$

If $\hat{O}_L$ has a finite second moment, then we may additionally appeal to the central limit theorem: as the number of samples n grows, we expect the distribution of the estimated mean to approach a normal distribution with variance equal to the variance of $\hat{O}_L$ divided by $\sqrt{n}$.

In particular, the **local energy**,

$$E_L(\mathbf{X}) = \frac{\hat{H}\Psi(\mathbf{X})}{\Psi(\mathbf{X})}, \tag{1.25}$$

can be understood as a loss function for the ground state problem. By definition the local energy will be constant in $\mathbf{X}$ when Ψ is an eigenfunction of $\hat{H}$. It follows that the variance of the local energy goes to zero as Ψ approaches an eigenfunction[6], known as the **zero-variance principle** [15]. This principle is a key feature of variational approaches to the ground state search problem, as it promises that the statistical quality of the energy estimation via expectation of the local energy improves as Ψ approaches the ground state wavefunction.

Importantly, the local energy (and other local observables) can be computed even when the wavefunction Ψ is unnormalized. Thus to compute the estimated expectation, it remains to acquire samples ξ from the distribution $p(\mathbf{X}) \propto |\Psi(\mathbf{X})|^2$. In order to sample from an unnormalized distribution, a common technique is **Metropolis-Hastings Monte Carlo**[7]. The two ingredients for Metropolis-Hastings Monte Carlo

[6]Here "approaches" is used a bit freely – some regularity is generally needed for this to hold, as the Laplacian is unbounded when the domain contains all square-integrable functions.

[7]This is a special case of the general **Markov chain Monte Carlo** technique, a powerful class of algorithms that create a Markov chain which has the target distribution as its stationary distribution.

are: a **proposal** distribution $g(\mathbf{X}' \mid \mathbf{X})$ which gives the conditional probability of proposing $\mathbf{X}'$ given $\mathbf{X}$, and a target distribution $p(\mathbf{X})$. Given these two ingredients, one defines the (Metropolis-Hastings) **acceptance ratio** (or acceptance probability):

$$A(\mathbf{X}', \mathbf{X}) = \min\left(\frac{p(\mathbf{X}')g(\mathbf{X} \mid \mathbf{X}')}{p(\mathbf{X})g(\mathbf{X}' \mid \mathbf{X})}, 1\right). \tag{1.26}$$

In the special case when g is a **symmetric proposal**, i.e. $g(\mathbf{X}' \mid \mathbf{X}) = g(\mathbf{X} \mid \mathbf{X}')$ for all $\mathbf{X}, \mathbf{X}'$, then the acceptance ratio is particularly simple:

$$A(\mathbf{X}', \mathbf{X}) = \min\left(\frac{p(\mathbf{X}')}{p(\mathbf{X})}, 1\right). \tag{1.27}$$

Then the algorithm proceeds as follows: starting from an initial state $\mathbf{X}_0$, one computes $\mathbf{X}_n$ from $\mathbf{X}_{n-1}$ by *proposing* a new state $\mathbf{X}'_{n-1}$ under the conditional distribution $g(\mathbf{X}'_{n-1} \mid \mathbf{X}_{n-1})$ and *accepting* the move with probability $A(\mathbf{X}'_{n-1}, \mathbf{X}_{n-1})$. That is, we make the transition $\mathbf{X}_n = \mathbf{X}_{n-1}$ with probability $1 - A(\mathbf{X}'_{n-1}, \mathbf{X}_{n-1})$ and we instead make the transition $\mathbf{X}_n = \mathbf{X}'_{n-1}$ with probability $A(\mathbf{X}'_{n-1}, \mathbf{X}_{n-1})$. The result of the Metropolis-Hastings algorithm is to create a Markov chain $\mathbf{X}_0, \mathbf{X}_1, \ldots$ which converges to the stationary distribution $p(\mathbf{X})$.

One must be careful when using Metropolis-Hastings Monte Carlo to take into account that the resulting chain is autocorrelated, which must be quantified in order to compute an accurate estimate of the standard error when using the samples for numerical integration. Specifically, if M chains of length n are obtained (usually by running many Markov chains in parallel), and $\rho_{t,m}$ gives the autocorrelation estimate at lag t for chain m, then we follow the Stan package [54] to combine the autocorrelation at lag t into a multi-chain estimate $\hat{\rho}_t$ and compute an effective sample size of

$$n_{\text{effective}} = \frac{M \cdot n}{\hat{\tau}}, \tag{1.28}$$

where $\hat{\tau}$ is the integrated autocorrelation

$$\hat{\tau} = 1 + 2 \sum_{t=1}^{2M_{\text{cutoff}}+1} \hat{\rho}_t \tag{1.29}$$

for some cutoff point M_{cutoff} chosen following Geyer's initial monotone sequence criterion [21]. We refer the reader to the Stan reference manual [54] and the VMCNet package [39] for details on how $\hat{\rho}_t$ is computed and how M_{cutoff} is chosen.

Spin-independent observables

The molecular Hamiltonian does not depend explicitly on electron spin, and so if $N_\uparrow$ and $N_\downarrow$ are fixed (i.e. $\Psi(\mathbf{X}) = 0$ for any electron configuration with a different number of spin-up or spin-down electrons), the antisymmetry constraint in Eq. (1.17) over the spatial-spin electron configuration can be rewritten using a spin-independent wavefunction $\Psi(\mathbf{R})$, which reduces the number of degrees of freedom in the wavefunction and improves the efficiency in the resulting Monte Carlo calculations [19]. Below, we derive this equivalence between the eigenstate problem for the complete wavefunction $\Psi(\mathbf{X})$ and a spatial wavefunction $\Psi(\mathbf{R})$.

Derivation. Let $\hat{O}$ be any totally symmetric spin-independent operator (e.g. electron density, Hamiltonian), and assume we are interested in expectation values of $\hat{O}$ with respect to the many body wavefunction $\Psi(\mathbf{X})$:

$$\langle \hat{O} \rangle = \frac{\sum_\sigma \int \Psi^*(\mathbf{X}) \hat{O}(\mathbf{R}) \Psi(\mathbf{X}) \, \mathrm{d}\mathbf{R}}{\sum_\sigma \int \Psi^*(\mathbf{X}) \Psi(\mathbf{X}) \, \mathrm{d}\mathbf{R}}. \tag{1.30}$$

For each spin-configuration $\sigma = (\sigma_1, \ldots, \sigma_N)$, we can always permute the particle index $1, \ldots, N$ so that the spin-up indices appear in front of the spin-down indices. The integrals in the numerator and denominator of Eq. (1.30) are independent of such a permutation operation. Therefore we may define the spatial wavefunction

$$\Psi(\mathbf{R}) = \Psi((\mathbf{r}_1, \uparrow), \ldots, (\mathbf{r}_{N_\uparrow}, \uparrow), (\mathbf{r}_{N_\uparrow+1}, \downarrow), \ldots, (\mathbf{r}_{N_\uparrow+N_\downarrow}, \downarrow)). \tag{1.31}$$

By renaming the integration variables,

$$\begin{aligned} \langle \hat{O} \rangle =& \frac{\sum_\sigma \int \Psi^*(\mathbf{R}) \hat{O}(\mathbf{R}) \Psi(\mathbf{R}) \, \mathrm{d}\mathbf{R}}{\sum_\sigma \int \Psi^*(\mathbf{R}) \Psi(\mathbf{R}) \, \mathrm{d}\mathbf{R}} \\ =& \frac{\int \Psi^*(\mathbf{R}) \hat{O}(\mathbf{R}) \Psi(\mathbf{R}) \, \mathrm{d}\mathbf{R}}{\int \Psi^*(\mathbf{R}) \Psi(\mathbf{R}) \, \mathrm{d}\mathbf{R}}. \end{aligned}$$

We further introduce the notation $\mathbf{R}^\uparrow \equiv (\mathbf{r}_1^\uparrow, \ldots, \mathbf{r}_{N_\uparrow}^\uparrow) \equiv (\mathbf{r}_1, \ldots, \mathbf{r}_{N_\uparrow})$ and $\mathbf{R}^\downarrow \equiv (\mathbf{r}_1^\downarrow, \ldots, \mathbf{r}_{N_\downarrow}^\downarrow) \equiv (\mathbf{r}_{N_\uparrow+1}, \ldots, \mathbf{r}_{N_\uparrow+N_\downarrow})$. Then the constraint that $\Psi(\mathbf{X})$ is antisymmetric with respect to the action of S_N is equivalent to the requirement that

$$\Psi(\mathbf{R}) := \Psi(\mathbf{R}^\uparrow, \mathbf{R}^\downarrow) \tag{1.32}$$

is an antisymmetric function with respect to $\mathbf{R}^\uparrow$ and $\mathbf{R}^\downarrow$ separately, but not necessarily antisymmetric across the spin-up and spin-down indices. We can thus work with $\Psi(\mathbf{R})$ instead of $\Psi(\mathbf{X})$ as long as we enforce this symmetry. The expectation over $p(\mathbf{X})$ instead becomes an expectation of $\hat{O}_L(\mathbf{R})$ over the continuous density $p(\mathbf{R}) \propto |\Psi(\mathbf{R})|^2$. $\square$

If needed, the original antisymmetric function $\Psi(\mathbf{X})$ can be recovered by **antisymmetrizing** the following spin-dependent function:

$$\Psi(\mathbf{X}) = \mathcal{A}(\Psi(\mathbf{R})\delta_{\sigma_1,\uparrow} \dots \delta_{\sigma_{N_\uparrow},\uparrow}\delta_{\sigma_{N_\uparrow+1},\downarrow} \cdots \delta_{\sigma_N,\downarrow}), \tag{1.33}$$

where $\delta_{\sigma,\rho}$ is the Kronecker delta function on the spin components given by

$$\delta_{\sigma,\rho} = \begin{cases} 1 & \sigma = \rho \\ 0 & \sigma \neq \rho \end{cases}. \tag{1.34}$$

The antisymmetrization operator $\mathcal{A}$ simply enforces the complete antisymmetry with respect to the particle index, by summing over all permutations of the particle index:

$$\mathcal{A}(F(\mathbf{X})) \equiv \frac{1}{N!} \sum_{\pi \in S_N} \operatorname{sgn}(\pi) F(\pi(\mathbf{X})), \tag{1.35}$$

where $\pi(\mathbf{X})$ is the usual representation $\pi(\mathbf{X}) = (\mathbf{x}_{\pi(1)}, \dots, \mathbf{x}_{\pi(N)})$. Because the antisymmetry in $F(\mathbf{X}) = \Psi(\mathbf{R})\delta_{\sigma_1,\uparrow} \dots \delta_{\sigma_{N_\uparrow},\uparrow}\delta_{\sigma_{N_\uparrow+1},\downarrow} \cdots \delta_{\sigma_N,\downarrow}$ is already enforced for permutations in $S_{N_\uparrow} \times S_{N_\downarrow}$, only N choose $N_\uparrow$ permutations of spin configurations need to be considered when recovering $\Psi(\mathbf{X})$ from $\Psi(\mathbf{R})$.

1.4 Optimization

When carrying out the variational Monte Carlo (VMC) algorithm, the strategy for solving the molecular many-body problem is to parametrize a family of wavefunctions $\Psi_\theta \in L^2$ on some parameter domain (usually $\mathbb{R}^m$, where m is the number of parameters) and combine the parameterization with the previously described Monte Carlo estimation. This gives a parametrization of the Rayleigh quotient objective function of the resulting approximate eigenproblem, and then techniques from stochastic optimization are used to minimize the objective in the parameter space. Explicitly, this is done by drawing a set of samples ξ_θ from $p_\theta(\mathbf{R})$ using Markov chain Monte Carlo and estimating the expected loss (risk) as

$$\mathcal{L}(\theta) = E(\Psi_\theta) \approx \frac{1}{|\xi_\theta|} \sum_{\mathbf{R} \in \xi_\theta} E_L(\mathbf{R}; \theta) = \widetilde{\mathcal{L}}(\theta), \tag{1.36}$$

where we have defined the objective function $\mathcal{L}(\theta)$ and its stochastic estimate $\widetilde{\mathcal{L}}$. In statistical machine learning parlance, $\widetilde{\mathcal{L}}$ is known as the empirical risk, and $\mathcal{L}$ is known as the true or population risk.

The capacity of the trial wavefunction family Ψ_θ to approximate the true ground state is key to the success of the variational algorithm. The discussion of the trial wavefunction form is left to Chapter 2.

Gradient calculation

When computing parameter updates, estimating the gradient of $\mathcal{L}(\theta)$ by directly differentiating the empirical risk $\tilde{\mathcal{L}}(\theta)$ defined in Eq. (1.36) using an automatic differentiation framework is generally difficult due to the dependence of the Monte Carlo sampling on the parameters θ. However, the following standard unbiased estimate of the gradient of the true expected energy $E(\Psi_\theta) \equiv \mathcal{L}(\theta)$ is available for real wavefunctions [7]:

$$\begin{aligned}\partial_\theta \mathcal{L}(\theta) &= \frac{\int 2(\partial_\theta \log|\Psi_\theta|)(E_L(\mathbf{R};\theta) - \mathcal{L}(\theta))\,|\Psi_\theta|^2 \, d\mathbf{R}}{\int |\Psi_\theta|^2 \, d\mathbf{R}} \\ &\approx \frac{1}{|\xi_\theta|} \sum_{\mathbf{R}\in\xi_\theta} 2(\partial_\theta \log|\Psi_\theta|)(E_L(\mathbf{R};\theta) - \tilde{\mathcal{L}}(\theta))\end{aligned} \tag{1.37}$$

where ξ_θ are a set of samples from the density $p_\theta(\mathbf{R}) = |\Psi_\theta(\mathbf{R})|^2 / \int |\Psi_\theta|^2 \, d\mathbf{R}$. The derivation is as follows:

Derivation. Recall that the expected energy is given by

$$\mathcal{L}(\theta) = \frac{\int E_L(\mathbf{R};\theta)\,|\Psi_\theta|^2 \, d\mathbf{R}}{\int |\Psi_\theta|^2 \, d\mathbf{R}}. \tag{1.38}$$

For the purposes of the following derivation, we will let θ be a real number representing any parameter. The derivative of the integrand in the denominator with respect to θ is

$$\partial_\theta |\Psi_\theta|^2 = (\partial_\theta \Psi_\theta)\Psi_\theta^* + \Psi_\theta(\partial_\theta \Psi_\theta^*) = \left(\frac{\partial_\theta \Psi_\theta}{\Psi_\theta} + \frac{\partial_\theta \Psi_\theta^*}{\Psi_\theta^*}\right)|\Psi_\theta|^2, \tag{1.39}$$

and the derivative of the integrand in the numerator with respect to θ is

$$\begin{aligned}\partial_\theta \left[E_L(\mathbf{R};\theta)\,|\Psi_\theta|^2\right] &= \partial_\theta \left[\Psi_\theta^* H \Psi_\theta\right] \\ &= \partial_\theta \Psi_\theta^* H \Psi_\theta + \Psi_\theta^* H \partial_\theta \Psi_\theta \\ &= \frac{\partial_\theta \Psi_\theta^*}{\Psi_\theta^*} E_L(\mathbf{R};\theta)\,|\Psi_\theta|^2 + \Psi_\theta^* H \partial_\theta \Psi_\theta\end{aligned} \tag{1.40}$$

To treat the latter term in the last expression above, we also take advantage of the following identity, which uses the essential self-adjointness of the Born-Oppenheimer Hamiltonian H [49]:

$$\int \Psi_\theta^*(H\partial_\theta\Psi_\theta)\,d\mathbf{R} = \int \partial_\theta\Psi_\theta(H\Psi_\theta^*)\,d\mathbf{R} = \int \frac{\partial_\theta\Psi_\theta}{\Psi_\theta} E_L(\mathbf{R};\theta)^*\,|\Psi_\theta|^2 \, d\mathbf{R} \tag{1.41}$$

Using these facts, the following calculation gives the derivative of the expected energy $\mathcal{L}$ with respect to θ:

$$\begin{aligned}\partial_\theta \mathcal{L}(\theta) &= \partial_\theta \frac{\int E_L(\mathbf{R};\theta)\,|\Psi_\theta|^2\,d\mathbf{R}}{\int |\Psi_\theta|^2\,d\mathbf{R}} \\ &= \frac{\partial_\theta \int E_L(\mathbf{R};\theta)\,|\Psi_\theta|^2\,d\mathbf{R}}{\int |\Psi_\theta|^2\,d\mathbf{R}} - \frac{\int E_L(\mathbf{R};\theta)\,|\Psi_\theta|^2\,d\mathbf{R}}{\int |\Psi_\theta|^2\,d\mathbf{R}} \frac{\partial_\theta \int |\Psi_\theta|^2\,d\mathbf{R}}{\int |\Psi_\theta|^2\,d\mathbf{R}} \\ &= \frac{\int ((\partial_\theta \Psi_\theta^*/\Psi_\theta^*) E_L(\mathbf{R};\theta) + (\partial_\theta \Psi_\theta/\Psi_\theta) E_L(\mathbf{R};\theta)^*)\,|\Psi_\theta|^2\,d\mathbf{R}}{\int |\Psi_\theta|^2\,d\mathbf{R}} \\ &\quad - \frac{\int E_L(\mathbf{R};\theta)\,|\Psi_\theta|^2\,d\mathbf{R}}{\int |\Psi_\theta|^2\,d\mathbf{R}} \frac{\int (\partial_\theta \Psi_\theta^*/\Psi_\theta^* + \partial_\theta \Psi_\theta/\Psi_\theta)\,|\Psi_\theta|^2\,d\mathbf{R}}{\int |\Psi_\theta|^2\,d\mathbf{R}} \\ &= \frac{\int [(\partial_\theta \Psi_\theta^*/\Psi_\theta^*)(E_L(\mathbf{R};\theta) - \mathcal{L}(\theta)) + (\partial_\theta \Psi_\theta/\Psi_\theta)(E_L(\mathbf{R};\theta)^* - \mathcal{L}(\theta))]\,|\Psi_\theta|^2\,d\mathbf{R}}{\int |\Psi_\theta|^2\,d\mathbf{R}}.\end{aligned} \tag{1.42}$$

When the wavefunction is real, we may simplify this to

$$\partial_\theta \mathcal{L}(\theta) = \frac{\int 2(\partial_\theta \Psi_\theta/\Psi_\theta)(E_L(\mathbf{R};\theta) - \mathcal{L}(\theta))\,|\Psi_\theta|^2\,d\mathbf{R}}{\int |\Psi_\theta|^2\,d\mathbf{R}}. \tag{1.43}$$

We may then use Monte Carlo sampling to estimate the gradient as

$$\partial_\theta \mathcal{L}(\theta) \approx \frac{1}{|\xi_\theta|} \sum_{\mathbf{R}\in\xi_\theta} 2(\partial_\theta \log|\Psi_\theta|)(E_L(\mathbf{R};\theta) - \tilde{\mathcal{L}}(\theta)) \tag{1.44}$$

where ξ_θ are a set of samples from the density $p_\theta(\mathbf{R}) = |\Psi_\theta(\mathbf{R})|^2 / \int |\Psi_\theta(\mathbf{R})|^2\,d\mathbf{R}$. $\square$

The zero-variance principle can also make the variance of the local energy an attractive target for minimization. Indeed, variance minimization has an extensive history in the quantum Monte Carlo space [64, 32, 62]. Nonetheless, in Chapter 3 we follow the work of FermiNet [47, 53] and only use energy minimization to optimize our wavefunctions.

Optimizer

The choice of efficient optimization algorithms for parameter updates in variational Monte Carlo has historically been a complex issue and is still under active debate (see e.g. [65, 44, 46, 5, 9, 51, 55, 47]). Among these works, Ref. [47] provided evidence that the use of the Kronecker Factorized Approximate Curvature (KFAC) method [42] can be advantageous when compared to standard stochastic gradient descent-like methods used in the machine learning community such as Adam [35]. KFAC is a method for approximating natural gradient descent efficiently by preconditioning the gradient with an approximate inverse of the Fisher information matrix. Both the overall structure of KFAC and the extra steps required to apply KFAC to an unnormalized wavefunction are described succinctly in [47]. For the convenience of the reader we reproduce here an overview of these topics.

In the exact natural gradient descent method, the gradient of the loss function is multiplied by the inverse of the Fisher information matrix before using the gradient to make a parameter update [1]. This has the effect of taking the path of steepest descent not in Euclidean parameter space, but in the space of probability distributions defined by the model, with distance measured by the KL-divergence [2]. Concretely, updates in natural gradient descent take the form

$$\theta' = \theta - \eta \mathcal{F}^{-1} \nabla_\theta \mathcal{L}(\theta). \tag{1.45}$$

Here $\eta \in \mathbb{R}$ is the learning rate and $\mathcal{F}$ is the Fisher information matrix defined as

$$\mathcal{F}_{ij} = \mathbb{E}_{p(\mathbf{R})}\left[\frac{\partial \log p(\mathbf{R})}{\partial \theta_i}\frac{\partial \log p(\mathbf{R})}{\partial \theta_j}\right]. \tag{1.46}$$

Note that in our case $p(\mathbf{R}) \propto |\Psi_\theta(\mathbf{R})|^2$, though the two are not equal as Ψ is not necessarily normalized. In fact, obtaining the Fisher information matrix requires a slightly different calculation in an unnormalized setting, which is usually referred to as stochastic reconfiguration [5]:

$$\mathcal{F}_{ij} \propto \mathbb{E}_{p(\mathbf{R})}\left[\left(\mathcal{O}_i - \mathbb{E}_{p(\mathbf{R})}\left[\mathcal{O}_i\right]\right)\left(\mathcal{O}_j - \mathbb{E}_{p(\mathbf{R})}\left[\mathcal{O}_j\right]\right)\right], \tag{1.47}$$

where $\mathcal{O}_i = \partial \log |\Psi_\theta(\mathbf{R})| / \partial \theta_i$. The equivalence of this formulation is proved in Appendix C of [47]. In the setting of quantum information geometry, the Fisher information matrix is proportional to, and perhaps more accurately viewed as, the Fubini-Study metric tensor or quantum geometric tensor [56].

Directly inverting the Fisher information matrix is infeasible for large models, as the matrix dimensions scale directly with number of parameters. KFAC solves this problem by making two approximations to the Fisher matrix to allow its efficient inversion. The first is to assume that the Fisher entries for weights in different layers of the network are zero. This assumption reduces the Fisher matrix to a block diagonal form, so that inverting the remaining matrix only requires inverting each block independently. The second is based on the observation that the block corresponding to each layer of the network can be written as the mean-centered covariance of a Kronecker product of two vectors, one consisting of neuron activation values for the inputs to the layer and the other consisting of gradients of the loss with respect to the outputs of the layer. KFAC replaces this with the Kronecker product of the mean-centered covariance of the same vectors. As discussed in [42], this is a significant and theoretically unsupported approximation, but seems to work well in practice, at least in some use cases.

Chapter 2

Neural Network Antisymmetries

In this chapter, we describe the wavefunction parametrizations that we explore. We first briefly describe a collection of conventional VMC ansatzes, from the Slater determinant to the inclusion of Jastrow factors and backflow. We then discuss the FermiNet architecture [47, 53], which consists of a (generalized) neural network backflow that produces permutation equivariant features, followed by a determinant layer to compute the wavefunction amplitude. Finally, we describe the explicitly antisymmetrized layers we explore, which are all composed with the same generalized backflow transformation used in the FermiNet.

Symmetries of interest

It will be convenient for us to first define the notion of group covariance of a function $f : X \to Y$ with respect to a group G. We say f is **G-covariant** with respect to actions of G on X and Y if for all $g \in G$ and $x \in X$, $f(g(x)) = g(f(x))$. For our purposes, $X = (\mathbb{R}^d)^{N_\uparrow + N_\downarrow}$ and $G = S_{N_\uparrow} \times S_{N_\downarrow}$. Recall the notation

$$\begin{aligned} \mathbf{R}^\uparrow &\equiv (\mathbf{r}_1^\uparrow, \ldots, \mathbf{r}_{N_\uparrow}^\uparrow) \equiv (\mathbf{r}_1, \ldots, \mathbf{r}_{N_\uparrow}) \\ \mathbf{R}^\downarrow &\equiv (\mathbf{r}_1^\downarrow, \ldots, \mathbf{r}_{N_\downarrow}^\downarrow) \equiv (\mathbf{r}_{N_\uparrow+1}, \ldots, \mathbf{r}_{N_\uparrow+N_\downarrow}) \end{aligned} \tag{2.1}$$

defined in Section 1.3. The action of G on X is given by the product of the canonical permutation representations of $S_{N_\uparrow}$ and $S_{N_\downarrow}$, defined by the action

$$(\pi \otimes \rho)(\mathbf{R}^\uparrow, \mathbf{R}^\downarrow) = (\pi(\mathbf{R}^\uparrow), \rho(\mathbf{R}^\downarrow)) = (\mathbf{r}_{\pi(1)}^\uparrow, \ldots, \mathbf{r}_{\pi(N_\uparrow)}^\uparrow, \mathbf{r}_{\rho(1)}^\downarrow, \ldots, \mathbf{r}_{\rho(N_\downarrow)}^\downarrow). \tag{2.2}$$

Given this action of G on X, there are three special cases of covariance which we discuss:

(a) $S_{N_\uparrow} \times S_{N_\downarrow}$ **invariant**: $Y = \mathbb{R}$, and the action of G on Y is the trivial one, i.e.

$$(\pi \otimes \rho)(y) = y \tag{2.3}$$

for all $y \in Y$.

(b) $S_{N_\uparrow} \times S_{N_\downarrow}$ **antisymmetric**: $Y = \mathbb{R}$, and the action of G on Y is given by the product of the sign representations of $S_{N_\uparrow}$ and $S_{N_\downarrow}$, i.e.

$$(\pi \otimes \rho)(y) = (-1)^\pi (-1)^\rho y \tag{2.4}$$

for all $y \in Y$.

(c) $S_{N_\uparrow} \times S_{N_\downarrow}$ **equivariant**: $Y = (\mathbb{R}^{d'})^{N_\uparrow + N_\downarrow}$, and the action of G on $\mathbf{y} \in Y$ is the same as that on $\mathbf{R} \in X$:

$$(\pi \otimes \rho)(\mathbf{y}^\uparrow, \mathbf{y}^\downarrow) = (\pi(\mathbf{y}^\uparrow), \rho(\mathbf{y}^\downarrow)) = (\mathbf{y}^\uparrow_{\pi(1)}, \dots, \mathbf{y}^\uparrow_{\pi(N_\uparrow)}, \mathbf{y}^\downarrow_{\rho(1)}, \dots, \mathbf{y}^\downarrow_{\rho(N_\downarrow)}). \tag{2.5}$$

As discussed in Chapter 1, our trial wavefunctions must be $S_{N_\uparrow} \times S_{N_\downarrow}$ antisymmetric. We begin by briefly reviewing the simplest of the antisymmetric ansatzes and how their representation capacity can be modified with invariant and equivariant components.

2.1 The Slater Determinant

Without the antisymmetry requirement for electronic wavefunctions, one might imagine the "simplest" many-body wavefunction ansatz to be a product, known as a **Hartree product**, of single-particle wavefunctions known as **orbitals**:

$$\begin{aligned}\Psi_{\text{Hartree}}(\mathbf{R}) &= \varphi_1(\mathbf{r}_1) \cdots \varphi_N(\mathbf{r}_N) \\ &= \left(\varphi_1^\uparrow(\mathbf{r}_1^\uparrow) \cdots \varphi_{N_\uparrow}^\uparrow(\mathbf{r}_{N_\uparrow}^\uparrow)\right) \left(\varphi_1^\downarrow(\mathbf{r}_1^\downarrow) \cdots \varphi_{N_\downarrow}^\downarrow(\mathbf{r}_{N_\downarrow}^\downarrow)\right).\end{aligned} \tag{2.6}$$

This ansatz is simple in the sense that the electrons are completely *uncorrelated*; the corresponding probability density is a product of N independent densities:

$$p(\mathbf{R}) \propto |\varphi_1(\mathbf{r}_1)|^2 \cdots |\varphi_N(\mathbf{r}_N)|^2 \tag{2.7}$$

However, even if the Hamiltonian includes no electron-electron interaction, the Hartree product cannot describe the electronic wavefunction, as it treats the electrons as distinguishable. Recall that we require our spatial wavefunctions to be $S_{N_\uparrow} \times S_{N_\downarrow}$ antisymmetric, i.e. antisymmetric with respect to the spin-up and spin-down electron positions separately. To introduce the correct symmetry to the Hartree product, let us define the antisymmetrization operator on spin σ as

$$\mathcal{A}_\sigma[f](\mathbf{R}) \equiv \sum_{\pi_\sigma \in S_{N_\sigma}} (-1)^{\pi_\sigma} f(\pi_\sigma(\mathbf{R})), \tag{2.8}$$

where S_{N_σ} is the symmetric group on $\{1, \ldots, N_\sigma\}$. Applying $\mathcal{A}_\uparrow$ and $\mathcal{A}_\downarrow$ to the Hartree product and following the Leibniz formula for the determinant, we arrive at the following ansatz, known as a **Slater determinant**[1]:

$$\begin{aligned}
&\Psi_{\text{Slater}}(\mathbf{R}) = \mathcal{A}_\uparrow \mathcal{A}_\downarrow [\Psi_{\text{Hartree}}](\mathbf{R}) \\
&= \sum_{\pi_\uparrow \in S_{N_\uparrow}} \sum_{\pi_\downarrow \in S_{N_\downarrow}} (-1)^{\pi_\uparrow} (-1)^{\pi_\downarrow} \left(\varphi_1^\uparrow(\mathbf{r}^\uparrow_{\pi_\uparrow(1)}) \cdots \varphi^\uparrow_{N_\uparrow}(\mathbf{r}^\uparrow_{\pi_\uparrow(N_\uparrow)}) \right) \left(\varphi_1^\downarrow(\mathbf{r}^\downarrow_{\pi_\downarrow(1)}) \cdots \varphi^\downarrow_{N_\downarrow}(\mathbf{r}^\downarrow_{\pi_\downarrow(N_\downarrow)}) \right) \\
&= \sum_{\pi_\uparrow \in S_{N_\uparrow}} (-1)^{\pi_\uparrow} \varphi_1^\uparrow(\mathbf{r}^\uparrow_{\pi_\uparrow(1)}) \cdots \varphi^\uparrow_{N_\uparrow}(\mathbf{r}^\uparrow_{\pi_\uparrow(N_\uparrow)}) \sum_{\pi_\downarrow \in S_{N_\downarrow}} (-1)^{\pi_\downarrow} \varphi_1^\downarrow(\mathbf{r}^\downarrow_{\pi_\downarrow(1)}) \cdots \varphi^\downarrow_{N_\downarrow}(\mathbf{r}^\downarrow_{\pi_\downarrow(N_\downarrow)}) \\
&= \det \Phi^\uparrow(\mathbf{R}^\uparrow) \det \Phi^\downarrow(\mathbf{R}^\downarrow),
\end{aligned} \tag{2.9}$$

where we have defined the orbital matrices

$$\Phi^\sigma(\mathbf{R}^\sigma) = \begin{pmatrix} \varphi_1^\sigma(\mathbf{r}_1^\sigma) & \cdots & \varphi_1^\sigma(\mathbf{r}^\sigma_{N_\sigma}) \\ \vdots & \ddots & \vdots \\ \varphi^\sigma_{N_\sigma}(\mathbf{r}_1^\sigma) & \cdots & \varphi^\sigma_{N_\sigma}(\mathbf{r}^\sigma_{N_\sigma}) \end{pmatrix}, \tag{2.10}$$

and the orbitals $\varphi_i^\sigma(\mathbf{r})$ are optimized to obtain the lowest variational energy. The optimization of this ansatz is the core of the **Hartree-Fock** (HF) method [59]. Typically, the orbitals are parametrized by linear combinations of fixed basis functions which are usually chosen specific to the component atoms (referred to as **atomic orbitals**). The Hartree-Fock problem can be solved efficiently for a wide range of systems of interest using matrix diagonalization methods[2], and the solution is sometimes known as the **mean-field** solution, as each orbital only accounts for interaction

[1]The word determinant is used in the singular sense, as the full spatial-spin wavefunction can be described by a single determinant of single-particle orbitals.

[2]Plugging the Slater determinant into a mean-field non-interacting Hamiltonian which approximates the many-body Hamiltonian gives a generalized eigenvalue problem involving the so-called **Fock matrix** (which depends on the orbitals) and an orbital-orbital overlap matrix. The solutions to this problem give another set of orbitals, and this procedure is iterated until the change in expected energy between ansatzes falls below a desired tolerance. The result is a set of orbitals which are consistent with the Fock matrix they generate. Thus this algorithm is also called the **self-consistent field** iteration.

with other electrons in an average way. Due to the product structure between the spin-up and spin-down terms, the only explicit electron correlation included in a single Slater determinant is the antisymmetry between electrons of like-spin. This type of interaction, present even in this simple ansatz, is known as the **exchange interaction**.

Because the Slater determinant is only able to account for the electron-electron Coulomb interaction in a mean-field way, the difference in energy between the Hartree-Fock energy and the true ground state energy is known as the **correlation energy**[3].

To improve the representation power of the trial wavefunction, sums of multiple determinants may be used, of the form

$$\Psi(\mathbf{R}) = \sum_{k=1}^{K} c_k \det \Phi^{k\uparrow}(\mathbf{R}^{\uparrow}) \det \Phi^{k\downarrow}(\mathbf{R}^{\downarrow}). \tag{2.11}$$

However, for larger systems which exhibit non-trivial electron correlation, it becomes rapidly impractical to use the many determinants required to reach a sufficiently accurate energy. This consideration has led to considerable interest in designing modifications to the Slater determinant-based wavefunction, such as Jastrow factors or backflow, which we describe below.

Jastrow factors

A standard way to incorporate additional electronic correlation into the wavefunction ansatz is to include a multiplicative factor of the form $\exp(J)$, where J is known as the **Jastrow** factor [22]. In order to preserve the antisymmetry within like-spin positions, the function J must be invariant with respect to the action of $S_{N_\uparrow} \times S_{N_\downarrow}$. One choice of Jastrow form [16] which explicitly handles electron-nuclei, electron-

[3]A number of different Hartree-Fock variants are used in practice: when $N_\uparrow = N_\downarrow = N/2$ (a **closed-shell** molecule) and the orbitals are restricted so that $\varphi_i^\uparrow \equiv \varphi_i^\downarrow$ for $1 \leq i \leq N/2$, the method is known as the **restricted Hartree-Fock** (RHF) method. When $N_\uparrow \neq N_\downarrow$ (an **open-shell** molecule) but the first $\min(N_\uparrow, N_\downarrow)$ orbitals are restricted to match across the spins, the method is known as the **restricted open-shell Hartree-Fock** (ROHF) method. Finally, when the orbitals are not restricted across the spins, the method is known as the **unrestricted Hartree-Fock** (UHF) method.

electron, and electron-electron-nuclei terms is given by

$$\begin{aligned}J(\mathbf{R};\mathbf{R}^{(a)}) = &\sum_{\alpha\in\{\uparrow,\downarrow\}}\sum_{i=1}^{N_\alpha}\sum_{I=1}^{M}\chi_I\left(\left|\mathbf{r}_i^\alpha-\mathbf{R}_I^{(a)}\right|\right)+\sum_{\alpha\in\{\uparrow,\downarrow\}}\sum_{\beta\in\{\uparrow,\downarrow\}}\sum_{i=1}^{N_\alpha}\sum_{j=1}^{N_\beta}u^{\alpha\beta}\left(\left|\mathbf{r}_i^\alpha-\mathbf{r}_j^\beta\right|\right)\\ &+\sum_{\alpha\in\{\uparrow,\downarrow\}}\sum_{\beta\in\{\uparrow,\downarrow\}}\sum_{i=1}^{N_\alpha}\sum_{j=1}^{N_\beta}\sum_{I}f_I^{\alpha\beta}\left(\left|\mathbf{r}_i^\alpha-\mathbf{r}_j^\beta\right|,\left|\mathbf{r}_i^\alpha-\mathbf{R}_I^{(a)}\right|,\left|\mathbf{r}_j^\beta-\mathbf{R}_I^{(a)}\right|\right),\end{aligned} \tag{2.12}$$

where the functions $\{\chi_I\}$, $\{u^{\alpha\beta}\}$, and $\{f_I^{\alpha\beta}\}$ are optimized along with the Slater orbitals. The resulting wavefunction has the form

$$\Psi_{\text{Slater-Jastrow}}(\mathbf{R}) = D(\mathbf{R})\exp(J(\mathbf{R})), \tag{2.13}$$

where $D(\mathbf{R})$ is the sum of one or more Slater determinant terms.

Backflow

Adding a multiplicative Jastrow factor cannot improve the nodal surface of the trial wavefunction. A different way of incorporating electronic correlation which does have the ability to modify the wavefunction nodes is known as **backflow**, which replaces the coordinates $\mathbf{R}$ in the Slater determinant with a set of transformed coordinates $\mathbf{Y}(\mathbf{R})$, which generalizes the original proposal of "backflow" by Feynman and Cohen [18]. In order to preserve the $S_{N_\uparrow}\times S_{N_\downarrow}$ antisymmetry, the transformation $\mathbf{Y}$ must be $S_{N_\uparrow}\times S_{N_\downarrow}$ equivariant. One possible form of the backflow transformation [40] is given by

$$\mathbf{y}_i(\mathbf{R}) = \mathbf{r}_i + \xi_i(\mathbf{R}), \tag{2.14}$$

where ξ_i can include electron-nuclei, electron-electron, and electron-electron-nuclei terms:

$$\begin{aligned}\xi_i(\mathbf{R}) = &\sum_{j\neq i}\eta\left(|\mathbf{r}_{ij}|\right)\mathbf{r}_{ij}+\sum_{I}\mu\left(|\mathbf{r}_{iI}|\right)\mathbf{r}_{iI}\\ &+\sum_{j\neq i}\sum_{I}\Phi\left(|\mathbf{r}_{ij}|,|\mathbf{r}_{iI}|,|\mathbf{r}_{jI}|\right)\mathbf{r}_{ij}+\Theta\left(|\mathbf{r}_{ij}|,|\mathbf{r}_{iI}|,|\mathbf{r}_{jI}|\right)\mathbf{r}_{iI},\end{aligned} \tag{2.15}$$

where we use the notation $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$ and $\mathbf{r}_{iI} = \mathbf{r}_i - \mathbf{R}_I^{(a)}$. Together with the Jastrow factor J, the resulting wavefunction has the form

$$\Psi_{\text{Slater-Jastrow-backflow}}(\mathbf{R}) = D(\mathbf{Y}(\mathbf{R}))\exp(J(\mathbf{R})), \tag{2.16}$$

where $D(\mathbf{R})$ is the sum of one or more Slater determinant terms.

2.2 FermiNet: The Generalized Slater Determinant

The overall structure of the FermiNet may be described as a composition of a general $S_{N_\uparrow} \times S_{N_\downarrow}$ equivariant feature map $\mathbf{Y}(\mathbf{R}) \equiv (\mathbf{Y}^\uparrow(\mathbf{R}), \mathbf{Y}^\downarrow(\mathbf{R}))$, with an antisymmetric layer constructed as a sum of products of determinants of orbital matrices $\Phi^{k\sigma}$:

$$\Psi(\mathbf{R}) = \sum_{k=1}^{K} \det \Phi^{k\uparrow}(\mathbf{Y}^\uparrow(\mathbf{R})) \det \Phi^{k\downarrow}(\mathbf{Y}^\downarrow(\mathbf{R})). \tag{2.17}$$

We describe the equivariant feature map and the orbital-determinant layer separately below.

Permutation equivariant features in FermiNet

The FermiNet backflow $\mathbf{Y}$ is a map from particle positions $\mathbf{R} \in \mathbb{R}^{N\times d}$ to generalized coordinates $\mathbf{Y} \in \mathbb{R}^{N\times d'}$. While $d = 3$ is the dimension of the physical space, d' is the number of features and can be chosen arbitrarily, which generalizes the form of the backflow transformation in Eq. (2.14).

Recall that the FermiNet map $\mathbf{Y}$ must be $S_{N_\uparrow} \times S_{N_\downarrow}$ equivariant. To ensure that $\mathbf{Y}$ is equivariant with respect to elements of $S_{N_\uparrow} \times S_{N_\downarrow}$ while incorporating information about the two-electron distances, the basic layer involves two streams: a "one-electron stream" that starts with the electron positions $\mathbf{R} = (\mathbf{r}_1, \dots \mathbf{r}_N)$, and a "two-electron stream" that starts with the electron-electron displacements $\mathbf{r}_{ij} \equiv \mathbf{r}_i - \mathbf{r}_j$. The streams are averaged over the electrons, concatenated onto the one-electron stream, and a dense layer followed by a nonlinear activation function such as tanh is applied. Residual connections [25] are also used between layers of the same shape for both streams.

More precisely, if $\mathbf{h}_i^{l\alpha}$ and $\mathbf{h}_{ij}^{l\alpha\beta}$ are the outputs of the one- and two- electron streams at layer l with spins $\alpha, \beta \in \{\uparrow, \downarrow\}$, then the concatenated vector for index i and spin α is

$$\mathbf{f}_i^{l\alpha} = \left(\mathbf{h}_i^{l\alpha}, \frac{1}{N_\uparrow} \sum_{j=1}^{N_\uparrow} \mathbf{h}_j^{l\uparrow}, \frac{1}{N_\downarrow} \sum_{j=1}^{N_\downarrow} \mathbf{h}_j^{l\downarrow}, \frac{1}{N_\uparrow} \sum_{j=1}^{N_\uparrow} \mathbf{h}_{ij}^{l\alpha\uparrow}, \frac{1}{N_\downarrow} \sum_{j=1}^{N_\downarrow} \mathbf{h}_{ij}^{l\alpha\downarrow} \right) \tag{2.18}$$

and the output of layer $l + 1$ is given by the two streams

$$\begin{aligned} \mathbf{h}_i^{(l+1)\alpha} &= \tanh\left(V^l \mathbf{f}_i^{l\alpha} + \mathbf{b}^l\right) + \mathbf{h}_i^{l\alpha}, \\ \mathbf{h}_{ij}^{(l+1)\alpha\beta} &= \tanh\left(W^l \mathbf{h}_{ij}^{l\alpha\beta} + \mathbf{b}^l\right) + \mathbf{h}_{ij}^{l\alpha\beta}. \end{aligned} \tag{2.19}$$

As an initial pre-processing step, the electron positions $\mathbf{R}$ are converted to "atomic coordinates" as

$$\mathbf{h}_i^{0\alpha} = \left(\mathbf{r}_i^\alpha - \mathbf{R}_1^{(a)}, \left|\mathbf{r}_i^\alpha - \mathbf{R}_1^{(a)}\right|, \ldots, \mathbf{r}_i^\alpha - \mathbf{R}_M^{(a)}, \left|\mathbf{r}_i^\alpha - \mathbf{R}_M^{(a)}\right|\right), \tag{2.20}$$

$$\mathbf{h}_{ij}^{0\alpha\beta} = \left(\mathbf{r}_i^\alpha - \mathbf{r}_j^\beta, \left|\mathbf{r}_i^\alpha - \mathbf{r}_j^\beta\right|\right), \tag{2.21}$$

which are invariant with respect to a simultaneous translation of the entire system. The explicit dependence on the absolute values enables the network to efficiently represent the derivative discontinuity due to the electron-nuclei cusp and electron-electron cusp, respectively [31, 47].

Each map $(\mathbf{h}^{l\uparrow}, \mathbf{h}^{l\downarrow}) \mapsto (\mathbf{h}^{(l+1)\uparrow}, \mathbf{h}^{(l+1)\downarrow})$ is $S_{N_\uparrow} \times S_{N_\downarrow}$ equivariant due to the averaging procedure [67] in Eq. (2.18), and the map $\mathbf{R} \mapsto \mathbf{h}^0$ is parallel in the particle index i and thus equivariant as well. Therefore the map $\mathbf{R} \mapsto \mathbf{Y} \equiv (\mathbf{Y}^\uparrow, \mathbf{Y}^\downarrow) \equiv (\mathbf{h}^{L\uparrow}, \mathbf{h}^{L\downarrow})$, where L is the total number of one-electron layers, is also $S_{N_\uparrow} \times S_{N_\downarrow}$ equivariant.

Note that both $\mathbf{Y}^\uparrow$ and $\mathbf{Y}^\downarrow$ depend on the positions of all electrons. The index α in the notation $\mathbf{Y}^\alpha$ does not denote a dependence of $\mathbf{Y}$ only on the corresponding α-spin inputs $\mathbf{R}^\alpha$, but rather denotes the symmetry constraint of $\mathbf{Y}^\alpha$ as equivariant with respect to the action of S_{N_α} on $\mathbf{R}^\alpha$ and invariant with respect to the action of S_{N_β} on $\mathbf{R}^\beta$. Hence in this paper, we shall refer to the index $\{\uparrow, \downarrow\}$ in the expression $(\mathbf{Y}^\uparrow, \mathbf{Y}^\downarrow)$ as the **pseudospin** index. It has been shown constructively, though without explicit error bounds, that a simplified version of this construction [29] can approximate all and only the equivariant continuous functions.

Antisymmetric layer in FermiNet

Once the equivariant feature maps $\mathbf{Y} = (\mathbf{Y}^\uparrow, \mathbf{Y}^\downarrow)$ are generated, they are then used to generate K pairs of orbital matrices $(\Phi^{k\uparrow}(\mathbf{Y}^\uparrow), \Phi^{k\downarrow}(\mathbf{Y}^\downarrow))$. These orbital matrices are constructed using a set of per-pseudospin single particle orbitals $\varphi_i^{k\sigma}$ $(\sigma \in \{\uparrow, \downarrow\})$, which are defined by applying a simple dense layer to the equivariant features and multiplying by an exponential envelope function:

$$\varphi_i^{k\sigma}(\mathbf{y}_j^\sigma) = (\mathbf{w}_i^{k\sigma} \cdot \mathbf{y}_j^\sigma + \mathbf{b}_i^{k\sigma}) \sum_I d_{i,I}^{k\sigma} \exp\left(-\left|\mathbf{\Sigma}_{i,I}^{k\sigma}(\mathbf{r}_j - \mathbf{R}_I^{(a)})\right|\right). \tag{2.22}$$

Here $\mathbf{w}_i^{k\sigma} \in \mathbb{R}^m$ and $b_i^{k\sigma} \in \mathbb{R}$ are the weights and biases of the dense layer, where m is the number of dimensions in the equivariant features $\mathbf{y}_i^\sigma$, which is chosen to be constant for all i, σ. The exponential envelopes are parameterized by $d_{i,I}^{k\sigma} \in \mathbb{R}$ and a matrix $\mathbf{\Sigma}_{i,I}^{k\sigma} \in \mathbb{R}^{d\times d}$, which can also be set to a scaled identity matrix to simplify the ansatz. The parameters $\mathbf{w}, \mathbf{b}, \mathbf{\Sigma}, \mathbf{d}$ are trainable and dependent on the pseudospin index σ. The exponential term ensures that the wavefunction is normalizable and that the support of each orbital does not extend too far away into the vacuum. While the orbital functions $\varphi_i^{k\sigma}$ are applied only to single feature vectors $\mathbf{y}_j$, the ansatz can express complex correlations because the feature vectors themselves depend on all of the particles in a complex way.

The orbital matrices follow naturally from these single particle orbitals as

$$\Phi^{k\sigma}(\mathbf{Y}^\sigma) = \begin{pmatrix} \varphi_1^{k\sigma}(\mathbf{y}_1^\sigma) & \cdots & \varphi_1^{k\sigma}(\mathbf{y}_{N_\sigma}^\sigma) \\ \vdots & \ddots & \vdots \\ \varphi_{N_\sigma}^{k\sigma}(\mathbf{y}_1^\sigma) & \cdots & \varphi_{N_\sigma}^{k\sigma}(\mathbf{y}_{N_\sigma}^\sigma) \end{pmatrix}. \tag{2.23}$$

Once these orbital matrices are constructed, FermiNet creates an antisymmetric wavefunction using a sum of products of determinants:

$$\Psi(\mathbf{R}) = \sum_k \det \Phi^{k\uparrow} \det \Phi^{k\downarrow} \tag{2.24}$$

These determinants are similar to Slater determinants except that their orbitals are general equivariant functions of all of the input particles rather than simple single particle orbitals. Such determinants are thus referred to as generalized Slater determinants. In fact, the original FermiNet architecture [47] used a weighted sum of products of generalized Slater determinants by adding in a set of trainable parameters w_k and letting

$$\Psi(\mathbf{R}) = \sum_k w_k \det \Phi^{k\uparrow} \det \Phi^{k\downarrow}. \tag{2.25}$$

However, these trainable weights were removed by the original authors in their follow-up work [53], as they do not add extra expressiveness on top of the ability to tune the scale of the orbital matrices themselves. We follow the simplified construction.

Full determinant FermiNet

We also explore a variant of the FermiNet called full determinant mode which, like the GA layer, does not assume a factorized form over the two pseudospins. In the full determinant mode, the single particle orbitals take exactly the same form as those used in the regular FermiNet architecture. The difference is that instead of using N_σ orbitals for each spin σ, we use N orbitals for both spins, where $N = N_\uparrow + N_\downarrow$. The up- and down-pseudospin orbital matrices are then concatenated into a square $N \times N$ matrix before taking the determinant. The new formula for the orbital matrices is

$$\Phi^{k\sigma}_{\text{full}}(\mathbf{Y}^\sigma) = \begin{pmatrix} \varphi^{k\sigma}_1(\mathbf{y}^\sigma_1) & \cdots & \varphi^{k\sigma}_1(\mathbf{y}^\sigma_{N_\sigma}) \\ \vdots & \ddots & \vdots \\ \varphi^{k\sigma}_N(\mathbf{y}^\sigma_1) & \cdots & \varphi^{k\sigma}_N(\mathbf{y}^\sigma_{N_\sigma}) \end{pmatrix}, \tag{2.26}$$

where the only difference from Eq. (2.23) is the change of the maximum orbital index from the pseudospin-specific N_σ to the total particle count N. Therefore $\Phi^{k\sigma}_{\text{full}}(\mathbf{Y}^\sigma)$ is a matrix of size $N \times N_\sigma$. The final ansatz is then generated as the sum of the determinants of the concatenated orbital matrices:

$$\Psi_{\text{FermiNet,full}}(\mathbf{R}) = \sum_{k=1}^{K} \det\left[\Phi^{k\uparrow}_{\text{full}}, \Phi^{k\downarrow}_{\text{full}}\right]. \tag{2.27}$$

The idea behind this construction is to provide a more flexible way to treat the interactions between the two pseudospin components. Importantly, because $\mathbf{Y}$ is only equivariant with respect to permutations which exchange particles of the same spin, the concatenated determinant does not enforce an antisymmetry constraint between particles of opposite spins. We also note that it is possible to reconstruct the original FermiNet ansatz as a special case of the full determinant ansatz, by setting $\varphi^{k\sigma}_i(\mathbf{y}^\sigma_j) = 0$ whenever $\sigma = \uparrow$ and $i > N_\uparrow$ or $\sigma = \downarrow$ and $i \leq N_\uparrow$. In that case the full matrix becomes block-diagonal and the determinant factors into a simple product of pseudospin determinants [47]. We are particularly interested in the evaluating the performance of this full determinant mode when $K = 1$, which we refer to as the full single-determinant FermiNet.

2.3 Explicit Antisymmetrization

Generic antisymmetric neural network layer

In order to assess the effectiveness of the antisymmetric layer of the FermiNet, we can replace the product of pseudospin determinant terms with a truly universal antisymmetric neural network layer. Following the Leibniz formula for the determinant, the antisymmetric structure for the standard single-determinant FermiNet can be

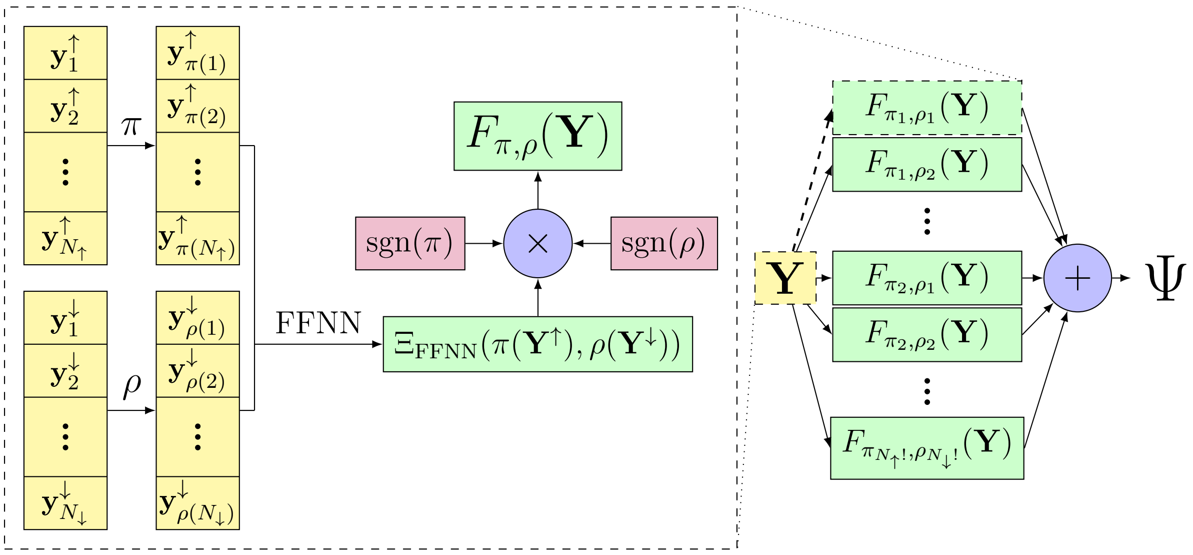

$\mathbf{y}_1^\uparrow$
$\mathbf{y}_2^\uparrow$
$\mathbf{y}_{N_\uparrow}^\uparrow$
π
$\mathbf{y}_{\pi(1)}^\uparrow$
$\mathbf{y}_{\pi(2)}^\uparrow$
$\mathbf{y}_{\pi(N_\uparrow)}^\uparrow$
$\mathbf{y}_1^\downarrow$
$\mathbf{y}_2^\downarrow$
$\mathbf{y}_{N_\downarrow}^\downarrow$
ρ
$\mathbf{y}_{\rho(1)}^\downarrow$
$\mathbf{y}_{\rho(2)}^\downarrow$
$\mathbf{y}_{\rho(N_\downarrow)}^\downarrow$
FFNN
$\Xi_{\text{FFNN}}(\pi(\mathbf{Y}^\uparrow), \rho(\mathbf{Y}^\downarrow))$
$\text{sgn}(\pi)$
$\times$
$\text{sgn}(\rho)$
$F_{\pi,\rho}(\mathbf{Y})$
$\mathbf{Y}$
$F_{\pi_1,\rho_1}(\mathbf{Y})$
$F_{\pi_1,\rho_2}(\mathbf{Y})$
$F_{\pi_2,\rho_1}(\mathbf{Y})$
$F_{\pi_2,\rho_2}(\mathbf{Y})$
$F_{\pi_{N_\uparrow!},\rho_{N_\downarrow!}}(\mathbf{Y})$
$+$
Ψ

Figure 2.1: The architecture for the generic antisymmetric layer. Left: calculation of the wavefunction contribution from a single pair of permutations π and ρ, denoted by $F_{\pi,\rho}(\mathbf{Y})$. Right: combination of contributions for all permutations π and ρ. This is implemented as a batch calculation over all pairs of permutations, but we show separate arrows for each pair of permutations to emphasize the factorial complexity of the operation.

written as

$$\begin{aligned}\Psi(\mathbf{R}) &= \det \Phi^\uparrow(\mathbf{Y}^\uparrow(\mathbf{R})) \cdot \det \Phi^\downarrow(\mathbf{Y}^\downarrow(\mathbf{R})) \\ &= \prod_{\sigma\in\{\uparrow,\downarrow\}} \left(\sum_{\pi_\sigma \in S_{N_\sigma}} (-1)^{\pi_\sigma} \varphi_1^\sigma(\mathbf{y}^\sigma_{\pi_\sigma(1)}) \cdots \varphi^\sigma_{N_\sigma}(\mathbf{y}^\sigma_{\pi_\sigma(N_\sigma)}) \right), \\ &\equiv \prod_{\sigma\in\{\uparrow,\downarrow\}} \mathcal{A}_\sigma[\Xi^\sigma_{\text{FermiNet}}](\mathbf{Y}^\sigma),\end{aligned} \tag{2.28}$$

where we have defined the orbital product function $\Xi^\sigma_{\text{FermiNet}}(\mathbf{Y}^\sigma) = \prod_{i=1}^{N_\sigma} \varphi_i^\sigma(\mathbf{y}_i^\sigma, \theta_i^\sigma)$.

However, we can treat the interaction between pseudospins in a more universal way by considering just a single function $f(\mathbf{R}^\uparrow, \mathbf{R}^\downarrow)$ of all electron positions and antisymmetrizing f with respect to only the subsets of the input indices that correspond to the two spins:

$$\mathcal{A}_\uparrow\mathcal{A}_\downarrow[f](\mathbf{R}^\uparrow, \mathbf{R}^\downarrow) = \sum_{\pi_\uparrow \in S_{N_\uparrow}} \sum_{\pi_\downarrow \in S_{N_\downarrow}} (-1)^{\pi_\uparrow}(-1)^{\pi_\downarrow} f(\pi_\uparrow(\mathbf{R}^\uparrow), \pi_\downarrow(\mathbf{R}^\downarrow)). \tag{2.29}$$

The standard FermiNet can be written as the doubly antisymmetrized product of orbitals over all particles

$$\Psi_{\text{FermiNet}}(\mathbf{R}) = \mathcal{A}_{\uparrow}\mathcal{A}_{\downarrow}\left[\Xi_{\text{FermiNet}}\right](\mathbf{Y}^{\uparrow}, \mathbf{Y}^{\downarrow}), \tag{2.30}$$

where we define the product over all orbitals

$$\Xi_{\text{FermiNet}}(\mathbf{Y}^{\uparrow}, \mathbf{Y}^{\downarrow}) = \varphi_1^{\uparrow}(\mathbf{y}_1^{\uparrow}) \cdots \varphi_{N_\uparrow}^{\uparrow}(\mathbf{y}_{N_\uparrow}^{\uparrow})\varphi_1^{\downarrow}(\mathbf{y}_2^{\downarrow}) \cdots \varphi_{N_\downarrow}^{\downarrow}(\mathbf{y}_{N_\downarrow}^{\downarrow}). \tag{2.31}$$

If we replace this all-particle product with a single feed-forward neural network (FFNN) denoted by $\Xi_{\text{FFNN}}(\mathbf{Y}^{\uparrow}, \mathbf{Y}^{\downarrow})$, we arrive at an architecture that we refer to as the generic antisymmetric layer (GA):

$$\Psi_{\text{GA}}(\mathbf{R}) = \mathcal{A}_{\uparrow}\mathcal{A}_{\downarrow}[\Xi_{\text{FFNN}}](\mathbf{Y}^{\uparrow}, \mathbf{Y}^{\downarrow}). \tag{2.32}$$

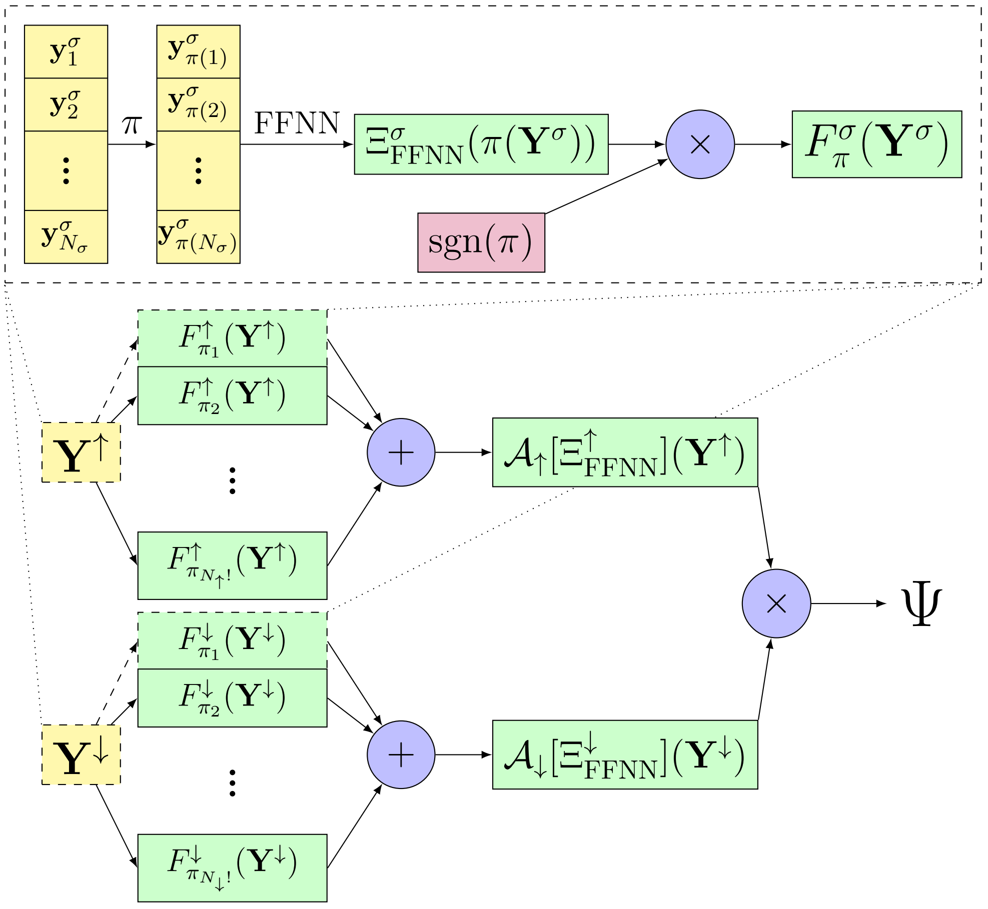
This ansatz can be evaluated explicitly with $N_\uparrow!N_\downarrow!$ evaluations of Ξ_{FFNN}. Importantly, due to the equivariance property of $\mathbf{Y}$, we do not need to antisymmetrize the composed function $\Xi_{\text{FFNN}} \circ \mathbf{Y}$, but only the comparatively small Ξ_{FFNN}. We have found that we can achieve sufficient expressiveness with a very simple choice of Ξ_{FFNN}, further reducing the computational cost. In our experiments, we use a single hidden layer with $N_{\text{FFNN}} = 64$ nodes and a single application of the tanh activation function, followed by a linear combination operation. The cost of a single evaluation of Ξ_{FFNN} with a single hidden layer with N_{FFNN} nodes is $O(d(N_\uparrow + N_\downarrow)N_{\text{FFNN}})$, resulting in an overall cost of $O(d(N_\uparrow + N_\downarrow)N_\uparrow!N_\downarrow!N_{\text{FFNN}})$. We depict the construction of the GA layer in Fig. 2.1.

This construction is a universal replacement for the antisymmetry layer in the original FermiNet, due to the universality of neural networks to approximate functions of various desired smoothness classes with an appropriate choice of activation function [4, 48, 50, 17]. In fact, the multiplication of two numbers can be approximated to arbitrary accuracy with just four neurons [37], and following equations (2.22) and (2.28), the FermiNet determinant is simply an antisymmetrized product of simple linear and exponential terms. In this work we choose Ξ_{FFNN} to be a one-hidden-layer feed-forward neural network for simplicity and efficiency, but one can choose any universal function class, e.g. residual neural networks.

Factorized antisymmetric neural network layer

Instead of replacing the entire antisymmetric layer of the FermiNet, we can consider replacing each product function $\Xi^{\sigma}_{\text{FermiNet}}$ in Eq. (2.28) with a feed-forward neural network, arriving at the ansatz

$$\Psi(\mathbf{R}) = \prod_{\sigma\in\{\uparrow,\downarrow\}} \mathcal{A}_{\sigma}[\Xi^{\sigma}_{\text{FFNN}}](\mathbf{Y}^{\sigma}). \tag{2.33}$$



$\mathbf{y}_1^\sigma$
$\mathbf{y}_2^\sigma$
$\mathbf{y}_{N_\sigma}^\sigma$
π
$\mathbf{y}_{\pi(1)}^\sigma$
$\mathbf{y}_{\pi(2)}^\sigma$
$\mathbf{y}_{\pi(N_\sigma)}^\sigma$
FFNN
$\Xi_{\mathrm{FFNN}}^\sigma(\pi(\mathbf{Y}^\sigma))$
$\mathrm{sgn}(\pi)$
$\times$
$F_\pi^\sigma(\mathbf{Y}^\sigma)$
$\mathbf{Y}^\uparrow$
$F_{\pi_1}^\uparrow(\mathbf{Y}^\uparrow)$
$F_{\pi_2}^\uparrow(\mathbf{Y}^\uparrow)$
$F_{\pi_{N_\uparrow!}}^\uparrow(\mathbf{Y}^\uparrow)$
$+$
$\mathcal{A}_\uparrow[\Xi_{\mathrm{FFNN}}^\uparrow](\mathbf{Y}^\uparrow)$
$\times$
Ψ
$\mathbf{Y}^\downarrow$
$F_{\pi_1}^\downarrow(\mathbf{Y}^\downarrow)$
$F_{\pi_2}^\downarrow(\mathbf{Y}^\downarrow)$
$F_{\pi_{N_\downarrow!}}^\downarrow(\mathbf{Y}^\downarrow)$
$+$
$\mathcal{A}_\downarrow[\Xi_{\mathrm{FFNN}}^\downarrow](\mathbf{Y}^\downarrow)$

Figure 2.2: The architecture for the factorized antisymmetric of rank 1 (FA-1). Top: calculation of the wavefunction contribution from a single pseudospin σ and permutation π, denoted by $F_\pi^\sigma(\mathbf{Y}^\sigma)$. Bottom: combination of contributions for all permutations π for both up and down pseudospin components. This is implemented as a batch calculation for each pseudospin, but we show separate arrows for each permutation to emphasize the factorial complexity of the operation. For FA-K this entire computation would be copied K times, with a different feed-forward-neural network in each copy, and the results would be added together.

We can relate this ansatz to the generic antisymmetric layer by assuming that Ξ_{FFNN} admits a functional low rank decomposition

$$\Xi_{\text{FFNN}}(\mathbf{Y}^{\uparrow}(\mathbf{R}), \mathbf{Y}^{\downarrow}(\mathbf{R})) \approx \sum_{k=1}^{K} \Xi_{\text{FFNN}}^{k\uparrow}(\mathbf{Y}^{\uparrow}(\mathbf{R}))\Xi_{\text{FFNN}}^{k\downarrow}(\mathbf{Y}^{\downarrow}(\mathbf{R})). \tag{2.34}$$

This is called the factorized antisymmetric layer of rank-K (FA-K). When $K = 1$, we recover the ansatz in Eq. (2.33). We treat this FA-1 layer, also depicted in Fig. 2.2, as an especially interesting case given the conjecture of [29] that a single generalized Slater determinant may be universal. We note that much like in the FermiNet pseudospin determinant terms, the electron positions of different spins do interact with each other in each FA pseudospin term due to the construction of the backflow (2.18).

We can evaluate the FA-1 layer using $N_\sigma!$ evaluations of $\Xi_{\text{FFNN}}^{\sigma}$ for each pseudospin. The cost of a single evaluation of $\Xi_{\text{FFNN}}^{\sigma}$ is $O(dN_\sigma N_{\text{FFNN}})$ operations for the matrix-vector multiplication, and the total cost of the explicit antisymmetrization for FA-1 is $O(d(N_\uparrow N_\uparrow! + N_\downarrow N_\downarrow!)N_{\text{FFNN}})$.

It is worth noting that both the factorized and the generic antisymmetric ansatzes have addtional drawbacks beyond the obviously prohibitive factorial scaling. In our experiments with these ansatzes, we observed a great deal of numerical instability due to the massive numerical sign cancellation of the generally non-zero terms in the summations over the symmetric groups. We partially ameliorated this issue by performing the wavefunction evaluation in double precision instead of the more standard single precision (or even half precision) for modern deep learning, but even with this adjustment the numerical stability properties are far too unfavorable to scale these ansatzes as $N_\uparrow$ and $N_\downarrow$ grow large. Thus these ansatzes are certainly not intended to be used to directly approximate the ground state wavefunction of heavy atoms or large molecules, but are instead used in this paper as a diagnostic tool for better understanding the empirical performance of the FermiNet backflow and antisymmetry layers. More details about the practical effects of the numerical instabilities on our experiments are available in Section 3.1.

Jastrow factors

Although the GA architecture can represent general antisymmetric functions on compact domains, we have found that without some mechanism of confining the support size of the wavefunction, the Monte Carlo sampling procedure often becomes unstable. In the standard FermiNet orbitals, this decay is handled by the simple exponential envelope terms in Eq. (2.22). In our generic antisymmetric layer, however, we have not directly included exponential decay terms in the antisymmetric part, so we require the presence of an additional decay term in the form of a Jastrow factor. We found that including an expressive Jastrow factor greatly increased the stability and accuracy of the ansatz, which suggests that the size of the wavefunction support and the behavior of the tails are of practical importance to the quality of the approximation.

Recall that we may implement a Jastrow factor by multiplying an antisymmetric wavefunction ansatz by $\exp\left(J\left(\mathbf{R}^{\uparrow}, \mathbf{R}^{\downarrow}; \mathbf{R}^{(a)}\right)\right)$, where the Jastrow factor J is a function of the electron positions $\mathbf{R}$ and the nuclei locations $\mathbf{R}^{(a)}$. In order for $J(\mathbf{R})$ to capture the decay of the wavefunction, it will need to satisfy $J \to -\infty$ as $|\mathbf{R}| \to \infty$. One possibility is to use a simple one-body Jastrow from the first term in Eq. (2.12) and let χ_I represent multiplication by a fixed constant for each nucleus, so that $\chi_I(r) = -a_I r$, with $a_I > 0$. Then we have

$$J(\mathbf{R}; \mathbf{R}^{(a)}) = \sum_{\alpha \in \{\uparrow,\downarrow\}} \sum_{i=1}^{N_\alpha} \sum_{I=1}^{M} \chi_I\left(\left|\mathbf{R}_i^\alpha - \mathbf{R}_I^{(a)}\right|\right) = \sum_{\alpha \in \{\uparrow,\downarrow\}} \sum_{i=1}^{N_\alpha} \sum_{I=1}^{M} -a_I \left|\mathbf{R}_i^\alpha - \mathbf{R}_I^{(a)}\right|, \tag{2.35}$$

Similarly, one could use the first two terms in Eq. (2.12) to form a simple two-body Jastrow, with a similar choice for the electron-electron interaction which is identical for the two spin species, i.e. $u^{\alpha\beta}(r) = -\gamma r$, with $\gamma > 0$. These approaches control the support size of the wavefunction, but do not allow much flexibility in the shape of the wavefunction tails outside of the asymptotic regime, where the wavefunction decay is known to be a simple isotropic exponential decay.

To build a more general Jastrow factor, we may leverage the generality of the FermiNet backflow construction to form the backflow-based Jastrow

$$J(\mathbf{R}; \mathbf{R}^{(a)}) = -\frac{1}{N} \sum_{\alpha \in \{\uparrow,\downarrow\}} \sum_{i=1}^{N_\alpha} \left|\mathbf{Y}^{\alpha}_{\text{Jastrow},i}(\mathbf{R})\right|. \tag{2.36}$$

For our numerical results involving the GA and FA-K architectures, we use this general Jastrow expression. The Jastrow factor needs to be able to grow small as the electron positions move far from the nuclei, which suggests the use of an unbounded activation function. To achieve this in practice, we simply swap out the tanh activation in Eq. (2.19) for an approximate GeLU activation [27],

$$\text{GeLU}(x) = 0.5x\left(1 + \tanh\left(\sqrt{\frac{2}{\pi}}\left(x + 0.044715x^3\right)\right)\right) \tag{2.37}$$

which, like the hyperbolic tangent function, has the desirable property of being smooth everywhere.

Chapter 3

Numerical experiments

In this chapter, we compare the previously described architectures on small atomic and molecular systems. We used A100 GPUs on the Google Cloud Platform (GCP) for any calculations that required double precision, and GTX 2080TI GPUs with the Berkeley Research Computing (BRC) program for all other calculations. In our experiments, we rely on the JAX implementation of KFAC provided by the work of [47]. In using this implementation, we register all dense layers in our networks with KFAC, including those within the feed-forward neural networks of our generic antisymmetric and factorized antisymmetric layers. This ensures that we use the Kronecker product approximation of the Fisher matrix for all layers in the network, rather than defaulting to a simpler diagonal approximation.

To estimate energy values accurately after training, we ran pure MCMC for a large number of iterations without performing parameter updates, collecting samples every 10 iterations. We also estimated the integrated autocorrelation of the local energy during these evaluation runs in order to get a robust estimate of the standard error of our energy estimates as described in Section 1.3.

Throughout this chapter, we use the following conventions to present our numerical results: all units are atomic units (a.u.) unless otherwise specified. The estimator of the energy used is the sample mean followed by, in parentheses, the standard error in the last digit(s) of the estimate. For example, -54.58868(4) means a sample mean of -54.58868 a.u. with a standard error of approximately 4×10^{-5} a.u., and -75.06314(13) means a sample mean of -75.06314 a.u. with a standard error of approximately 1.3×10^{-4} a.u.. The error of the energy is also measured by the percentage of the correlation energy recovered. As defined in Section 2.1, the correlation energy is defined to be the difference between the Hartree-Fock energy and the exact ground state energy, so that recovering 0% of the correlation energy means that the calculation produces the Hartree-Fock energy, and recovering 100%

means the calculation is exact. The correlation energy itself only contributes a tiny amount, usually less than 1%, to the ground state total energy, but capturing the correlation energy accurately is extremely important in chemistry.

Sampling and gradient clipping

We use an adaptive step width Metropolis-Hastings algorithm to sample electron configurations from the distribution defined by $\Psi(\mathbf{R})$. We use a gaussian proposal function with an isotropic step width, which we dynamically update throughout the optimization in order to keep the average acceptance ratio near a target value, for which we use 0.5. We maintain this ratio through a simple adaptive scheme that increases the step width by a small amount if the acceptance ratio strays too far above the target, and similarly decreases it by a small amount if the ratio strays too far below the target. We perform such updates every 100 moves, averaging the acceptance ratio over the previous hundred steps in order to avoid overzealously updating the step width due to noise in the acceptance ratio.

In order to reduce the amount of correlation between the samples used for subsequent parameter updates, we take 10 walker steps between each gradient calculation and parameter update. While skipping steps theoretically does not produce a higher effective sample size than simply using every step, it is practically beneficial to skip steps because the local energy calculation required for a parameter update is significantly more computationally expensive than the wave function amplitude calculation required for each move. This means we can take a number of intermediate steps in order to produce significantly less correlated samples with a small computational overhead.

As is common in quantum Monte Carlo [63], in order to reduce the noise in the training process, we additionally clip the local energies calculated in each batch of samples to be closer to some estimator of the energy intended to reduce the effect of outliers in the gradient. Specifically, given a batch of local energies $E_1, E_2, \ldots, E_n$, we calculate the median or mean local energy E_M and then calculate the average deviation from the E_M (total variation) as

$$TV = \frac{1}{n} \sum_i |E_i - E_M| \,. \tag{3.1}$$

We then replace E_i with E_M whenever $|E_i - E_M| > 5 \cdot TV$. For our atomic experiments and the H_4 model, we chose E_M to be the median local energy. In the case of the N_2 molecule, we found that choosing E_M to be the mean local energy (equal to the expected energy estimate) was more stable. In practice, we have found

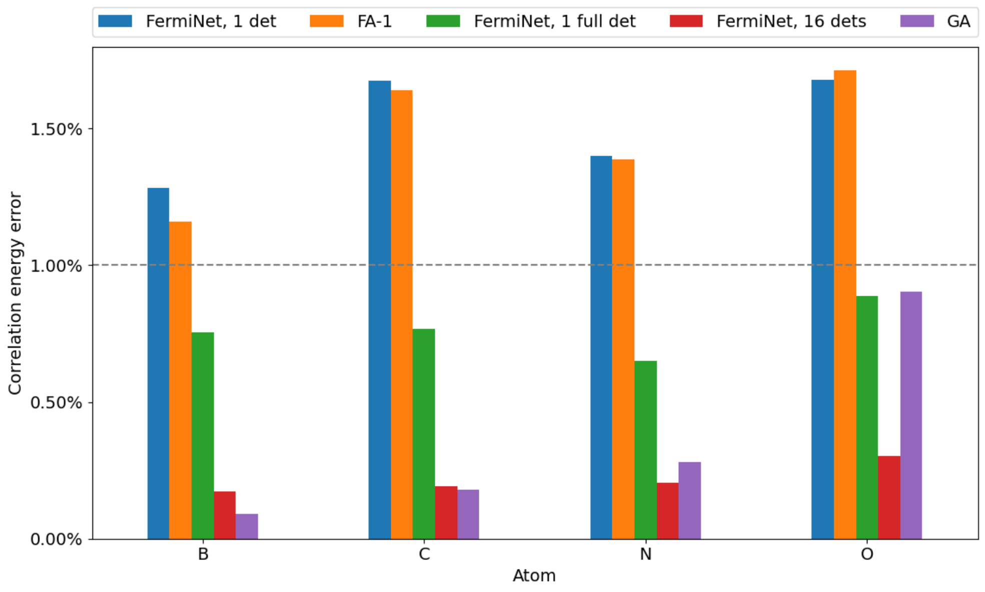


Figure 3.1: Comparison of methods on atomic systems. FA stands for factorized antisymmetric layer, and GA stands for generic antisymmetric layer, as discussed in section 2. The 16 determinant FermiNet numbers are taken from Ref. [47]. The GA result on the oxygen atom uses the more restrictive one-body Jastrow, and the parameters may not be fully optimized due to the limitations of our resources. Dashed line indicates 1% of the error in correlation energy.

that the local energy clipping produces a less noisy and more effective optimization process than including all of the unclipped local energies. During the final Monte Carlo evaluation of the energy after training, no local energy clipping is performed in order to avoid bias in the energy estimate.

3.1 Performance: atomic systems

We test our generic and factorized antisymmetric architectures on a few small atoms with nuclear charge from five to eight and compare these results to the results of FermiNet with 1 determinant, FermiNet with 1 full determinant, and FermiNet with 16 determinants. In Table 3.1, we compare the attained energies on these architectures after training with KFAC. These results are depicted as well in Figure 3.1.

Table 3.1: Comparison of methods on atomic systems. FA stands for factorized antisymmetric layer, and GA stands for generic antisymmetric layer, as discussed in section 2. The 16 determinant FermiNet and Hartree-Fock results are taken from Ref. [47].

	FermiNet 1 det	FA-1	FermiNet 1 full det	FermiNet 16 dets [47]	GA	HF [47]	Reference [11]
B	-24.65236(3)	-24.65251(2)	-24.65300(3)	-24.65370(3)	-24.65380(2)	-24.53316	-24.65391
corr %	98.71(2)%	98.84(2)%	99.25(2)%	99.83(3)%	99.91(1)%		
C	-37.84247(4)	-37.84252(3)	-37.84384(4)	-37.84471(5)	-37.84473(3)	-37.6938	-37.8450
corr %	98.33(3)%	98.36(2)%	99.23(3)%	99.81(3)%	99.82(2)%		
N	-54.58662(5)	-54.58664(4)	-54.58800(8)	-54.58882(6)	-54.58868(4)	-54.4047	-54.5892
corr %	98.60(3)%	98.61(2)%	99.35(5)%	99.79(3)%	99.72(4)%		
O	-75.06314(13)	-75.06305(6)	-75.06510(6)	-75.06655(7)	-75.06506(6)[a]	-74.8192	-75.0673
corr %	98.32(5)%	98.29(3)%	99.11(2)%	99.70(3)%	99.10(3)%		

[a]Due to the limitations of our computational resources, this result uses the simple but more restrictive one-body Jastrow from Eq. (2.35), and the parameters may not be fully optimized.

We find that the generic antisymmetric layer attains highly accurate energies when paired with the backflow-based Jastrow, achieving greater than 99.7% of the correlation energy. For the smallest systems, i.e. boron and carbon, the FermiNet-GA ansatz does at least as well as many-determinant FermiNet. For the larger systems, the performance of FermiNet-GA in our implementation began to suffer noticeably as we hit the limitations of our computational resources. For example, our result on oxygen for the generic antisymmetric architecture used only the simple one-body Jastrow in Eq. (2.35) and may not have reached the lowest energy that could be attained with additional training. We provide further discussion of the challenges with numerical stability and computational cost in Section 3.1.

Factorized antisymmetric layer of rank K versus K-determinant FermiNet

Interestingly, we do not see a gap in the attained energy between the factorized antisymmetric layer of rank 1 and single-determinant FermiNet for any system except for boron. To explore this trend further, we compare in Figure 3.2 the factorized antisymmetric layers of rank 1 through 4 against the FermiNet with 1 through 4 determinants, all on the nitrogen atom. We find that FA-K performs approximately equivalently to k-determinant FermiNet in all cases, and in most cases it performs slightly worse. This comparison is telling since replacing each generalized Slater determinant in the K-determinant FermiNet with an explicitly antisymmetrized feed-forward neural network yields exactly the FA-K architecture. The fact that this does not yield a performance improvement suggests that the reason K-determinant FermiNet is not fully general is not due to the structure of the individual generalized Slater determinants, but rather due to the sum of products structure that is used to combine the generalized Slater determinants together. This appears to be true even though the sum of products is only taken with respect to the pseudospin components generated by the backflow rather than the original spins.

We record the numerical results comparing FermiNet-FA-K to the standard K-determinant FermiNet in Table 3.2, also shown in Figure 3.2. Due to the limits of our computational resources, we did not compute the $K = 3, 4$ results for FA-K with the backflow-based Jastrow.

Hyperparameters and optimizer choice

In Table 3.3 we list the hyperparameters used in our runs for the KFAC optimizer.

We also provide additional evidence in favor of the claim in [47] that KFAC provides an advantage over Adam when optimizing FermiNet-like architectures for

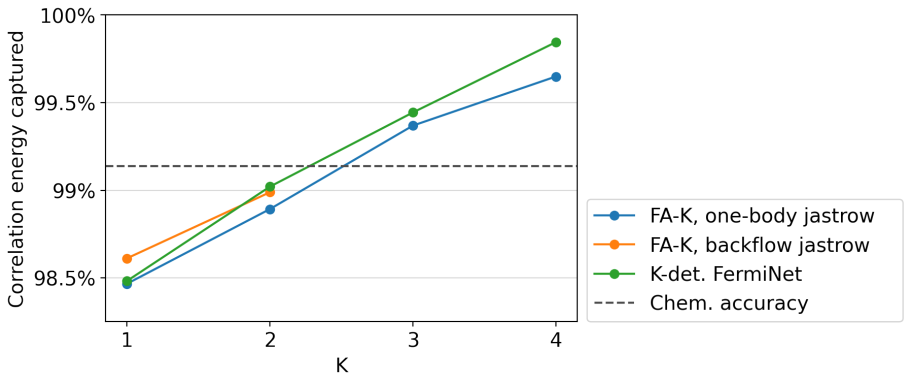
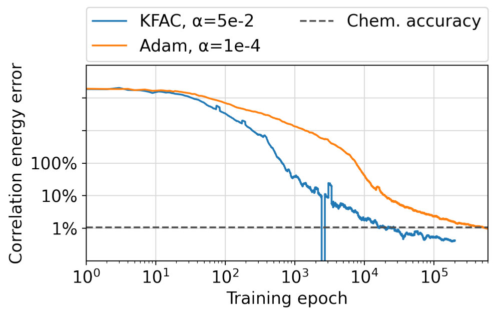
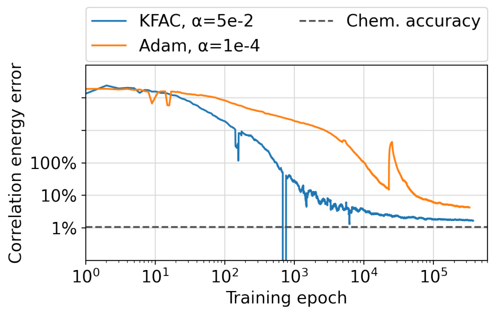


Figure 3.2: Comparison of factorized antisymmetric layer of rank K with K-determinant FermiNet for $K = 1, 2, 3, 4$ on the nitrogen atom. Data for FA-K is presented with both the simple one-body Jastrow and the more expressive backflow based Jastrow, though results for the backflow Jastrow are limited to $K \leq 2$ due to numerical stability issues and computational resource constraints. Data for multi-determinant FermiNet represent the best of several runs to account for run-to-run variance.



(a) Generic antisymmetry



(b) Factorized antisymmetric layer of rank 1

Figure 3.3: Adam versus KFAC for the optimization of the generic and factorized antisymmetry architectures on the carbon atom. The proposed updates prior to the application of the learning rate may have different scales for the two optimizers, so we choose the largest stable initial learning rate for each from a coarse sweep of learning rates α with $\log_{10} \alpha \in [-4, -1]$. At each epoch, rolling averages of the previous 10% of training epochs are shown here for clarity. One epoch is one parameter update.

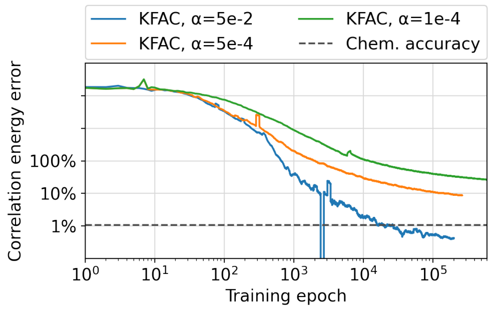
Table 3.2: Numerical results for comparison of factorized antisymmetry of rank K with K-determinant FermiNet for $K = 1, 2, 3, 4$ on the nitrogen atom. Noe that the data for multi-determinant FermiNet ($K > 1$) represent the best of several runs to account for run-to-run variance.

	FA-K, one-body Jastrow	corr%	FA-K, backflow Jastrow	corr%	K-det. FermiNet	corr%
K=1	54.58637(4)	98.47(2)	-54.58664(4)	98.61(2)	-54.5864(1)	98.48(8)
K=2	-54.58715(4)	98.89(2)	-54.58733(4)	98.99(2)	-54.58739(4)	99.02(2)
K=3	-54.58803(3)	99.37(1)	–	–	-54.58817(4)	99.44(2)
K=4	-54.58855(5)	99.65(3)	–	–	-54.58891(4)	99.85(2)

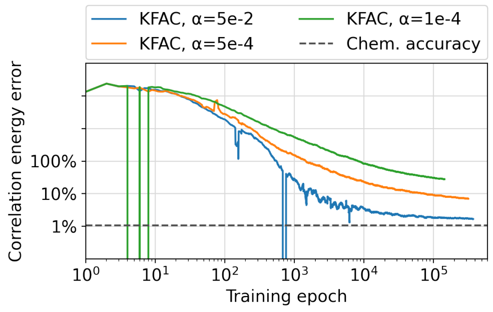
Table 3.3: Table of hyperparameters for KFAC used during training.

Hyperparameter	Value
Dense nodes per layer in antisymmetrized part	64
Layers per ResNet in antisymmetrized part	2
One-electron stream width	256
Two-electron stream width	16
Number of layers in equivariant part	4
Kernel initializers for dense layers	orthogonal
Bias initializers for dense layers	random normal
Backflow activation function	tanh
ResNet antisymmetry activation function	tanh
Jastrow (backflow) activation function	gelu
Number of walkers	2000
Learning rate	$5 \cdot 10^{-2}/(1 + 10^{-4}t)$
Optimizer	KFAC
Threshold constant for local energy clipping	5.0
MCMC steps between updates	10
Training iterations (number of parameter updates)	2e5
Evaluation iterations (samples collected every 10)	2e5

small atoms and molecules. In Figure 3.3 we provide a log-log plot of the correlation energy error during training of the generic and factorized antisymmetric architectures on the carbon atom. In this figure, the learning rate schedule for KFAC was chosen to be $5 \cdot 10^{-2}/(1+10^{-4}t)$. The learning rate schedule differed slightly for Adam, chosen instead to be $10^{-4}/(1+10^{-4}t)$. To determine the initial learning rate for both Adam and KFAC, we coarsely swept over a range of initial learning rates between 1e-4 and 1e-1 on the Carbon atom and picked the learning rate which resulted in the lowest final energies without encountering numerical instability or NaNs. The difference in the learning rates which we found were best for the two optimizers may be due to the different scale of the updates prior to the learning rate scaling. Figure 3.4 shows that the choice of learning rate is quite important, and we generally found that in our experiments, using the highest consistently stable initial learning rate resulted in the lowest energies overall. When the learning rate is chosen in this way, KFAC reaches similar energy levels to Adam with as many as two orders of magnitude fewer epochs.



(a) Generic antisymmetry



(b) Factorized antisymmetric layer of rank 1

Figure 3.4: KFAC at a few learning rates α for the optimization of the generic and factorized antisymmetry architectures on the carbon atom. At each epoch, rolling averages of the previous 10% of training epochs are shown here for clarity. One epoch is one parameter update. There is occasionally some initial instability within the first 10 or so epochs.

Numerical stability and computational cost of the antisymmetric layer

One challenge we faced when training the generic antisymmetric architecture was the numerical sign cancellation near the nodal hypersurface. When computing FermiNet-GA in single precision, we invariably encounted NaNs (not-a-number). Some investigation revealed that the computation of Ψ could yield slightly different results depending on whether it was calculated during a simple forward pass evaluation or a gradient calculation involving a forward and backward pass. Due to this numerical inconsistency, the Metropolis-Hastings procedure would sometimes sample points on or extremely close to the nodal hypersurface. To contend with this in our experiments, we used double precision end-to-end, i.e. converted all arrays to double precision. It is possible that a more efficient implementation might use double precision only in the antisymmetric layer or only when evaluating the local energy. The use of double precision led us to use A100 GPUs, which have significantly better performance for these higher precision calculations than consumer GTX GPUs. We used GTX 2080TI GPUs for our experiments which did not require double precision. Despite these powerful GPUs, the unfavorable scaling of the brute-force antisymmetry meant that we reached the limits of our group's resources with the calculations on the oxygen atom. On 4 A100 GPUs, training the FermiNet-GA architecture with the

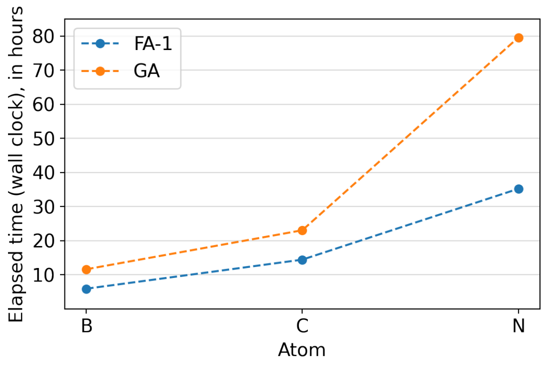


Figure 3.5: Wall clock hours elapsed during training for the generic antisymmetry and the factorized antisymmetrized of rank 1 with the backflow-based Jastrow for the boron, carbon, and nitrogen atoms. The training was performed for 2×10^5 epochs on 2 A100 GPUs.

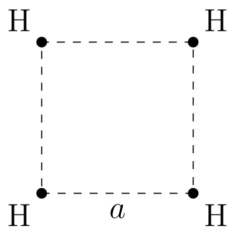


Figure 3.6: Atomic configuration for the square H_4 model. The side length is a.

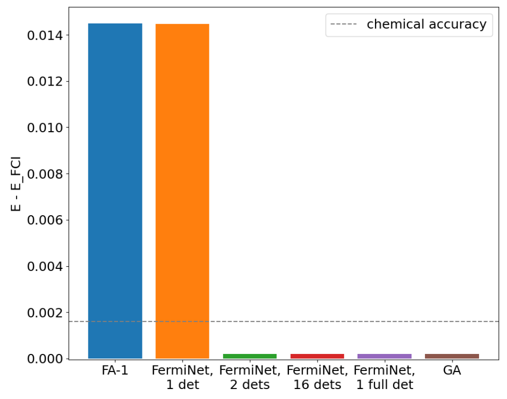
simplified Jastrow on the oxygen atom took 137 hours. In Figure 3.5, we show the wall clock time used to train the generic and factorized antisymmetric architectures for boron through nitrogen.

3.2 Performance: square H_4 model

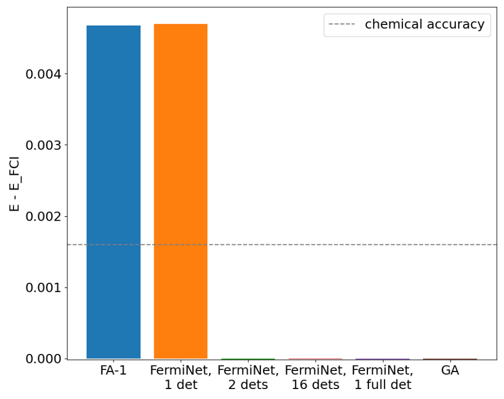
The square H_4 model (Fig. 3.6) provides an interesting case study as a prototypical strongly correlated system [30]. Unlike the atomic case, the simple product of a pseudospin-up and pseudospin-down antisymmetry is inadequate to capture the ground state within a few percent of the correlation energy. In Fig. 3.7, we see that in both FA-1 and the standard single-determinant FermiNet, the energy attained was significantly higher than that of any of the other neural network ansatzes

Table 3.4: Comparison of methods on the square H_4 model.

a (Bohr)	RHF	UHF	FA-1	FermiNet 1 det	FCI
1.0	-1.2863	-1.3358	-1.424518(7)	-1.424531(6)	-1.4390
4.0	-1.7492	-2.0092	-2.029525(4)	-2.029502(4)	-2.0342
a (Bohr)	FermiNet 2 dets	FermiNet 16 dets	FermiNet 1 full det	GA	FCI
1.0	-1.438804(5)	-1.438796(5)	-1.438805(4)	-1.438799(4)	-1.4390
4.0	-2.034212(3)	-2.034208(3)	-2.034217(2)	-2.034218(3)	-2.0342



(a) $a = 1.0$ Bohr



(b) $a = 4.0$ Bohr

Figure 3.7: Error of the total energy attained by various VMC ansatzes on the square H_4 model of different bond lengths.

tested here. We find excellent agreement between multiple-determinant FermiNet, full single-determinant FermiNet, and FermiNet-GA, agreeing within the estimated Monte-Carlo error. The failure of FA-1 to capture more of the ground-state energy than the standard single-determinant FermiNet again suggests that, at least for the small atomic and molecular systems modeled here, the FermiNet architecture is already very expressive for each pseudospin antisymmetry, even without an explicit Jastrow factor. The fact that even the addition of the general backflow-based Jastrow to the FA-1 architecture does not yield better results than FermiNet suggests that the lack of expressiveness of these simple "rank-one" product wavefunction ansatzes has to do with their nodal structure.

Basis set extrapolation

The Hartree-Fock (HF) and full configuration interaction (FCI) values for the square H_4 model were extrapolated to the complete basis limit using cc-pvXz basis sets using PySCF [57, 58]. For completeness, we reproduce the details of the extrapolation here.

The complete basis set Hartree-Fock energies were obtained by a fit to the function

$$E_{\mathrm{HF}}(X) = E_{\mathrm{HF}}(\mathrm{CBS}) + a\exp(-bX), \tag{3.2}$$

where $E_{\mathrm{HF}}(X)$ is the Hartree-Fock energy computed with cc-pvXz and the parameters $E_{\mathrm{HF}}(\mathrm{CBS})$, a, and b are determined with a non-linear least-squares fit. Similarly, the complete basis set correlation energies are obtained by a fit to the function

$$E_{\text{corr}}(X) = E_{\text{corr}}(\text{CBS}) + aX^{-3}, \tag{3.3}$$

where $E_{\text{corr}}(X)$ is the difference between the FCI and Hartree-Fock energies on the cc-pvXz basis and the parameters $E_{\text{corr}}(\text{CBS})$ and a are determined with a non-linear least-squares fit.

For RHF/UHF at bond length 1.0, we used $X = 2, 3, 4, 5$, as the orbital overlap matrix became too ill-conditioned for larger X. For RHF/UHF at bond length 4.0, we used X $= 5, 6, 8$. The FCI calculations were done using restricted Hartree-Fock (RHF) as the initial reference, and the correlation energy was computed as the difference between the FCI and RHF energies. For the extrapolation of the RHF-FCI correlation energy we used X $= 3, 4$. Due to the relative unreliability of the data points from the small double-zeta basis set and the cost of the quintuple-zeta basis set, these points were not included in the extrapolation for the correlation energy. Judging simply from the square root of the variance of the parameter fit, the basis set extrapolation error is at least two orders of magnitude larger than that of the Monte Carlo error in the estimates of the VMC-derived energies, so fewer significant digits are reported for the RHF/UHF/FCI results.

3.3 Comparison of nodal surfaces

Given a sufficiently general Jastrow correlation factor, the essential difficulty in the expressiveness of trial wavefunctions for quantum Monte Carlo methods lies in the accurate modeling of the nodal hypersurface [10]. We thus explore the nodal hypersurfaces generated by several of our ansatzes in Figures 3.8 and 3.9, taking inspiration from [10]. In these figures, we fix the locations of all but one electron in the lithium and beryllium atoms and plot the nodal surface of the resulting one-body functions in the final electron position for four of our ansatzes: standard single-determinant FermiNet, FA-1, full single-determinant FermiNet, and GA. This plotted nodal surface is thus a 3-dimensional cross-section of the full $(3N - 1)$-dimensional nodal hypersurface of the many-body wavefunction, where for example for the beryllium atom $3N - 1 = 11$.

The nodal surface of the lithium wavefunction is essentially described by the two-particle antisymmetry between the two electrons of the same spin. In this two-particle regime, Ref. [29] shows the universality of the single generalized Slater determinant. Indeed, a generic antisymmetry of two particles can be exactly written as a single two-particle determinant with an appropriately general backflow, and so

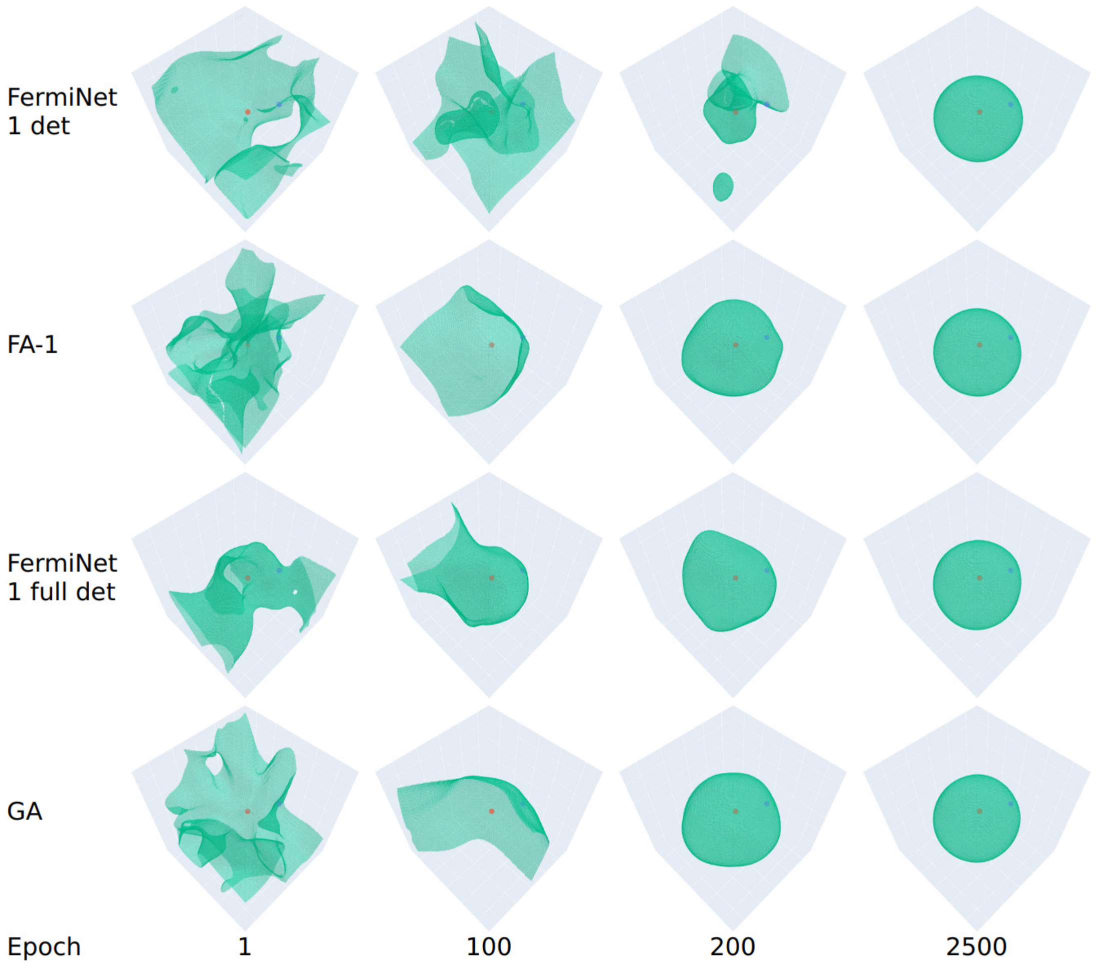


Figure 3.8: Nodal surface cross-sections of various ansatzes on the lithium atom during training. The locations of two electrons are fixed, with one spin-up (same spin) electron in blue and one spin-down (opposite spin) electron in red. The fixed electrons are in random locations. The surfaces shown are produced by evaluating the sign of the wavefunction on the position of the remaining (spin-up) electron in the box $[-5, 5]^3$ on 150 points in each direction and using the Isosurface graph object of the Plotly python graphing library.

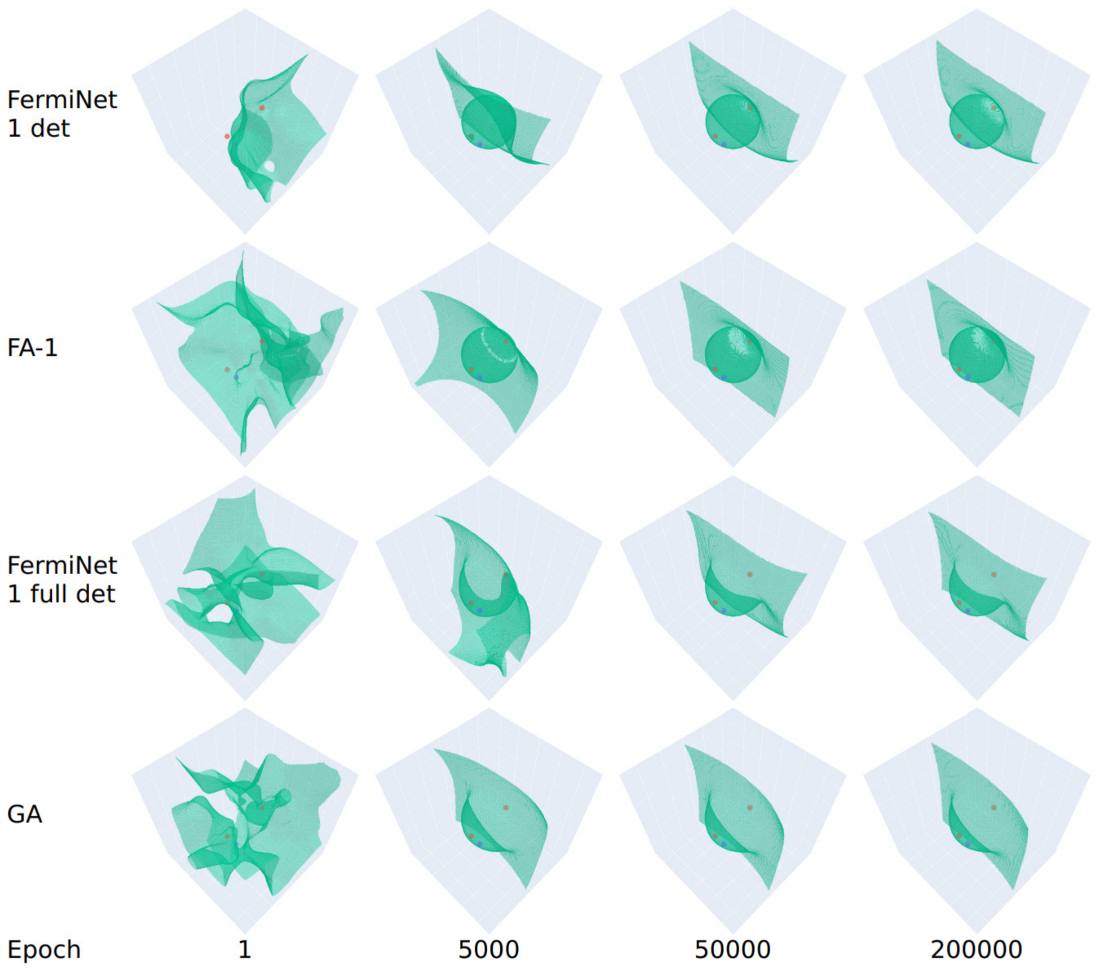


Figure 3.9: Nodal surface cross-sections of various ansatzes on the beryliium atom during training. The locations of three electrons are fixed, with one spin-up (same spin) electron in blue and two spin-down (opposite spin) electron in red. The fixed spin-up electron is placed at $(2, 1, 0)$, and the two spin-down electrons are placed at $(0, \pm 2, 0)$. The surfaces shown are produced by evaluating the sign of the wavefunction on the position of the remaining (spin-up) electron in the box $[-5, 5]^3$ on 150 points in each direction and using the Isosurface graph object of the Plotly python graphing library.

the architectures compared here are functionally equivalent in terms of their representation power. We see good agreement between the nodal surface cross-sections as early as epoch 2500 (Figure 3.8).

However, in beryllium, we observe qualitative differences between the nodal surface cross-sections between the different architectures. If we choose random locations for the three fixed electron positions, we find that the nodal surface cross-sections look much like the smooth spheres in the lithium figure for all architectures. However, in Figure 3.9 we choose the two opposite-spin electrons to be placed at $(0, \pm 2, 0)$, and we see that the nodal surface cross-sections for FermiNet and FA-1 (Figure 3.9) appear to be the union of two smooth surfaces. We were able to confirm that these two surfaces originate from the product structure of the pseudospin terms by removing a psuedospin term and replotting the resulting nodal surface. On the other hand, the nodal surface cross-section obtained from GA and the full single-determinant FermiNet appear to consist of only one smooth surface. This difference aligns with our assertion that the product structure of FermiNet and FA-1 may limit their ability to represent the true nodal surface of the ground state wavefunction. A video is available with rotating views of the final cross-sections for all four wavefunction ansatzes [1].

Our study of the nodal surfaces in this section is importantly limited by the fact that we can only observe a 3-dimensional cross-section of the full nodal surface, so we are not able to directly draw conclusions about the global structure of the nodal surface when using only a single cross-section. Ideally, we could benchmark these plots against the ground truth of the nodal surface generated by the FCI wavefunction for this system. We found, however, that even the qualitative shape of the FCI nodal surface (not depicted here) depends strongly on the choice of the finite sized basis set, and is thus difficult to compare systematically to the VMC-derived wavefunctions.

3.4 Nitrogen molecule: Performance of the full determinant FermiNet

We finally provide a comparison of a few different FermiNet architectures on the stretched nitrogen molecule, which is a challenging strongly correlated system. For our N_2 experiments, to replicate the results reported by Ref. [47] as closely as possible, 4000 walkers were used instead of 2000, a two-electron stream width of 32 was used instead of 16, pretraining was also used for 1000 iterations as it provided additional stability to the training process.

[1] `https://youtu.be/67SQXEUCYyY`

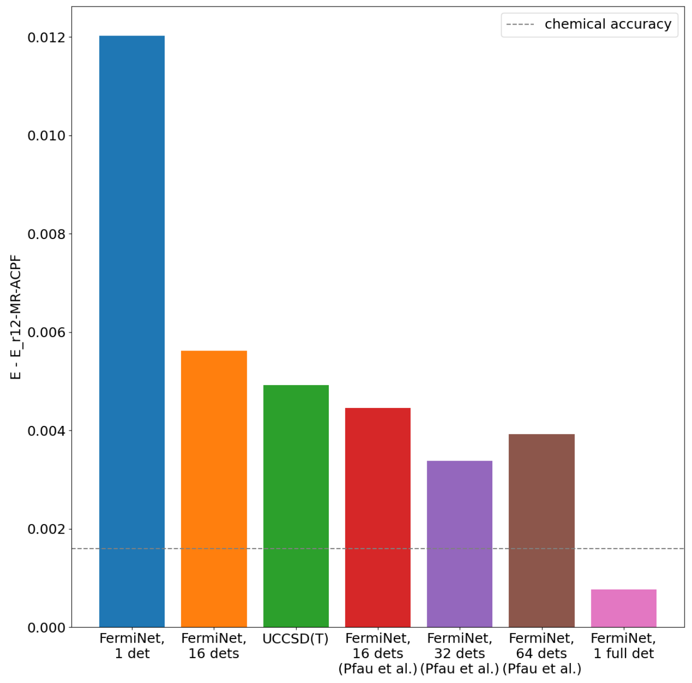


Figure 3.10: Visual comparison of different FermiNet architectures on the nitrogen molecule with bond length stretched to 4.0 Bohr. Energies shown are the absolute energy difference (in Hartrees) between the attained energy and the r_{12}-MR-ACPF computational reference [20]. The UCCSD(T) and the Pfau et al. energies were extracted from Figure 5 in reference [47]. The dashed line indicates chemical accuracy. Note that some run-to-run variance was observed, especially in the 16 determinant result, and so the result reported is the lowest energy attained from a few independent training runs.

Table 3.5: Comparison of FermiNet architectures on the nitrogen molecule with bond length stretched to 4.0. The experimental result is computed from the $\mathrm{MLR}_4(6, 8)$ fitted potential curve recommended by the authors of [36]. Note that some run-to-run variance was observed, especially in the 16 determinant result, and so the result reported is the lowest energy attained from a few independent training runs.

	energy
RHF	-108.3101
FermiNet, 1 det	-109.1827
FermiNet, 16 dets	-109.1891
FermiNet, 16 dets (Pfau et al. [47])	-109.1903
FermiNet, 32 dets (Pfau et al. [47])	-109.1913
FermiNet, 64 dets (Pfau et al. [47])	-109.1908
FermiNet, 1 full det	-109.1940
UCCSD(T) (Pfau et al. [47])	-109.1898
r_{12}-MR-ACPF [20]	-109.1947
experiment [36]	-109.2021

Pfau et al. [47] demonstrate that the standard FermiNet architecture is relatively inaccurate (compared to the same architecture at equilibrium and complete dissociation) around the dissociating bond length of 4.0 Bohr. We corroborate this finding using 1 and 16 determinants, with results reported in Figure 3.10 and in Table 3.5. Because pretraining with the Hartree-Fock wavefunction as a target has not be implemented in VMCNet as of this writing, the parameters were pretrained on the FermiNet repository, and then reloaded and trained using the VMCNet repository. The UCCSD(T) and Pfau et al. results were extracted from Figure 5 in [47].

At bond length 4.0 Bohr, the performance of the standard multiple-determinant FermiNet is improved somewhat over the standard single-determinant FermiNet, but the energy does not approach the best available computational benchmark obtained by the r_{12}-MR-ACPF method [20]. We are unable to test the FA and GA architectures on this system within the constraints of our computational resources. However, we find that the full single-determinant FermiNet is able to outperform the standard 64-determinant FermiNet and come within chemical accuracy of the computational benchmark value. An important limitation of our experiments is that we observed a non-trivial amount of run-to-run variance in our results, indicating that initialization and optimization methods for FermiNet-like architectures may require further investigation. As alluded to in the Preface, a careful understanding of this issue is important for the application of these methods to chemically relevant problems involving energy differences, especially if this optimization-related uncertainty manifests differently in (for example) different molecular geometries. A more thorough quantitative analysis of this effect is out of the reach of this dissertation. Nonetheless, we were able to replicate this result on several distinct optimization runs, and we verified the energies obtained by doing a pure MCMC evaluation with rigorous estimates of the multi-chain autocorrelation. This astonishing result implies the need for further exploration into the potential universality properties of the full single-determinant FermiNet for strongly correlated problems in quantum chemistry.

Chapter 4

VMCNet

In this chapter we describe some of the structure of VMCNet [39] and its capabilities. All numerical experiments presented in Chapter 3 with the factorized and generic antisymmetric layers are performed using the VMCNet repository, which is based on the JAX framework [6]. Using JAX allows us to leverage the implementation of DeepMind's KFAC repository[1], take advantage of the flexibility provided by JAX's clean functional style, and enjoy the performance benefits granted by its excellent out-of-the-box GPU utilization and just-in-time compilation. As of this writing, the most recent release (and thus the capabilities described here) is VMCNet version 0.1.0, for which the primary contributors were Jeffmin Lin and Gil Goldshlager.

The VMCNet repository was designed to serve two purposes:

(a) Provide a general python application programming interface (API) for variational Monte Carlo calculations compatible with JAX, with a number of built-in neural network architectures for ready-use.

(b) Provide a command-line interface exposing a large number of options for more streamlined (but somewhat less custom) experimentation with architecture/optimization/sampling hyperparameters.

In what follows, we describe a subset of the Python API, the command-line interface, and finally demonstrate that the VMCNet repository produces results on the FermiNet architecture which are comparable to that of the JAX branch of the FermiNet repository [53].

[1]`https://github.com/deepmind/deepmind-research/tree/master/kfac_ferminet_alpha`

4.1 Python API

The primary routine exposed by this repository which implements the VMC training loop is the `train.vmc.vmc_loop` function. This function implements a very generic unsupervised training loop. A skeleton of a script which performs varational Monte Carlo is shown in Listing 1.

The two primary ingredients necessary to implement the script in Listing 1 are `walker_fn`, which handles the MCMC portion and `update_param_fn`. Within these functions a number of subroutines are needed, including but not limited to trial wavefunction evaluation, MCMC routines, estimation of physical quantities, and interfacing with optimizers. We first discuss the utilities offered by VMCNet to help construct `walker_fn`, in `vmcnet.mcmc`. Next, we discuss the utilities for estimating physics-related quantities (`vmcnet.physics`) and interfacing with optimizers (`vmcnet.updates`). We then discuss some of the trial wavefunctions which have been implemented in `vmcnet.models`. Finally, we mention two simple single-parameter examples in `vmcnet.examples` (with the full loop implemented in the accompanying test suite) which can be trained quickly and studied for a sense of how to implement a custom VMC loop using the above tools. The discussion in this section is not comprehensive, and additional reference information is available on the documentation website[2].

Markov chain Monte Carlo: `vmcnet.mcmc`

Variational Monte Carlo relies on Monte Carlo sampling techniques to ameliorate the difficulty in estimating the energy and related high dimensional integrals. VMCNet provides general utilities and objects to assist with setting up the Metropolis algorithm, particularly for adaptive step sizes. Because the metropolis step created by the factories in the `metropolis` submodule is a pure function, it can be just-in-time compiled and dispatched to multiple GPUs in parallel easily using standard JAX calls to `jax.jit` and `jax.pmap`. The `PositionAmplitudeData` object in the `position_amplitude_core` submodule holds both positions and wavefunction amplitudes, as well as any additional metadata needed. It subclasses Python's `TypedDict` for excellent compatibility with JAX utilities which traverse pytrees[3].

VMCNet supports performing the Metropolis sampling on many chains of walkers in an embarrassingly parallel fashion. A set of utilities are provided by VMCNet in the `statistics` submodule to estimate the mean, variance, multi-chain integrated

[2]`https://jeffminlin.github.io/vmcnet/`

[3]`https://jax.readthedocs.io/en/latest/pytrees.html`

Listing 1 Example skeleton of a VMC script using the VMCNet Python API.

```python
import jax

import vmcnet.mcmc as mcmc
import vmcnet.train as train

# Training hyperparameters
nchains = ...
nburn = ...
nepochs = ...

seed = ...
logdir = ...
checkpoint_every = ...
checkpoint_dir = ...

# Initialize parameters, data, and optimization state
params = ...
data = ...
optimizer_state = ...

# Walker function (get new data)
def walker_fn(params, data, key):
    ...
    return accept_ratio, data, key

# Define how the parameters are updated
def update_param_fn(params, data, optimizer_state, key):
    ...
    return params, optimizer_state, metrics, key

# (Optionally) burn samples
def burning_step(params, data, key):
    ...
    return data, key

key = jax.random.PRNGKey(seed)
data, key = mcmc.metropolis.burn_data(burning_step, nburn, params, data, key)

# Train!
params, optimizer_state, data, key = train.vmc.vmc_loop(
    params,
    optimizer_state,
    data,
    nchains,
    nepochs,
    walker_fn,
    update_param_fn,
    key,
    logdir,
    checkpoint_every=checkpoint_every,
    checkpoint_dir=checkpoint_dir,
)
```

autocorrelation, and standard error. Geyer's initial minimum sequence [8, 21, 54] is used to estimate the integrated autocorrelation.

Physical quantities: `vmcnet.physics`

Once a set of samples following the trial wavefunction distribution have been obtained and before a parameter update can be determined, gradient-based optimizers such as stochastic gradient descent and KFAC require an estimate of the gradient of the expected energy. In VMCNet, the terms of the Born-Oppenheimer Hamiltonian have been implemented separately to allow for easy implmentation of custom Hamiltonians, e.g. lattice Hamiltonians. An energy gradient function is implemented which by default supports the calculation in Eq. (1.37), and a custom post-processing function can be applied on the local energies (e.g. a clipping function).

Optimizer interface: `vmcnet.updates`

After the gradient of the expected energy has been computed, it can be passed to an optimizer which updates the parameters. VMCNet supports arbitrary optimizers that map (gradient, parameters, optimizer state, data) to (new parameters, new optimizer state). In particular, factories to create update functions for the Adam and Stochastic Gradient Descent optimizers from the `optax` package have been implemented in the `parse_config` submodule.

An experimental stochastic reconfiguration (SR) implementation is available. The SR implementation uses the approximation to the Fisher/quantum geometric tensor which is available in expectation from the samples collected by the Monte Carlo portion of the VMC loop, which has rank bounded by the number of samples. Depending on the parametrization, the approximate numerical rank of the Fisher may be much smaller than the number of samples, and so when using the inverse Fisher as a gradient preconditioner in the stochastic reconfiguration method, a small damping constant is added to improve the stability of the method. The Fisher is then approximately inverted using the conjugate gradient method implemented in `jax.scipy`.

In addition to the provided interface with `optax` optimizers and the SR implementation, utilities to create parameter update functions which call an instance of the KFAC optimizer from DeepMind's KFAC repository[4] are provided. The wavefunctions implemented in `vmcnet.models` support this use of the DeepMind KFAC repository, and custom models built from `vmcnet.models.core.Dense` or

[4]`https://github.com/deepmind/deepmind-research/tree/master/kfac_ferminet_alpha`

`vmcnet.models.core.SimpleResNet` will automatically be compatible in this way as well.

Trial wavefunctions: `vmcnet.models`

The models in VMCNet have been built using the open source Flax library [26]. Usage of the Flax library grants flexibility in model architecture, an API similar to other popular libraries such as Keras [12], and streamlined factories to produce pure JAX-compatible functions which perform model evaluation and initialization. As alluded to previously, VMCNet provides custom `Dense` and `SimpleResNet` Flax Modules in the `core` submodule which are interfaced with the DeepMind KFAC repository[5].

Multiple variants of the FermiNet architecture have been implemented in VMCNet, with variations on the backflow, input, and determinant layer. These include the standard and full determinant modes referenced in Chapter 2. The factorized and generic antisymmetric layers are also implemented using an underlying ResNet [25] (results in Chapter 3 use a single hidden layer, so we refer to them as a feed forward neural network).

VMCNet also makes it straightforward to add a Jastrow factor of a few types, including a one-body Jastrow (Eq. (2.35)), a two-body Jastrow, and a backflow-based Jastrow (Eq. (2.36)). In the default model configuration dictionaries, these are available for the FA and GA models, but they may be added to other models as well. An involved example of adding a backflow-based Jastrow factor used with the FA or GA models to the standard FermiNet for the boron atom is shown in Listing 2.

Examples: `vmcnet.examples` and tests

A couple of simple examples of end-to-end VMC loops which combine parts of previously discussed components of VMCNet and custom models are available in the `vmcnet.examples` module and the integration test suite included in the GitHub repository [39]. Models and energy functionals for the harmonic oscillator and hydrogen-like atom examples discussed in Chapter 1 are implemented, and both a simple stochastic gradient descent and KFAC optimizer are demonstrated in the integration test suite. In Listings 3.1, 3.2, and 3.3, pieces from the test suite are combined into a single script which demonstrates a complete VMC example on the quantum harmonic oscillator with five independent particles.

[5]`https://github.com/deepmind/deepmind-research/tree/master/kfac_ferminet_alpha`

Listing 2 Adding a backflow-based Jastrow factor to the standard FermiNet.

```python
from typing import Callable

import flax
import jax.numpy as jnp
from ml_collections import ConfigDict

import vmcnet.models as models
import vmcnet.train as train
from vmcnet.utils.typing import Array, ComputeInputStreams, Jastrow, SLArray


class FermiNetJastrow(models.core.Module):
    """FermiNet with a backflow-based Jastrow."""

    ferminet_model: Callable[[Array], SLArray]
    compute_jastrow_input: ComputeInputStreams
    jastrow: Jastrow

    @flax.linen.compact
    def __call__(self, elec_pos: Array) -> SLArray:
        """Add the log Jastrow output to the log FermiNet output."""
        sign_ferminet, log_abs_ferminet = self.ferminet_model(elec_pos)
        input_1e, input_2e, _, _ = self.compute_jastrow_input(elec_pos)
        jastrow_out = self.jastrow(input_1e, input_2e, None, None, None)
        return sign_ferminet, log_abs_ferminet + jastrow_out


# specify the boron atom
nelec = jnp.array([3, 2])
ion_pos, ion_charges = jnp.array([[0.0, 0.0, 0.0]]), jnp.array([5.0])
model_config = ConfigDict(train.default_config.get_default_model_config())

# make the default FermiNet model
ferminet_config = train.default_config.choose_model_type_in_model_config(model_config)
ferminet_model = models.construct.get_model_from_config(
    ferminet_config, nelec, ion_pos, ion_charges
)

# get the default backflow-based Jastrow from the explicit antisymmetry models
model_config.type = "explicit_antisym"
antisym_config = train.default_config.choose_model_type_in_model_config(model_config)
compute_jastrow_input = models.construct.get_compute_input_streams_from_config(
    antisym_config.input_streams, ion_pos
)
jastrow_config = antisym_config.jastrow
jastrow_backflow = models.construct.get_backflow_from_config(
    jastrow_config.backflow_based.backflow, (nelec[0],)
)
jastrow = models.jastrow.BackflowJastrow(backflow=jastrow_backflow)

# combine the models
ferminet_with_jastrow = FermiNetJastrow(ferminet_model, compute_jastrow_input, jastrow)
```

Listing 3.1 Harmonic oscillator VMC with five independent particles. (part 1/3)

```python
import logging

import jax
import jax.numpy as jnp
import numpy as np

import vmcnet.examples.harmonic_oscillator as qho
import vmcnet.mcmc as mcmc
import vmcnet.physics as physics
import vmcnet.train as train
import vmcnet.updates as updates
from vmcnet.mcmc.position_amplitude_core import get_position_from_data
from vmcnet.mcmc.simple_position_amplitude import (
    make_simple_pos_amp_gaussian_step,
    make_simple_position_amplitude_data,
)


def _make_initial_params_and_data(model_omega, nchains):
    key = jax.random.PRNGKey(0)
    key, subkey = jax.random.split(key)
    random_particle_positions = jax.random.normal(subkey, shape=(nchains, 5, 1))

    # because there are 5 particles total, the spin split is (3, 2)
    log_psi_model = qho.make_harmonic_oscillator_spin_half_model(2, model_omega)

    key, subkey = jax.random.split(key)
    params = log_psi_model.init(subkey, random_particle_positions)
    amplitudes = log_psi_model.apply(params, random_particle_positions)
    return log_psi_model, params, random_particle_positions, amplitudes, key
```

Listing 3.2 Harmonic oscillator VMC with five independent particles. (part 2/3)

```python
def sgd_vmc_loop_with_logging(
    data,
    params,
    key,
    nchains,
    nburn,
    nepochs,
    nsteps_per_param_update,
    std_move,
    optimizer_state,
    log_psi_model,
    local_energy_fn,
):
    """Run a VMC test with a very simple SGD optimizer and given model."""
    # Setup metropolis step
    metrop_step_fn = make_simple_pos_amp_gaussian_step(log_psi_model.apply, std_move)

    burning_step = mcmc.metropolis.make_jitted_burning_step(
        metrop_step_fn, apply_pmap=False
    )
    walker_fn = mcmc.metropolis.make_jitted_walker_fn(
        nsteps_per_param_update, metrop_step_fn, apply_pmap=False
    )

    # Define parameter updates
    def sgd_apply(grad, params, learning_rate, data):
        del data
        return (
            jax.tree_map(lambda a, b: a - learning_rate * b, params, grad),
            learning_rate,
        )

    energy_data_val_and_grad = physics.core.create_value_and_grad_energy_fn(
        log_psi_model.apply, local_energy_fn, nchains
    )
    update_param_fn = updates.params.create_grad_energy_update_param_fn(
        energy_data_val_and_grad, sgd_apply, get_position_from_data, apply_pmap=False
    )

    # Train!
    logging.getLogger().setLevel("INFO")
    data, key = mcmc.metropolis.burn_data(burning_step, nburn, params, data, key)
    params, optimizer_state, data, key = train.vmc.vmc_loop(
        params, optimizer_state, data, nchains, nepochs, walker_fn, update_param_fn, key
    )
    return data, params, optimizer_state, key
```

Listing 3.3 Harmonic oscillator VMC with five independent particles. (part 3/3)

```python
def test_harmonic_oscillator_vmc():
    """Test that the trainable sqrt(omega) converges to the true sqrt(spring constant)."""
    # Problem parameters
    model_omega = 2.5
    spring_constant = 1.5

    # Training hyperparameters
    nchains = 100 * jax.local_device_count()
    nburn = 100
    nepochs = 50
    nsteps_per_param_update = 5
    std_move = 0.25
    learning_rate = 1e-2

    # Initialize model and chains of walkers
    (
        log_psi_model,
        params,
        random_particle_positions,
        amplitudes,
        key,
    ) = _make_initial_params_and_data(model_omega, nchains)
    data = make_simple_position_amplitude_data(random_particle_positions, amplitudes)

    # Local energy function
    local_energy_fn = qho.make_harmonic_oscillator_local_energy(
        spring_constant, log_psi_model.apply
    )

    _, params, _, _ = sgd_vmc_loop_with_logging(
        data,
        params,
        key,
        nchains,
        nburn,
        nepochs,
        nsteps_per_param_update,
        std_move,
        learning_rate,
        log_psi_model,
        local_energy_fn,
    )

    # Grab the one parameter and make sure it converged to sqrt(spring constant)
    np.testing.assert_allclose(
        jax.tree_leaves(params)[0], jnp.sqrt(spring_constant), rtol=1e-6
    )


if __name__ == "__main__":
    test_harmonic_oscillator_vmc()
```

Listing 4 Example of a simple command-line call to run VMC on the nitrogen molecule at bond length 4.0 Bohr.

```
vmc-molecule \
    --config.problem.ion_pos="((0.0, 0.0, -2.0), (0.0, 0.0, 2.0))" \
    --config.problem.ion_charges="(7.0, 7.0)" \
    --config.problem.nelec="(7, 7)" \
    --config.model.ferminet.full_det="True" \
    --config.logging_level="INFO"
```

4.2 Command-line

Alternatively, a command-line interface has been implemented which provides more streamlined access to subsets of the repository via setting ConfigDict objects. There are two scripts which have been exposed thus far: `vmc-molecule` and `vmc-statistics`.

The primary command `vmc-molecule` calls `train.runners.run_molecule`. See `train.default_config.get_default_config()` to explore the options which have been exposed to the command-line. To edit these options at the command-line, use the "`--config.`" prefix. For example, the command in Listing 4 will train the full single-determinant FermiNet on the nitrogen molecule at dissociating bond length 4.0 Bohr for 2e5 epochs on 2000 walkers (which are distributed across any available GPUs, if supported by the installation). The user can also reload and evaluate or continue training from previous checkpoints via the "`--reload.`" prefix. Training options can be seen in `train.default_config.get_default_reload_config()`. The reloading will only occur if `--reload.logdir` is set.

The `vmc-statistics` command calls `train.runners.vmc_statistics`. This simple script is designed to be compatible with the output of an evaluation run with `vmc-molecule`, but can accept any path to a file which contains local energies (a file with nchains x nepochs energies). It computes and saves a json file containing the average energy, the sample variance, the estimated integrated autocorrelation, and the estimated standard error. The options can be viewed simply via `vmc-statistics -h`.

4.3 Code benchmarking

To demonstrate that our results for the original FermiNet are comparable to those reported by [47, 53], we show that results obtained using the VMCNet repository are quantitatively comparable to that of the JAX branch of the FermiNet repository presented in [53] on several small systems. We compare the behavior on both the

Table 4.1: Comparison of the VMCNet repository with the FermiNet repository on the nitrogen atom, with 1, 2, and 4 determinants. Data for 2 and 4 determinants represent the best of several runs to account for run-to-run variance observed in both repositories.

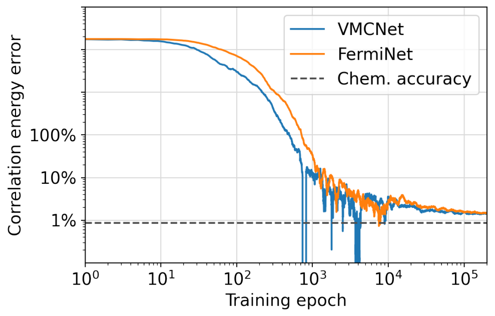
Repository	1 det	2 det	4 det
VMCNet	-54.5864(1)	-54.58739(4)	-54.58891(4)
corr %	98.48(8)%	99.02(2)%	99.85(2)%
FermiNet	-54.58654(5)	-54.58711(6)	-54.58870(4)
corr %	98.56(3)%	98.87(3)%	99.73(4)%

Table 4.2: Comparison of the VMCNet repository with the FermiNet repository on the H_4 square, with 1 and 2 determinants.

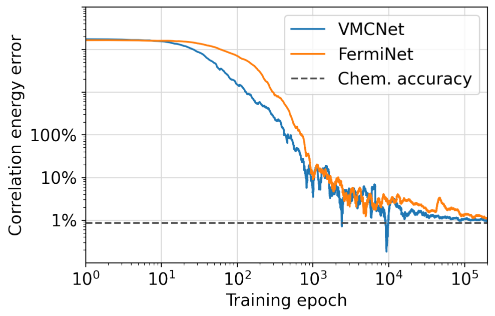
Repository	1 det	2 det
VMCNet	-1.424531(7)	-1.438804(5)
FermiNet	-1.424429(7)	-1.438796(5)

nitrogen atom and the square H_4 model (Figure 3.6), using settings corresponding to the original FermiNet model in both repositories. For the nitrogen atom, we compare results with 1, 2, and 4 determinants, while for the H_4 square we compare results with just 1 and 2 determinants, since 2 determinants already captures essentially 100% of the correlation energy. All results presented here come from our own numerical experiments with either the VMCNet repository or the publicly available JAX branch of the FermiNet repository. Since VMCNet does not support Hartree-Fock based pretraining, we turned this feature off in the FermiNet repository to make the comparison fair. Turning off pretraining reduces the consistency of the FermiNet optimization on some systems. In particular, when using multiple determinants for the nitrogen atom, we found that some runs both of our own code and of the FermiNet code without pretraining get stuck in local minima and never reach the lowest energy possible. This phenomenon may merit further investigation. For now, to account for this run-to-run variance, we have taken the best of several runs for all multi-determinant experiments on the nitrogen atom.

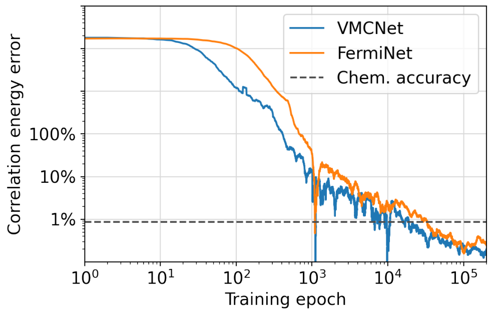
Representative training graphs can be found for the nitrogen atom in Figure 4.1 and for the H_4 square in Figure 4.2. The values of the final energies obtained are



(a) 1 determinant



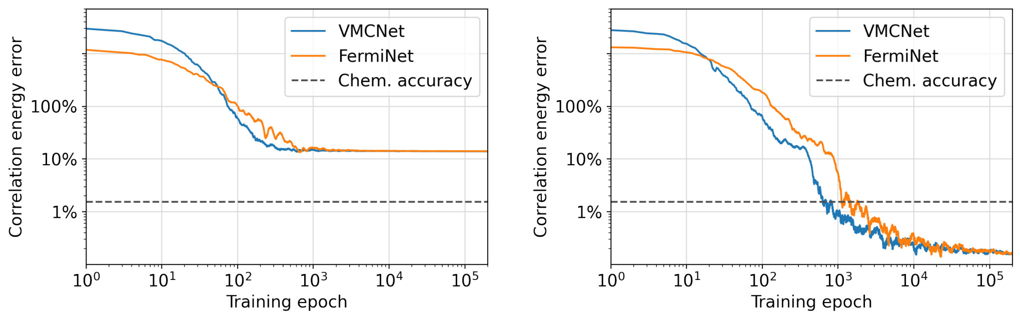
(b) 2 determinants



(c) 4 determinants

Figure 4.1: Training runs on VMCNet and FermiNet repositories with the FermiNet architecture on the nitrogen atom. At each epoch, rolling averages of the previous 10% of training epochs are shown here for clarity. One epoch means one parameter update. Data for 2 and 4 determinants represent the best of several runs to account for run-to-run variance observed in both repositories.

presented in Tables 4.1 and 4.2, respectively. On both systems, the results of VMCNet are approximately equivalent to the results of FermiNet. The two repositories behave somewhat differently in the first 1,000 epochs of training, with VMCNet often optimizing more quickly in this regime. However, the optimization trajectories are largely indistinguishable by 10,000 epochs and the final energies achieved are within a small margin of error of each other in all cases.



(a) 1 determinant (b) 2 determinants

Figure 4.2: Training runs on VMCNet and FermiNet repositories with the FermiNet architecture on the H4 square. At each epoch, rolling averages of the previous 10% of training epochs are shown here for clarity. One epoch means one parameter update.

Bibliography

[1] Shun-ichi Amari. "Natural Gradient Works Efficiently in Learning". In: *Neural Computation* 10.2 (Feb. 1998), pp. 251–276. ISSN: 0899-7667. DOI: 10.1162/089976698300017746.

[2] Shun-ichi Amari and Hiroshi Nagaoka. "Methods of information geometry". In: vol. 191. Translations of Mathematical monographs. Oxford University Press, 2000.

[3] Markus Bachmayr, Geneviève Dusson, and Christoph Ortner. "Polynomial Approximation of Symmetric Functions". In: *arXiv:2109.14771* (2021).

[4] Andrew R Barron. "Approximation and estimation bounds for artificial neural networks". In: *Machine Learning* 14.1 (1994), pp. 115–133.

[5] Federico Becca and Sandro Sorella. *Quantum Monte Carlo Approaches for Correlated Systems.* Cambridge University Press, 2017. DOI: 10.1017/9781316417041.

[6] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs.* Version 0.2.5. 2018. URL: http://github.com/google/jax.

[7] Dario Bressanini and Peter J. Reynolds. "Between Classical and Quantum Monte Carlo Methods: "Variational" QMC". In: *Advances in Chemical Physics.* John Wiley & Sons, Ltd, 1999, pp. 37–64. ISBN: 9780470141649. DOI: 10.1002/9780470141649.ch3. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780470141649.ch3.

[8] Steve Brooks et al., eds. *Handbook of Markov Chain Monte Carlo (Chapman & Hall/CRC Handbooks of Modern Statistical Methods).* English. Hardcover. Chapman and Hall/CRC, May 10, 2011, p. 619. ISBN: 978-1420079418.

[9] Giuseppe Carleo and Matthias Troyer. "Solving the quantum many-body problem with artificial neural networks". In: *Science* 355.6325 (Feb. 2017), pp. 602–606. ISSN: 1095-9203. DOI: 10.1126/science.aag2302.

[10] David M Ceperley. "Fermion nodes". In: *Journal of statistical physics* 63.5 (1991), pp. 1237–1267.

[11] Subhas J. Chakravorty et al. "Ground-state correlation energies for atomic ions with 3 to 18 electrons". In: *Phys. Rev. A* 47 (5 May 1993), pp. 3649–3670. DOI: `10.1103/PhysRevA.47.3649`.

[12] François Chollet et al. *Keras*. `https://keras.io`. 2015.

[13] Kenny Choo, Antonio Mezzacapo, and Giuseppe Carleo. "Fermionic neural-network states for ab-initio electronic structure". In: *Nature Communications* 11.1 (May 2020), p. 2368. ISSN: 2041-1723. DOI: `10.1038/s41467-020-15724-9`.

[14] Kenny Choo et al. "Symmetries and Many-Body Excitations with Neural-Network Quantum States". In: *Phys. Rev. Lett.* 121 (16 Oct. 2018), p. 167204. DOI: `10.1103/PhysRevLett.121.167204`.

[15] Robert Lynn Coldwell. "Zero Monte Carlo error or quantum mechanics is easier". In: *International Journal of Quantum Chemistry* 12.S11 (1977), pp. 215–222. DOI: `https://doi.org/10.1002/qua.560120826`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/qua.560120826`.

[16] N. D. Drummond, M. D. Towler, and R. J. Needs. "Jastrow correlation factor for atoms, molecules, and solids". In: *Phys. Rev. B* 70 (23 Dec. 2004), p. 235119. DOI: `10.1103/PhysRevB.70.235119`. URL: `https://link.aps.org/doi/10.1103/PhysRevB.70.235119`.

[17] Dennis Elbrächter et al. *Deep Neural Network Approximation Theory*. 2021. arXiv: `1901.02220 [cs.LG]`.

[18] R. P. Feynman and Michael Cohen. "Energy Spectrum of the Excitations in Liquid Helium". In: *Phys. Rev.* 102 (5 June 1956), pp. 1189–1204. DOI: `10.1103/PhysRev.102.1189`.

[19] W. M. C. Foulkes et al. "Quantum Monte Carlo simulations of solids". In: *Rev. Mod. Phys.* 73 (1 Jan. 2001), pp. 33–83. DOI: `10.1103/RevModPhys.73.33`.

[20] Robert J. Gdanitz. "Accurately solving the electronic Schrödinger equation of atoms and molecules using explicitly correlated (r12-)MR-CI: the ground state potential energy curve of N2". In: *Chemical Physics Letters* 283.5 (1998), pp. 253–261. ISSN: 0009-2614. DOI: `https://doi.org/10.1016/S0009-2614(97)01392-4`.

[21] Charles J. Geyer. "Practical Markov Chain Monte Carlo". In: *Statistical Science* 7.4 (1992), pp. 473–483. DOI: `10.1214/ss/1177011137`. URL: `https://doi.org/10.1214/ss/1177011137`.

[22] James Gubernatis, Naoki Kawashima, and Philipp Werner. *Quantum Monte Carlo Methods*. Cambridge Univ. Pr., 2016.

[23] Jiequn Han, Linfeng Zhang, and Weinan E. "Solving many-electron Schrödinger equation using deep neural networks". In: *Journal of Computational Physics* 399 (2019), p. 108929. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2019.108929.

[24] Jiequn Han et al. "Universal approximation of symmetric and anti-symmetric functions". In: *arXiv:1912.01765* (2019).

[25] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016, pp. 770–778.

[26] Jonathan Heek et al. *Flax: A neural network library and ecosystem for JAX.* Version 0.3.5. 2020. URL: http://github.com/google/flax.

[27] Dan Hendrycks and Kevin Gimpel. *Gaussian Error Linear Units (GELUs).* 2016. arXiv: 1606.08415 [cs.LG].

[28] Jan Hermann, Zeno Schätzle, and Frank Noé. "Deep-neural-network solution of the electronic Schrödinger equation". In: *Nature Chemistry* 12.10 (Sept. 2020), pp. 891–897. ISSN: 1755-4349. DOI: 10.1038/s41557-020-0544-y.

[29] Marcus Hutter. *On Representing (Anti)Symmetric Functions.* 2020. DOI: 10.48550/ARXIV.2007.15298. URL: https://arxiv.org/abs/2007.15298.

[30] K Jankowski and J Paldus. "Applicability of coupled-pair theories to quasidegenerate electronic states: A model study". In: *Int. J. Quantum Chem.* 18.5 (1980), pp. 1243–1269.

[31] T. Kato. "On the eigenfunctions of many-particle systems in quantum mechanics". In: *Commun. Pure Appl. Math.* 10 (1957), p. 151.

[32] P. R. C. Kent, R. J. Needs, and G. Rajagopal. "Monte Carlo energy and variance-minimization techniques for optimizing many-body wave functions". In: *Phys. Rev. B* 59 (19 May 1999), pp. 12344–12351. DOI: 10.1103/PhysRevB.59.12344.

[33] Nicolas Keriven and Gabriel Peyré. "Universal invariant and equivariant graph neural networks". In: *Advances in Neural Information Processing Systems* 32 (2019), pp. 7092–7101.

[34] Jan Kessler, Francesco Calcavecchia, and Thomas D Kühne. "Artificial neural networks as trial wave functions for quantum monte carlo". In: *Advanced Theory and Simulations* 4.4 (2021), p. 2000269.

[35] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations.* 2015.

[36] Robert J. Le Roy, Yiye Huang, and Calvin Jary. "An accurate analytic potential function for ground-state N2 from a direct-potential-fit analysis of spectroscopic data". In: *The Journal of Chemical Physics* 125.16 (2006), p. 164310. DOI: 10.1063/1.2354502.

[37] Henry W. Lin, Max Tegmark, and David Rolnick. "Why Does Deep and Cheap Learning Work So Well?" In: *Journal of Statistical Physics* 168.6 (July 2017), pp. 1223–1247. ISSN: 1572-9613. DOI: 10.1007/s10955-017-1836-5. URL: http://dx.doi.org/10.1007/s10955-017-1836-5.

[38] Jeffmin Lin, Gil Goldshlager, and Lin Lin. *Explicitly antisymmetrized neural network layers for variational Monte Carlo simulation.* 2021. DOI: 10.48550/ARXIV.2112.03491. URL: https://arxiv.org/abs/2112.03491.

[39] Jeffmin Lin, Gil Goldshlager, and Lin Lin. *VMCNet: Flexible, general-purpose VMC framework, built on JAX.* Version 0.1.0. 2021. URL: http://github.com/jeffminlin/vmcnet.

[40] P. López Ríos et al. "Inhomogeneous backflow transformations in quantum Monte Carlo calculations". In: *Phys. Rev. E* 74 (6 Dec. 2006), p. 066701. DOI: 10.1103/PhysRevE.74.066701. URL: https://link.aps.org/doi/10.1103/PhysRevE.74.066701.

[41] Di Luo and Bryan K. Clark. "Backflow Transformations via Neural Networks for Quantum Many-Body Wave Functions". In: *Phys. Rev. Lett.* 122 (22 June 2019), p. 226401. DOI: 10.1103/PhysRevLett.122.226401.

[42] James Martens and Roger Grosse. "Optimizing Neural Networks with Kronecker-factored Approximate Curvature". In: *Proceedings of the 32nd International Conference on Machine Learning.* Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, 2015, pp. 2408–2417.

[43] Alexandra Nagy and Vincenzo Savona. "Variational Quantum Monte Carlo Method with a Neural-Network Ansatz for Open Quantum Systems". In: *Phys. Rev. Lett.* 122 (25 June 2019), p. 250501. DOI: 10.1103/PhysRevLett.122.250501.

[44] Eric Neuscamman, CJ Umrigar, and Garnet Kin-Lic Chan. "Optimizing large parameter sets in variational quantum Monte Carlo". In: *Phys. Rev. B* 85.4 (2012), p. 045103.

[45] Yusuke Nomura et al. "Restricted Boltzmann machine learning for solving strongly correlated quantum systems". In: *Phys. Rev. B* 96 (20 Nov. 2017), p. 205152. DOI: 10.1103/PhysRevB.96.205152.

[46] Leon Otis and Eric Neuscamman. “Complementary first and second derivative methods for ansatz optimization in variational Monte Carlo”. In: *Phys. Chem. Chem. Phys.* 21.27 (2019), pp. 14491–14510.

[47] David Pfau et al. “Ab initio solution of the many-electron Schrödinger equation with deep neural networks”. In: *Physical Review Research* 2.3 (Sept. 2020). ISSN: 2643-1564. DOI: `10.1103/physrevresearch.2.033429`.

[48] Allan Pinkus. “Approximation theory of the MLP model in neural networks”. In: *Acta Numerica* 8 (1999), pp. 143–195. DOI: `10.1017/S0962492900002919`.

[49] Michael Reed and Barry Simon. *II: Fourier Analysis, Self-Adjointness (Methods of Modern Mathematical Physics, Volume 2).* Academic Press, Oct. 1975. ISBN: 978-0125850025.

[50] David Rolnick and Max Tegmark. “The power of deeper networks for expressing natural functions”. In: *International Conference on Learning Representations.* 2018. URL: `https://openreview.net/forum?id=SyProzZAW`.

[51] Iliya Sabzevari, Ankit Mahajan, and Sandeep Sharma. “An accelerated linear method for optimizing non-linear wavefunctions in variational Monte Carlo”. In: *J. Chem. Phys.* 152.2 (2020), p. 024111.

[52] Akiyoshi Sannai, Yuuki Takai, and Matthieu Cordonnier. “Universal approximations of permutation invariant/equivariant functions by deep neural networks”. In: *arXiv:1903.01939* (2019).

[53] James S. Spencer et al. *Better, Faster Fermionic Neural Networks.* 2020. arXiv: `2011.07125 [physics.comp-ph]`.

[54] *Stan Reference Manual.* Version 2.29. Stan Development Team. URL: `https://mc-stan.org/docs/2_29/reference-manual/index.html`.

[55] James Stokes et al. “Phases of two-dimensional spinless lattice fermions with first-quantized deep neural-network quantum states”. In: *Phys. Rev. B* 102 (20 Nov. 2020), p. 205122. DOI: `10.1103/PhysRevB.102.205122`.

[56] James Stokes et al. “Quantum Natural Gradient”. In: *Quantum* 4 (May 2020), p. 269. ISSN: 2521-327X. DOI: `10.22331/q-2020-05-25-269`.

[57] Qiming Sun et al. “PySCF: the Python-based simulations of chemistry framework”. In: *WIREs Computational Molecular Science* 8.1 (2018), e1340. DOI: `10.1002/wcms.1340`.

[58] Qiming Sun et al. “Recent developments in the PySCF program package”. In: *The Journal of Chemical Physics* 153.2 (2020), p. 024109. DOI: `10.1063/5.0006074`.

[59] A. Szabo and N.S. Ostlund. *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory.* McGraw-Hill, New York, 1989.

[60] Luca F. Tocchio et al. “Role of backflow correlations for the nonmagnetic phase of the t–t' Hubbard model”. In: *Phys. Rev. B* 78 (4 July 2008), p. 041101. DOI: 10.1103/PhysRevB.78.041101.

[61] Julien Toulouse, Roland Assaraf, and Cyrus J. Umrigar. “Chapter Fifteen - Introduction to the Variational and Diffusion Monte Carlo Methods”. In: *Electron Correlation in Molecules – ab initio Beyond Gaussian Quantum Chemistry.* Ed. by Philip E. Hoggan and Telhat Ozdogan. Vol. 73. Advances in Quantum Chemistry. Academic Press, 2016, pp. 285–314. DOI: 10.1016/bs.aiq.2015.07.003. URL: https://www.sciencedirect.com/science/article/pii/S0065327615000386.

[62] C. J. Umrigar and Claudia Filippi. “Energy and Variance Optimization of Many-Body Wave Functions”. In: *Phys. Rev. Lett.* 94 (15 Apr. 2005), p. 150201. DOI: 10.1103/PhysRevLett.94.150201.

[63] C. J. Umrigar, M. P. Nightingale, and K. J. Runge. “A diffusion Monte Carlo algorithm with very small time-step errors”. In: *The Journal of Chemical Physics* 99.4 (1993), pp. 2865–2890. DOI: 10.1063/1.465195.

[64] C. J. Umrigar, K. G. Wilson, and J. W. Wilkins. “Optimized trial wave functions for quantum Monte Carlo calculations”. In: *Phys. Rev. Lett.* 60 (17 Apr. 1988), pp. 1719–1722. DOI: 10.1103/PhysRevLett.60.1719.

[65] CJ Umrigar and Claudia Filippi. “Energy and variance optimization of many-body wave functions”. In: *Phys. Rev. Lett.* 94.15 (2005), p. 150201.

[66] Li Yang et al. “Deep learning-enhanced variational Monte Carlo method for quantum many-body physics”. In: *Physical Review Research* 2.1 (Feb. 2020). ISSN: 2643-1564. DOI: 10.1103/physrevresearch.2.012039.

[67] Manzil Zaheer et al. “Deep sets”. In: *Advances in neural information processing systems.* Vol. 30. 2017, pp. 3391–3401. URL: https://proceedings.neurips.cc/paper/2017/file/f22e4747da1aa27e363d86d40ff442fe-Paper.pdf.