Neural Network Solutions to the Fermionic Schrödinger Equation

Thesis submitted to the University of Surrey
for the degree of Doctor of Philosophy

James Keeble

Supervisor:	Paul Stevenson
Co-Supervisor:	Wilton Catford
Ext-Supervisor:	Arnau Rios

Department of Physics
University of Surrey
Guildford GU2 7XH, United Kingdom

July 18, 2022

E-mail address: `j.keeble@surrey.ac.uk`

Declaration

Scientific abstract

The outline of this thesis is the application of Machine Learning (ML) techniques to solving the Time Independent Schrödinger Equation (TISE). The solving of the TISE is an explicitly non-polynomially (NP) hard computational problem that suffers from the curse of dimensionality. The motivation of this thesis is to see if the computational techniques of ML, which have solved previously intractable problems, are applicable for the NP hard problems of quantum mechanics. If a polynomially scaling algorithm that could solve the TISE accurately existed, it would revolutionise computational modelling of quantum mechanical problems allowing for efficient drug design, simulating super heavy nuclei, and much more.

In the work of this thesis, we begin with a proof-of-principle exercise of solving a variety of Hamiltonians with the use of Deep Neural Networks (DNNs) as the ansatz of the wavefunction. We exploit the use of the Universal Approximation Theorem (UAT), which in principle allows any function to be approximated to an arbitrary precision, to effectively learn the ground-state wavefunction of a Hamiltonian directly via Variational Monte Carlo (VMC) techniques. This work can be seen as an extension to VMC with DNNs which is an emerging field in computation modelling of the Schrödinger equation. The main limitation of VMC is the functional form of the wavefunction by representing the it as a DNN allows the UAT to be exploited to efficiently solving TISE.

In summary, the ML-based approach of this thesis shows strong evidence of efficient wavefunction representation via DNN ansätze with results that compare favourably with exact diagonalisation. This has the implication of DNNs being an efficient approximation to exact numerical methodologies without suffering from the explicit NP hardness of exact diagonalisation.

Keywords: Machine Learning; Deep Neural Networks; Quantum many-body theory.

Acknowledgements

First and foremost I'd like to thank my supervisors, Dr. Arnau Rios, Dr. Paul Stevenson, and Prof. Wilton Catford for their support throughout my entire time at the University of Surrey. I wouldn't be where I am today without their support, guidance, and a hefty dose of patience over my many years at the University of Surrey. I want to thank Dr. Mehdi Drissi for their support in helping me over my PhD, and especially the numerous Zoom meetings which most definitely overran the estimated runtime. I'd like to thank Dr Pierre Arthuis for helping me during my PhD in my understanding of Monte Carlo techniques.

I want to acknowledge and thank the University of Surrey for access to the ORCA computing pool without which the work of this thesis wouldn't be possible. I'd like to thank my office mates, Dr. Chris Mcilroy and Dr Michael Dinmore for great conversation and tolerating my bombardment of questions throughout the PhD. I also want to extend thanks to my housemates; Louie, Michael, and Dominic with whom I experienced lockdown throughout the pandemic. The lockdowns were made manageable by such great company. I would also like to thank Dr. Daniel Gardham for great discussion on Set and Group theory. I also thank and acknowledge Nick Frymann for use of their LaTeX thesis template of which I modified and help in debugging LaTeX's pedantic bibliography syntax which sometimes has a mind of its own. I would like to personally thank Richard Zou and the rest of the FuncTorch team for helping me understand not only FuncTorch but also the inner workings of PyTorch which allow parts of this thesis to be computationally possible.

Finally, I'd like to thank my parents who, throughout my time at University, have always supported me in what I'm doing. I wouldn't the man I am today without their unwavering support, and for that I'm forever grateful.

Contents

Chapter 1

Introduction and Motivation

1.1 Introduction

This thesis introduces the use of Machine Learning (ML) techniques into the field of theoretical nuclear physics as a new method for solving many-body problems. This work is inspired from the recent developments of Refs. [1, 2] which showed that Artificial Neural Networks (ANNs) are efficient variational ansätze for many-body systems. Within these references, ANNs were applied to spin-systems on a lattice and free bosons in continuous space respectively which highlights the dexterity of such techniques. Three different Hamiltonians are studied within this thesis. The first Hamiltonian is the Entem-Machleidt N^3LO interaction which is used to study the deuteron [3, 4]. The second Hamiltonian is a non-interacting Harmonic Oscillator (HO) for a generic A-body system. The final Hamiltonian is a HO with a finite-range inter-particle interaction of strength, V_0, and range, σ_0, for a generic A-body system. The latter two systems involve the use of Fermionic Neural Networks that allow for fermionic many-body wavefunctions to be efficiently represented as a Deep Neural Network (DNN) [5, 6]. The results of all physical systems are compared against direct diagonalisation of the Hamiltonian, and in the case of the generic A-body system it is also compared against the mean field approach of Hartree-Fock (HF) [7, 8, 9].

Within the realm of Quantum Mechanics (QM), objects from single particles to complex molecules, are fundamentally represented by a mathematically quantity called the wavefunction, Ψ, which holds no equivalent in the classical world. In principle an exponential amount of information with respect to the number of particles is required to describe a quantum system, however, a *physical* many-body system can be represented by an amount of information much smaller than the maximum capacity of the Hilbert space of the system [1, 10]. In order to extract quantifiable information from this wavefunction, quantum-mechanical operators need to be applied which will results in eigenvalue-eigenstate pairs. The most notable pair is the Time-Independent Schrödinger equation

(TISE),

$$\hat{H}|\Psi\rangle = E|\Psi\rangle \tag{1.1}$$

where $\hat{H}$ is the Hamiltonian operator, E is the energy eigenvalue of $\hat{H}$, and $|\Psi\rangle$ is the wavefunction eigenstate of $\hat{H}$. This exponential growth is referred to as non-polynomial (**NP**) hard and is the main obstacle in solving quantum mechanical problems. This **NP**-hardness results in numerical approximations to be required in order to solve for such systems. One such polynomially (**P**) scaling algorithm is Variational Monte Carlo (VMC) which is a powerful numerical method in simulating the many-body problem [11]. However, the main problem with VMC is, although it scales polynomially, it suffers from the 'sign-problem' where negative Boltzmann weights result in an exponential growth in statistical error which mitigates VMC's main strength [11].

1.2 Motivation

The motivation behind this thesis is to see if ML techniques can provide a more efficient numerical method in solving quantum many-body problems. Within the ML field, DNNs have shown promise in solving previously intractable problems and have shown superior performance in tasks like the ImageNet competition [12]. In the QM setting, they have already shown promising results across a broad range of physical systems. Ref. [1] was the pioneering paper that kicked off the intersection of ML techniques and applied a Restricted Boltzmann Machine (RBM) to a quantum system of N spins. The application of RBMs in this setting allowed for a superior upper bound on the ground-state energy by over an order of magnitude. The work of Ref. [2] used a Feed-Forward Neural Network (FFNN) to study bosons on a lattice. This model managed to achieve a fidelity of over 99% with the exact ground-state wavefunction. Finally, the work of Ref. [5] directly solved for the many-body wavefunction of a many-electron system and achieved state-of-the-art results, and even managed to beat more sophisticated methods like Diffusion Monte Carlo (DMC). The combination of all these works shows strong indication that ML techniques can solve these QM problems efficiently, and, the wide variety of the physical systems studied, show these techniques are applicable in numerous situations.

In the case of solving for the deuteron, a FFNN ansatz will be used to solve for the ground-state wavefunction and will be compared with direct diagonalisation of the Hamiltonian. In the case of 1D fermionic systems, they will be benchmarked against a whole host of different physical properties; ground-state energies, one/two-body densities, one-body density matrix, interaction strength and range, occupation numbers, and more. This allows for a robust test in the accuracy of such methods across a wide range of physical properties to measure the quality of ML ansätze in numerous quantitative settings. The motivation for the use of DNNs directly ties in with the Universal Approximation Theorem (UAT) which is a core component of ML theory. This theorem is an existence theorem which shows that DNNs can, in principle, approximate arbitrary functions to an arbitrary degree of accuracy [13]. In this thesis it will be applied to QM systems to see if DNNs can outperform traditional forms of ansätze in efficiently representing many-body wavefunctions.

1.3 State-of-the-Art Applications to Quantum Mechanics

Before delving into the work of this thesis, other applications of DNNs to QM systems will be mentioned and discussed in order to highlight the dexterity and compressibility of such numerical methods. Three types of Artificial Neural Networks (ANNs) will be described with their architecture and application of current physical system. These are: Restricted Boltzmann Machines (RBMs), Feed-Forward Neural Networks (FFNNs), and Fermionic Neural Networks (FNNs).

1.3.1 Restricted Boltzmann Machines

The work of Ref. [1] explicitly showed the power of using ANNs to compress many-body problems. The ANN used was an RBM and represents a quantum system of N spins. This method was used to solve for both the ground-state energy and describing the unitary time evolution of the system. An RBM contains two layers of nodes, the first layer (or visible layer) contains the N physical spins of the quantum system. The second layer (or hidden layer) contains M auxiliary spins that represent the underlying ‘feature’ representation of the system being studied. The term ‘feature’ defines an abstract term as a measurable property of the phenomenon being studied. It can effectively be viewed as an embedding within an enlarged physical space [14]. The RBM is viewed as an ‘energy-based’ model within ML literature where a given pair of N physical spins, and M auxiliary spins, have the following associated energy,

$$E\left(v_i, h_j\right) = -\sum_{i=1}^{N} a_i v_i - \sum_{j=1}^{M} b_j h_j - \sum_{i=1,j=1}^{N,M} v_i h_j w_{ij} \tag{1.2}$$

where a_i, b_j denotes the bias of the i^{th} physical and j^{th} auxiliary spins, and w denotes the weights connecting the i^{th} physical spin with the j^{th} auxiliary spin. These weights and biases are complex in order to represent both the amplitude and phase of the wave-function. A given probability of a physical configuration and an auxiliary configuration is defined as,

$$p(v,h) = \frac{1}{Z}\exp\left(-E\left(v,h\right)\right) \tag{1.3}$$

where Z is the partition function, and is given by summing over all combinations of h such that its normalised. The probability of a given physical spin configuration is defined as,

$$p(v) = \frac{1}{Z}\sum_{j=1}^{M} \exp\left(-E\left(v, h_j\right)\right) \tag{1.4}$$

As there are no intra-layer connections, the wavefunction can be explicitly written as,

$$\Psi_M(S; W) = \sum_{j=1}^{M} \exp\left(\sum_{i=1}^{N} a_i \sigma_i^z + \sum_{j=1}^{M} b_j h_j + \sum_{i=1,j=1}^{N,M} \sigma_j^z h_i w_{ij}\right) \tag{1.5}$$

where S is the set of N physical spins, W is the set of total weights and biases, and M denotes the number of hidden units (or auxiliary spins), and σ_i^z is a physical spin. Furthermore, as h_i is in $\{-1, 1\}$, Eq. 1.5 can be simplified to,

$$\Psi_M(S; W) = \exp\left(\sum_{i=1} a_i \sigma_i^z\right) \times \prod_{i=1}^{M} 2\cosh\left(b_i + \sum_j w_{ij}\sigma_j^z\right). \tag{1.6}$$

The ground-state energy, $E_{g.s.}$, is then found via variationally minimising the expectation value of energy with respect to W,

$$E_{g.s.} = \text{argmin}_E \frac{\langle\Psi|\hat{H}|\Psi\rangle}{\langle\Psi|\Psi\rangle}. \tag{1.7}$$

The principles of variational ML states can also be extended to the time-dependent case, where unitary dynamics are described by the Dirac-Frenkel time-dependent variational principle by minimising,

$$R\left(t, \dot{W}(t)\right) = \text{dist}\left(\partial_t \Psi, -i\mathcal{H}\Psi\right). \tag{1.8}$$

The Hamiltonian, $\mathcal{H}$, of Ref. [1] was the Anti-Ferromagnetic Heisenberg (AFH) model,

$$\mathcal{H} = \sum_{ij} \sigma_i^x \sigma_j^x + \sigma_i^y \sigma_j^y + \sigma_i^z \sigma_j^z \tag{1.9}$$

where σ^x, σ^y, and σ^z are the Pauli matrices [1]. Within Ref. [1] this RBM-ansatz was compared against a traditional Jastrow-ansatz as a baseline for a physical system of 80 spins with periodic boundary conditions (PBCs). The RBM-ansatz had a varying number of hidden units denoted via a hidden unit density, α, defined as the ratio of the auxiliary spins, M, to physical spins, N. The Jastrow-ansatz reached an error on the ground-state energy on the order of 10^{-3} with the RBM-ansatz reaching a similar error for $\alpha = 1$. However, given that α is changeable allows the RBM-ansatz to systematically improve its ground-state energy by increasing α. Increasing α to 2 results in a relative error of 10^{-4}, and $\alpha = 4$ reaches a relative error of 10^{-5}. The impact of α on the relative error indicates that an RBM-ansatz is a systemically expandable ansatz for achieving accurate ground-state energy calculations.

1.3.2 Feed-Forward Neural Networks

In Ref. [2] a FFNN is used to solve the Bose-Hubbard model. This model is composed of bosons on a lattice where the state is expanded in terms of Fock states,

$$|\Psi\rangle = \sum \psi(n_1, n_2, \ldots)|n_1, n_2, \ldots\rangle = \sum \psi(\mathbf{n})|\mathbf{n}\rangle \tag{1.10}$$

where n_i is the number of bosons at site, i. The FFNN will be explained in full detail in Sec. 3.2. A succinct description is that it's a multidimensional non-linear function that takes a vector and performs a series of linear maps with a non-linear activation function. This eventually maps the input vector into an output vector which in the case of Ref. [2] is a vector of length 2, denoting the real and imaginary component of the wavefunction. The Bose-Hubbard Hamiltonian is defined as,

$$\mathcal{H} = -J \sum_{\langle ij \rangle} \hat{a}_i \hat{a}_j^\dagger + \sum_i \left[V_i \hat{n}_i + \frac{U}{2} \hat{n}_i (\hat{n}_i - 1) \right] \tag{1.11}$$

where J is the tunnelling coefficient, $\langle ij \rangle$ denotes summing over all pairs of adjacent sites, $\hat{a}$ ($\hat{a}^\dagger$) is the creation (destruction) operators on the lattice, V_i is the site-dependent potential, $\hat{n}_i$ is the number operator (defined as $\hat{a}_i^\dagger \hat{a}_i$), and U is the on-site interaction energy. The main issue of this model is its scalability. For N particles and M sites the number of bases states increases exponentially,

$$|\mathbf{n}\rangle = \frac{(N+M-1)!}{N!\,(M-1)!}. \tag{1.12}$$

This results in 125,970 bases to represent the exact ground-state energy. However, the FFNN-ansatz efficiently encapsulates the information with only 282 weights which reproduces the exact ground-state wavefunction with an overlap of 99%. This indicates that a FFNNN-ansatz can efficiently reproduce the exact ground-state in a compressible manner [15].

1.3.3 Equivariant Neural Networks

In Ref. [5] an antisymmetric DNN is defined to represent a fermionic wavefunction for the electronic Schrödinger equation. The Hamiltonian of a system of N electrons is,

$$\mathcal{H} = -\frac{1}{2} \sum_{i=1} \nabla_i^2 + \sum_{i>j} \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} - \sum_{iI} \frac{Z_I}{|\mathbf{r}_i - \mathbf{R}_I|} + \sum_{I>J} \frac{Z_I Z_J}{|\mathbf{R}_I - \mathbf{R}_J|} \tag{1.13}$$

where $\mathbf{r}_i$ is the position of electron, i, in 3D space, ∇_i^2 is the Laplacian operator with respect to electron, i, and $\mathbf{R}_I$ and Z_I is the position in 3D space and atomic number of nucleus, I [5]. This DNN defines a set of Slater determinants whose 'single'-particle orbitals, $\phi_i^k(x_j; \{x_{\setminus j}\})$, are actually functions of all particle positions. This incorporates an inherent form of backflow [16] to the DNN which defines a *generalised* Slater determinant as,

$$\det\left[\phi_i^k(x_j;\{x_{\backslash j}\})\right] = \begin{bmatrix} \phi_1^k(x_1;\{x_{\backslash 1}\}) & \dots & \phi_1^k(x_n;\{x_{\backslash n}\}) \\ \vdots & \ddots & \vdots \\ \phi_n^k(x_1;\{x_{\backslash 1}\}) & \dots & \phi_n^k(x_n;\{x_{\backslash n}\}) \end{bmatrix} \tag{1.14}$$

where the total wavefunction is defined as a weighted sum of all configurations,

$$\Psi = \sum_{i=1} w_i \det\left[\phi_i^k(x_j;\{x_{\backslash j}\})\right]. \tag{1.15}$$

The orbitals of each configuration are effectively constructed by a DNN which allows for efficient compression of the many-body Schrödinger equation. This form of DNN was compared against other superior numerical methods like Diffusion Monte Carlo (DMC). It was found in Ref. [5] by explicitly representing the ansatz as a DNN, VMC could achieve lower expectation values of the energy. This concept of using DNNs for VMC is referred to as 'Deep' VMC. This indicates that the functional form of the ansatz is the main limitation of traditional VMC methodologies and motivates the design of representing the wavefunction via DNN ansätze. The architecture of the DNN is characterised by the number of hidden nodes, number of layers, and number of determinants (or configurations). All of these are expandable and allow the network to systematically improve the ground-state energy of the network. The entire DNN is shown in Fig. 1 of Ref. [5], and its main results compared against other numerical methods in Tab. 1 of Ref. [5].

1.4 Work of this thesis

The work of this thesis entails two different physical systems (the deuteron and 1D fermions) over 3 Hamiltonians. The first DNN ansatz is a FFNN which is used to solve for the ground-state wavefunctions of the deuteron for a Entem-Machleidt Hamiltonian in momentum space [4]. The second DNN ansatz is a FNN used to solve the TISE for A 1D fermions under two different Hamiltonians.The roadmap of this thesis is set as follows.

The relevant nuclear theory required for this thesis is laid out within chapter 2. It explains the Hamiltonians of this thesis from the Entem-Machleidt interaction of Ref. [4] to the classic Harmonic Oscillator (HO), and HO with finite-range interactions. Both Hartree-Fock (HF), and VMC are both explained with this chapter in order to fully explain how the expectation value of the energy are calculated. HF is an effective mean field approach at solving TISE, and VMC is beyond mean field approach which yields lower expectation values of the energy. Understanding both of these methodologies is vital in how DNN ansätze efficiently embed correlations within the wavefunction.

This lead to chapter 3 in which the required ML theory is described in full. It begins with Statistical Learning Theory (SLT) which motivates how functions can be 'learned' from data. The required components of FFNN are explained here, and the Universal Approximation Theorem (UAT) which allow DNN to approximate functions to an arbitrary accuracy is described in full. The limitations of standard FFNNs are explored and simple modifications are defined which allow required symmetries to be embedded in order to represent many-body wavefunctions which concludes with the DNN ansatz of this thesis. In order to ensure numerical stability when constructing such an ansatz, we define our own custom signed-log determinant function which has its own custom backward (first-derivative) and double-backward (second-derivative) methods. The entire derivation via reverse-mode AD is described in full in appendix A. The remainder of this chapter entails Automatic Differentiation (AD) techniques which allow for tractable gradient calculation for large DNNs, and theoretical aspects of convergence under gradient-descent algorithms are also discussed.

Moving from theory towards results, leads to chapter 4 in which the deuteron ground-state is solved via novel ML techniques. In this chapter, the FFNN ansatz is discussed in full and the methodology of pre-training and energy minimisation are explained. This two section approached methodology will be extended for all results chapters. The pre-training section follows the methodology and FFNN ansatz of Ref. [2] where we first maximise an overlap to an arbitrary target wavefunction for each state of the deuteron. The energy minimisation section takes an expectation value of the energy and uses a gradient-descent algorithm to iteratively reduce the energy until it plateaus to the ground-state energy. The results of this work are discussed and quantified via the binding-energy, wavefunction fidelities, and the D-state probability. The bias and variance of the FFNN ansatz are also discussed in order to quantify the uncertainty of this numerical technique. The wavefunction is optimised via using the root-mean-square propagation (RMSprop) optimiser of Ref. [17].

Within chapter 5, a FNN is used to solve an A-body TISE for the HO Hamiltonian. The methodology is separated into two sections of pre-training and energy minimisation. The energy minimisation stage is the same as Chp. 4 albeit with the use of VMC rather than Gauss Quadrature integration. The pre-training section is modified to fit a set of single-particle orbitals rather than the wavefunction directly by following the pre-training scheme of Ref. [5]. The quality of the results for this chapter are quantified against exact diagonalisation over various properties of the ground-state wavefunction. The one-body density (OBD) and the one-body density matrix (OBDM) are computed via the stochastic approaches of Ref. [18], and are benchmarked against configuration interaction (CI) methods which are exact. The wavefunction is optimised via using the adaptive moment (Adam) optimiser of Ref. [19]. In addition to discussing the Adam optimiser, brief results are shown for a novel extension of the KFAC optimiser. In this section, we compare Adam against KFAC (from Ref. [20]) and provide a novel quasi-Newton extension to KFAC, denoted QN-KFAC, which significantly accelerates convergence. The motivation of this novel extension are discussed in full within appendix B with pseudocode in Sec. B.10

These results are expanded upon towards an interacting Hamiltonian in chapter 6 in which we extend the Hamiltonian of Chapter 5 to include a two-body finite-range Gaussian interaction which has a given interaction strength, V_0, and interaction range, σ_0. The statistical analysis for this chapter is significantly more in-depth than Chp. 5. The interaction strength, and range, are both varied to fundamentally test how the DNN ansätze can represent different physical phenomena. The pairing-gap is also measured for this physical system which indicates how fermions group together to aid in the stability of the physical system. The OBD, OBDM, root-mean square (RMS), and two-body density (TBD) are discussed which define the physical structure of the wavefunction. The individual components of the energy are also discussed in detail and how they related to the different physical phenomena that emerge as the interaction is varied. Analysis of the eigenvalues of the OBDM are also discussed which directly links with the efficiency of the utilising GSDs within the DNN ansätze as opposed to single-particles orbitals. The wavefunction is optimised via using the adaptive moment (Adam) optimiser of Ref. [19].

Chapter 2

Nuclear Theory

2.1 The Deuteron

The deuteron is the simplest nuclear bound state consisting of a neutron and a proton bound together at a binding energy of -2.22 MeV. One of the fermionic problems solved within this thesis will be solving for the ground-state of the deuteron. This proof-of-principle example was chosen to study a simple system with many experimentally verifiable quantities. The ground-state of the deuteron has a permanent quadrupole moment which emerges from the admixture of S-state (L=0) and D-state (L=2) by the strong interaction. As there exists two states within the deuteron's ground-state wavefunction, the FFNN must encapsulate both states simultaneously which is a fundamental test of compressibility for this methodology. By focusing on the deuteron allows the analysis to be purely focus on the use of ML techniques, and direct benchmarks with exact methods which becomes impossible for higher many-body problems. The deuteron is solved within momentum space in the centre-of-mass reference frame which effectively solves the problem in terms of relative momentum. The binding energy, E_B, of the deuteron is experimentally determined by measuring γ-rays from radiative np-capture with thermal neutrons. This was measured using a crystal diffraction spectrometer to relative accuracy of 2 x 10^{-7} [21]. The asymptotic D/S state, η, is determined via measurements of tensor analysing powers in sub-Coulomb (d,p) reactions on heavy nuclei and comparing those to calculation of distorted wave Born approximation (DWBA) [22]. The radius of the deuteron, r_d, is determined experimentally via elastic electron scattering experiments - this has been a standard technique used since the 1950s [23]. The rms charge radius, r_{ch}, is related to the nucleonic rms radius, r_m, by,

$$r_{ch}^2 = r_m^2 + \Delta r_m^2 + r_p^2 + r_n^2 + \frac{3}{4}\left(\frac{\hbar}{m_p}\right)^2. \tag{2.1}$$

	N^{3}LO	Expt.	Units
E$_B$	2.224575	2.224575(9)	(MeV)
A$_s$	0.8843	0.8846(9)	(fm$^{-1/2}$)
η	0.0256	0.0256(4)	(1)
r$_d$	1.978	1.97535(85)	(fm)
Q	0.285	0.2859(3)	(fm^2)
P_D	4.51	N/A	(%)

Table 2.1: The properties of the deuteron for a N^{3}LO interaction compared with experimental values [4]

Where Δr_m^2, is the contribution from non-nucleonic degrees of freedom of size $\approx \pm 0.01$fm^2, r_p is the proton charge rms radius, and r$_n$ is the neutron charge rms radius of values 0.862(12) fm and -0.113(5) fm respectively [24, 25]. The neutron ms radius is negative due to the negative-π-meson cloud [25]. The final term emerges from relativistic origins [26] which leads to the radius of the deuteron being calculated by,

$$r_d^2 = r_m^2 + \Delta r_m^2. \tag{2.2}$$

The quadruple moment, Q, of the deuteron was discovered in 1939, and lead to the realisation that the nuclear force was not central but more complex providing evidence for pions being involved in nuclear physics [27, 28]. The D state probability of the deuteron, P_D, is correlated with the non-zero quadruple moment, i.e. if $P_D = 0$ then $Q = 0$ [29]. The tensor component of the nucleon-nucleon interactions plays a key role in the binding energy of few- and many-body systems [3].

2.2 Towards an Effective Field Theory

In 1932 James Chadwick discovered a subatomic particle that held no charge, the neutron. This subatomic particle revolutionised nuclear theory by allowing for the discovery of the strong nuclear force. This resulted in the fact that nucleus contain two types of particles, protons and neutrons, which can't be held together via electromagnetic nor gravitational forces. This lead to the revelation that a new force was at play [30].

The first attempt at deriving nucleon-nucleon (NN) interactions were derived by Yukawa in 1935 where one pion exchanges were strong force mediators [31]. This initially worked well for NN scattering data and the properties of the deuteron. However, when multi pion exchange occurred serious ambiguities arose which didn't agree well with the empirical information about the nuclear force [4, 32].

This model was saved by the discovery of heavy mesons in early 1960s, leading to the creation of a one boson exchange (OBE) model which is the most economical and quantitative phenomenology for describing the NN interaction to this day. A shortcoming of the OBE model emerged from controversial evidence of the sigma and epsilon scalar-isospin bosons [4]. These bosons are associated with the correlated (or resonant) exchange of two pions. This lead to a decade long expenditure to derive a 2π-exchange contribution to the nuclear force which creates the intermediate range attraction. During this period both dispersion theory and field theory were invoked to produce the Paris and Bonn potentials respectively [33, 34, 35].

With the discovery of Quantum Chromodynamics (QCD) the progress made with these “meson” theories were moderate and attempts were made to derive the nuclear force from QCD [4]. The issue with deriving a nuclear force from QCD is that the theory is non-perturbative in the low-energy regime of nuclear physics, therefore, making direct solutions impossible [3]. Initial attempts at resolving this issue were QCD inspired quark models which attempted to resolve the non-perturbative problem [36]. Although being successful at reproducing some features of the nuclear force, they marked no real significant progress [4]. The solution to this issue was to build an effective field theory which recognises different energy scales within nature. The most notable scale is the chiral breaking symmetry at $\Lambda_\chi \approx 1$ GeV below which the degrees of freedom are pions and nucleons interacting via a force governed by the symmetries of QCD [3].

Weinberg applied an effective field theory (EFT) to low-energy QCD in 1979 to yield a general Lagrangian consistent with symmetry principles and most importantly broken chiral symmetry of QCD [37]. At this low energy, pions and nucleons are the degrees of freedom as opposed to quarks and gluons, and heavier mesons are ‘integrated out’. This effectively brings nuclear theory full circle back to the pion theory of the 1950s, except with the knowledge of broken chiral symmetry which emerges from QCD.

The effective chiral Lagrangian consists of an infinite series with an increasing number of derivatives and/or nucleon fields where each term has a dependence on the pion field defined by the rules of broken chiral symmetry [38]. The general idea is to apply this Lagrangian to scattering data, i.e. NN and πN scattering, to encapsulate to underlying properties of the nuclear force as a pion beam interacting with a nucleus is equivalent to the pion-exchanges occurring within nuclei. However, when the Lagrangian is fitted to NN scattering data an unlimited number of Feynman diagrams are generated which suggests an intractable problem [4]. In 1990 Weinberg showed that chiral EFT can be encapsulated in a NN potential. This can be systematically expanded on the order of $(Q/\Lambda_\chi)^\nu$ where Q is the momentum, Λ_χ is the chiral symmetry breaking scale, and ν is a given order of the systematic expansion. This expansion at a given order allows for a given accuracy but most importantly it solves the aforementioned intractability. This systematic expansion is known as Weinberg’s power counting scheme [38], and the overall scheme is referred to as chiral perturbation theory (χPT) [4]. The power of ν of a diagram is determined by,

$$\nu = 2l + \sum_j \left(d_j - 1\right) \tag{2.3}$$

Where, l, represents the number of loops in the diagram, d_j, is the number of derivatives in vertex, j, and the sums acts on all vertices of the diagram. The important one pion exchange (OPE) and the two pion exchange (TPE) are shown within Fig. 1 of Ref. [3].

2.3 The N^{3}LO Potential

The Entem-Machleidt N^{3}LO interaction is a fourth-order chiral perturbation theory (χPT) interaction which is based below the chiral symmetry breaking scale of 1 GeV [4]. The starting point for χPT is to begin with an effective chiral πN Lagrangian based on a series of derivatives,

$$\mathcal{L}_{\pi N} = \mathcal{L}^{(1)}_{\pi N} + \mathcal{L}^{(2)}_{\pi N} + \mathcal{L}^{(3)}_{\pi N} \tag{2.4}$$

Where $\mathcal{L}^{(j)}_{\pi N}$ is a πN-Lagrangian where the superscript refers to the number of derivatives, j. If a heavy baryon (HB) formulation of χPT is used the relativistic Lagrangian is subjected to an non-relativistic expansion in terms of powers of $1/M_N$ [39]. At leading order,

$$\hat{\mathcal{L}}_{\pi N} \approx \bar{N}\left[i\partial_0 - \frac{1}{4f_\pi^2}\cdot(\times\partial_0) - \frac{g_A}{2f_\pi}\cdot\left(\vec{\sigma}\cdot\vec{\nabla}\right)\right]N \tag{2.5}$$

where M_N = 938.919 MeV is the nucleon mass, m$_\pi$ is the pion mass equal to 138.04 MeV, f_π is the pion-decay constant equal to 92.4 MeV, and g_A is the axial-vector coupling constant equal to g$_\pi NNf_\pi/$M$_N$ = 1.29 [3]. At second order, the Lagrangian can be composed of a fix and contact-term Lagrangian,

$$\hat{\mathcal{L}}^{(2)}_{\pi N} = \hat{\mathcal{L}}^{(2)}_{\pi N,\text{fix}} + \hat{\mathcal{L}}^{(2)}_{\pi N,\text{ct}}. \tag{2.6}$$

The term, $\hat{\mathcal{L}}_{\pi N,\text{fix}}$, is 'fixed' due to there being no free parameters which emerges from the HB expansion [3]. The parameters of $\hat{\mathcal{L}}_{\pi N,\text{ct}}$, are the low-energy constants (LECs) which are determined via fitting to πN scattering data. The value of these LECs are c_1 = -0.81 GeV^{-1}, c_2 = 2.80 GeV^{-1}, c_3 = -3.20 GeV^{-1}, and c_4 = 5.40 GeV^{-1} [4]. The LECs $c_{1,3,4}$ are fitted to πN scattering data of Ref. [40], and c_2 is fitted to πN scattering data of Ref. [41]. These c_i constants belong to $\hat{\mathcal{L}}_{\pi N,\text{ct}}$ [4]. In order to make accurate predictions the NN interaction must be of the highest precision. The NN interaction must reproduce NN data below 300 MeV in the lab frame with χ^2/datum $\approx$ 1.

An important feature of χPT is that many-body forces are specifically predicted. At a given order of χPT, both 2NF and 3NF are generated. At NLO, all 3NF cancel [42], and at NNLO and above, non-zero 3NF terms emerge [43]. The contributions of 3NF at NNLO are small but ever-increasing with higher orders of χPT [4]. At fourth-order χPT, N^{3}LO the accuracy is near-equivalent to high precision phenomenological interactions. Therefore, this allows for exact few-body nuclear structure calculations to be done, e.g. the deuteron.

The contributions in χPT are determined by contact terms and pion-exchange diagrams. For the N^{3}LO interaction, there are 24 contact terms which yield 24 parameters. The parameters emerge from different orders of momentum Q; $\mathcal{O}\left(Q^0\right)$ has 2 parameters, $\mathcal{O}\left(Q^2\right)$ has 7 parameters, $\mathcal{O}\left(Q^4\right)$ has 15 parameters which totals 24 contact parameters for the N^{3}LO interaction which are crucial in fitting partial waves of L≤2 [3]. The pion contributions begin with a static one-pion exchange $\mathcal{O}\left(Q^0\right)$. Two-pion exchange begin at $\mathcal{O}\left(Q^2\right)$ and exist at higher orders of Q [44, 45]. Also, at N^{3}LO three-pion exchanges (3PE) first emerge, however, these terms are negligible and hence not included [46].

As χPT is a low-momentum expansion, below the chiral symmetry breaking scale of Λ_χ = 1 GeV, a regulator function needs to be introduced to enforce that all terms (contact and pion exchanges) are only working in the low-momentum region below Λ_χ. This regulator function is defined as,

$$\exp\left[-\left(\frac{p}{\Lambda}\right)^{2n}-\left(\frac{p'}{\Lambda}\right)^{2n}\right] \tag{2.7}$$

where p and p' are the initial and final nucleon momenta in the centre of mass frame. A value of $\Lambda = 0.5$ GeV is used within Ref. [4]. The 2n factor in the exponent exists to ensure that the regulator function is always above the order at which calculations are made [4]. The NN interaction at $\nu = 4$ is defined as the sum of the contact terms and pion-exchange diagrams multiplied by the regulator function of Eq. 2.7 [4]. For peripheral partial waves of $L \geq 4$ there is good agreement between χPT and conventional 2π models/empirical phase shifts [35]. The agreement holds fairly well for L = 3, but for $L \leq 2$ the agreement deteriorates. To remedy this deterioration, repulsive short-range contributions need to be included [3]. In conventional meson model these short-range correlation are created by heavy mesons, i.e. ω-meson. However, within χPT, heavy mesons don't exist, therefore, these short-range contributions are parametrised by contact potentials for the D state, an order of $\nu = 4$ is required [3]. The contact terms contribute solely to L ≤ 2 states, and L ≥ 3 states are determined exclusively from one- and two-pion exchanges [4]. These pion exchanges are determined via the axial-vector coupling constant, g_A, the pion-decay constant, f_π, and 8 low-energy constants (LECs). To ensure that the N^{3}LO potential accurately describes NN (pp and np) data, two charge dependent contacts interactions at Q^0 are included.

The resultant N^{3}LO potential has an accuracy on par with phenomenological potentials like Argonne V$_{18}$ [47]. The systematic expansion of Weinberg's power counting theoretical uncertainty can be estimated by calculating the next order at $\left(Q/\Lambda_\chi\right)^{\nu+1}$. This gives a theoretical uncertainty for N^{3}LO, $\nu = 4$, at 3% for fitting np-scattering data [4].

2.4 The Harmonic Oscillator

Nucleon-Nucleon (NN) interactions govern all of nuclear physics and describe how nucleons within the nucleus behave. The general properties of the nucleus can be described as independent particles moving within a mean field of single particle orbitals [9]. The interaction has a hard core at short distances but given nucleons are far apart they only interact via the 'tail' of the nuclear force [48] and hence such repulsion is seldom experienced.

In the construction of a NN interaction the following behaviour must be defined; nucleons must have no net force towards the centre of the nucleus, the binding force must increase as a nucleon moves towards the surface, and the binding force must decay once leaving the surface due to the finite range of the nuclear force [9]. This is defined mathematically for an arbitrary interaction below by the following conditions,

$$\left.\frac{\partial V(r)}{\partial r}\right|_{r=0} = 0 \tag{2.8}$$

$$\left.\frac{\partial V(r)}{\partial r}\right|_{r<R_0} > 0 \tag{2.9}$$

$$V(r) \approx 0 \,\text{if}\, r > R_0. \tag{2.10}$$

An analytical form of this potential, which satisfy the 3 aforementioned criteria, is the Woods-Saxon potential,

$$V^{W.S.}(r) = -V_0\left(1+\exp\left(\frac{r-R_0}{a}\right)\right)^{-1}, \tag{2.11}$$

where a is the skin thickness of 0.5 fm, R_0 is the size of the nucleus defined as $R_0 = r_0 A^{1/3}$ with r_0 being a proportionality constant of 1.2 fm, and the value V_0 is the interaction strength [49]. When solving such an interaction we must solve the TISE to determine the energy (eigenvalue) and single-particle orbitals (eigenfunctions). However, such an interaction does not have analytical solutions, hence, we must further approximate this interaction via the use of the Harmonic Oscillator (HO) interaction which works well near the centre of the nucleus,

$$V(r) \approx -V_0\left(1-\left(\frac{r}{R_0}\right)^2\right) = \frac{m}{2}\omega^2\left(r^2 - R_0^2\right). \tag{2.12}$$

In this thesis the HO interaction is studied within one-dimensional (1D) space. In 1D space, the HO is non-degenerate meaning there's a one-to-one correspondence between eigenvalues and eigenfunctions. This leads to the energy spectrum being defined as,

$$E_n = \hbar\omega\left(n+\frac{1}{2}\right), \tag{2.13}$$

where n is the principal quantum number, and the degeneracy of each state equal to unity. When A fermions without spin are confined in a 1D HO interaction, the ground-state energy of the total system is defined as the sum of the lowest lying A shells which equals,

$$E = \sum_{n=0}^{A-1} E_n = \frac{A^2}{2}\hbar\omega. \tag{2.14}$$

In this setting the total energy for A particles in the 1D HO interactions is quadratic in the number of particles in units of $\hbar\omega$. In the three-dimensional (3D) case the oscillator frequency, ω, is determined from the mean square radius of a sphere [9]. In the nuclear setting this results in $\hbar\omega$ being defined as $40A^{-1/3}$ MeV with an associated oscillator length of $1.0165A^{1/6}$ fm [9]. Therefore, we will use ho units throughout this thesis with energies measured in $\hbar\omega$ and length in units of oscillator length. The eigenvalues of each state are defined by Eq. 2.13 with the corresponding eigenfunctions being the Hermite polynomials of order n multiplied by a Gaussian in the 1D case [50]. Within the harmonic trap, $E > V_0$, the single-particle orbitals oscillate and outside the trap, $E < V_0$, the single-particle orbitals exponentially decay. This leads to the unique quantum mechanical behaviour of particles existing within classically forbidden regions where particles can penetrate an energy barrier they don't have enough energy to pass over. Finally, the Hamiltonian, $\hat{H}$, of this interaction is defined by its kinetic energy contribution and the potential energy contribution of the HO interaction,

$$\hat{H} = -\frac{\hbar^2}{2m}\sum_{i=1}^{A}\nabla_i^2 + \frac{1}{2}m\omega^2\sum_{i=1}^{A}x_i^2. \tag{2.15}$$

Where $\hbar$ is the reduced Planck's constant, m is the mass of a nucleon, ω is the oscillator frequency of the well.

2.5 Extending the Harmonic Oscillator with a 2-body Gaussian Interaction

One way to extend the HO interaction is to include a two-body inter-particle interaction. In the ultracold gas literature for 1D systems [51, 52], this is primarily a contact interaction. This is not applicable with spinless interactions as the Pauli Exclusion Principle (PEP) prohibits fermions from occupying the same position which leads to zero contribution to the energy from the interaction. Instead one can construct a finite-range interaction of the following functional form,

$$\hat{H} = -\frac{\hbar^2}{2m}\sum_{i=1}^{A}\nabla_i^2 + \frac{1}{2}m\omega^2\sum_{i=1}^{A}x_i^2 + \sum_{i<j}^{A}\frac{V}{\sqrt{2\pi}\sigma}\exp\left(-\frac{\left(x_i - x_j\right)^2}{2\sigma^2}\right). \tag{2.16}$$

In the cold-atom community the units of choice is HO units wherein units are measured with respect to $\hbar\omega$, and lengths are defined in terms of $a_{ho} = \sqrt{\frac{\hbar}{m\omega}} \approx 1.0165 \times A^{1/6}\text{fm}$ with A being the number of particles of mass, m. Therefore, the Hamiltonian of Eq. 2.16 can be rewritten in HO units as,

$$\hat{H} = -\frac{1}{2}\sum_{i=1}^{A}\nabla_i^2 + \frac{1}{2}\sum_{i=1}^{A}x_i^2 + \sum_{i<j}^{A}\frac{V_0}{\sqrt{2\pi}\sigma_0}\exp\left(-\frac{\left(x_i - x_j\right)^2}{2\sigma_0^2}\right). \tag{2.17}$$

The additional two-body interaction term is of a given interaction strength, V_0, and finite-range σ_0 which relate to the V and σ respectively via,

$$V = a_{ho}\hbar\omega V_0 = \sqrt{\frac{\hbar^3\omega}{m}} \approx 40.66A^{-1/6}V_0, \tag{2.18}$$

$$\sigma = a_{ho}\sigma_0 \approx 1.0165A^{1/6}\sigma_0. \tag{2.19}$$

By adding this interaction to the Hamiltonian it becomes a contact interaction in the limit of $\sigma_0 \mapsto 0$. Although this Gaussian interaction is relatively simple, it can lead to drastically different behaviour in either limit of V_0. As $V_0 \mapsto -\infty(\infty)$ the phenomena of bosonisation (Wigner crystallisation) will occur which will be a main focal point of the data analysis within chapter 6. The DNN ansatz shows remarkable ability to accurately reproduce both behaviours in their respectively limits of V_0 highlighting its dexterity in representing quantum mechanical phenomena.

2.6 Hartree-Fock Theory

As mentioned in Sec. 2.4 the independent particle model states that nucleons interact via an 'average' or 'mean-field' potential that emerges from all nucleons interacting with each other. The idea of Hartree-Fock is to take an inherently two-body interaction and defined an approximate one-body interaction that adequately describes the interaction via a mean-field. This results in the exact Hamiltonian,

$$\hat{H} = \sum_{i=1}^{A}\left(-\frac{\hbar^2}{2m}\nabla_i^2 + \hat{U}(x_i)\right) + \sum_{i<j}^{A}\hat{V}(x_i, x_j) \tag{2.20}$$

where $\hat{U}$ and $\hat{V}$ are one- and two-body interaction respectively, being represented as an effective one-body interaction,

$$\hat{H}_{HF} = \sum_{i=1}^{A}\hat{h}_{HF,i} = \sum_{i=1}^{A}\left(\hat{t}_i + \hat{V}_{HF,i}\right) \tag{2.21}$$

where $\hat{h}_{HF,i}$ is the Hartree-Fock interaction for nucleon, i, which is initially unknown. Solving the Schrödinger equation for this interaction, $\hat{H}_{HF}\Psi = E_{HF}\Psi$, requires a functional form for the wavefunction, Ψ. A valid starting wavefunction ansatz is a Slater determinant of single-particle orbitals which reduces the eigenvalue problem to that of the one-body effective interaction [9],

$$\hat{h}_{HF}\phi_j = \varepsilon_j\phi_j. \tag{2.22}$$

In fact, the Hartree-Fock method is explicitly restricted to the use of a set of single-particle orbitals within a Slater determinant [8]. This restricts the one-body density matrix to have the properties of idempotency (Eq. 2.23) with a trace-norm equal to the number of particles (Eq. 2.24), i.e.,

$$\rho^2 = \rho \tag{2.23}$$

$$\mathrm{Tr}\,(\rho) = A. \tag{2.24}$$

The set of orbitals are varied to reduce the variational energy via an iterative algorithmic procedure [9]. In order to generate the effective one-body interaction the Hartree-Fock equation (Eq. 2.25) must be solved,

$$\left(-\frac{\hbar^2}{2m} + \Gamma_H(x)\right)\phi_k(x) + \int dx'\Gamma_{Ex}\left(x,x'\right)\phi_k(x') = \varepsilon_k\phi_k(x). \tag{2.25}$$

This equation contains two terms; the Hartree (direct), and the Fock (exchange) terms which represent a local and non-local interaction respectively [8]. The Hartree term, Γ_H, is defined as,

$$\Gamma_H(x) = \int dx'V(x,x')\sum_{j=1}^{A}\left|\phi_j(x)\right|^2 = \int dx'V(x,x')\rho(x') \tag{2.26}$$

where $V(x,x')$ is our exact two-body interaction we seek to reduce to an effective one-body interaction, and $\rho(x')$ is the one-body density of our Slater determinant, Ψ. The Fock term, Γ_{Ex}, is defined as,

$$\Gamma_{Ex}(x,x') = -V(x,x')\sum_{j=1}^{A}\phi_j^*(x')\phi_j(x) = -V(x,x')\rho(x,x') \tag{2.27}$$

with $V(x,x')$ being the exact two-body interaction we seek to reduce to an effective one-body interaction, and $\rho(x,x')$ is the one-body density matrix of our Slater determinant, Ψ, which is defined as the integral of Ψ over all by two positions. By solving Eq. 2.25 we can obtain an upper bound on the expectation value of the energy. However, the Hartree-Fock equation isn't a standard eigenvalue problem as the corresponding eigenvalues are dependent upon an effective one-body interaction which is derived via a set of single-particle orbitals. This changes the eigenvalue problem towards a self-consistent approach which can be viewed as an iterative algorithmic procedure [8]. Firstly, a set of single-particle orbitals, $\phi_i(x)\, i \in 1,\ldots,N$, are defined which are used to construct the one-body density matrix. These are substituted into Eqs. 2.26 and 2.27 to compute the

Hartree-Fock one-body interaction. Subsequently, the eigenvalue problem of Eq. 2.25 is solved for the current Hartree-Fock interaction. This leads to a set of eigenvalues and *new* eigenvectors. The eigenvectors with the N lowest eigenvalues are selected and defined the set of new orbitals. The total energy is sum over all eigenvalues [9]. This process is iteratively repeated until the difference between iterations meets a threshold value which is usually of order of magnitude of 10^{-8} or a maximum number of iteration is exceeded [7]. The ground-state energy of the Hartree-Fock method is a variational upper bound on the true ground-state energy. One limitation of the Hartree-Fock method is that it lacks correlation energies between particles [8].

2.7 Variational Monte Carlo

Variational Monte Carlo (VMC) is a statistical algorithm for accurately solving the many-body Schrödinger equation which comprises of two core components; the Variational principle, and Monte Carlo Sampling. In this chapter, both of these techniques will be discussed in depth to highlight how their combination allows for efficient calculation of many-body operators; most notably the expectation of the energy. In addition, traditional ansätze will be discussed for a variety of many-body problems and how particular components of the wavefunction can be essential in accurately describing specific physical systems.

2.7.1 Variational Principle

When solving for the ground-state of a many-body wavefunction, $|\Psi\rangle$, under a given Hamiltonian, $\hat{H}$, the variational principle can be exploited to define an upper bound on the ground-state energy [53],

$$\langle E\rangle = \frac{\langle\Psi|\hat{H}|\Psi\rangle}{\langle\Psi|\Psi\rangle} \geq E_{gs}, \tag{2.28}$$

where the expectation value of the energy for a given wavefunction can only equal the ground-state iff the wavefunction is the ground-state wavefunction [53], i.e.,

$$\frac{\langle\Psi_{gs}|\hat{H}|\Psi_{gs}\rangle}{\langle\Psi_{gs}|\Psi_{gs}\rangle} = E_{gs}. \tag{2.29}$$

As the Hamiltonian used within this work is local in position space we can decompose the integral of Eq. 2.28 into an expectation over the Born distribution of the wavefunction [54],

$$\langle E\rangle = \frac{\int_{-\infty}^{\infty} |\Psi(X)|^2 \frac{\hat{H}\Psi(X)}{\Psi(X)}\,\mathrm{dX}}{\int_{-\infty}^{\infty} |\Psi(X)|^2\,\mathrm{dX}}. \tag{2.30}$$

where X is a vector encoding of the positions of each particle, $X = \{x_1, \ldots, x_A\}$. By moving the denominator of Eq. 2.30 into the $|\Psi|^2$ term defines the normalised Born probability of the wavefunction. This allows for the many-body integral to be represented as a statistical expectation which is significantly more efficient to compute than the direct integral. This statistical approximation is directly equivalent to the deterministic integral in the infinite limit of samples. The statistical expectation is defined as,

$$\langle E \rangle = \int P(X) \frac{\hat{H}\Psi(X)}{\Psi(X)} dX \approx \mathbb{E}_{x \sim |\Psi(X)|^2} \left[\frac{\hat{H}\Psi(X)}{\Psi(X)} \right]. \tag{2.31}$$

The term within the statistical expectation is referred to as the *local* energy. This ability to decompose the many-body integral to a statistical expectation only applies to local hamiltonians. For example, the expectation of non-local interaction of Chp. 4 cannot be computed via Eq. 2.31 and hence must be reduced to standard integral techniques.

One way of representing a wavefunction is to define it within a many-body basis, $\{|x\rangle\}$, which spans the entire Hilbert space via the completeness relation,

$$\sum_x |x\rangle\langle x| = \mathbb{I}. \tag{2.32}$$

However, one limitation of this method is that the number of basis functions scales exponentially with the number of particles which for large systems requires truncating the basis that introduces a notable source of error [5]. Another way of representing a wavefunction is to construct a parameterisable function which holds a set of variational parameters, W, a so-called ansatz, $\Psi_W(X)$. The parameters of the ansatz can be updated by gradient descent where the expectation value of the energy is computed. The gradient of the energy with respect to these parameters is calculated and then used to iteratively update the weights such that the energy decreases until its reaches a local minima. This is efficiently computed via Eq. 2.31 and the variational principle (Eq. 2.28) to provide an upper bound on the ground-state [1, 5, 6]. In principle, this allows for a polynomially scaling algorithm to solve the Schrödinger equation because the inherent exponential scaling of integration is converted to an efficient sampling and local energy calculation instead.

In addition to being a polynomially scaling algorithm, VMC techniques hold two key properties that only apply within the quantum mechanical setting; the zero-bias property; and zero-variance property [55]. As the trial wavefunction approaches the ground-state wavefunction, its energy will approach the ground-state energy which losses any dependence on X,

$$\frac{\hat{H}\Psi_{gs}(X)}{\Psi_{gs}(X)} = \frac{E\Psi_{gs}(X)}{\Psi_{gs}(X)} = E \tag{2.33}$$

due to it being an eigenfunction. As it is the ground-state wavefunction, it will equal the lowest eigenvalues of $\hat{H}$ in accordance to the variational principle of Eq. 2.28. Secondly, the energy of the ground-state wavefunction also has zero variance,

$$\sigma_E^2 = \frac{\int \Psi_{gs}(X)\hat{H}^2\Psi_{gs}(X)}{\int |\Psi_{gs}(X)|^2} - \left(\frac{\int \Psi_{gs}\hat{H}\Psi_{gs}}{\int |\Psi_{gs}|^2}\right)^2 = \mathbb{E}_{x\sim|\Psi|^2}\left[E^2\right] - \mathbb{E}_{x\sim|\Psi|^2}\left[E\right]^2 = 0. \quad (2.34)$$

In practice, our ansatz will have a non-zero bias and non-zero variance as it will only be an approximation of the ground-state wavefunction. However by utilising the Universal Approximation Theorem (UAT) (3.3), which is a core component in the effectiveness in ML, we can systematically improve the representation of our wavefunction to approximate the ground-state eigenfunction. In fact, a main limitation of current VMC methods is the functional form of the wavefunction ansatz. By explicitly representing the wavefunction as a DNN we can exploit the UAT to directly improve the trial wavefunction thereby overcoming a key limitation in standard VMC algorithms [5].

The inherent stochastic nature to the integration of Eq. 2.31 is the main focus of scalability of VMC algorithms. If we compare stochastic estimation against a deterministic method, e.g. Simpsons' rule, we can see that Simpsons' rule scales exponentially in the number of function evaluations with respect to the dimensionality of the integral. The number of function evaluations in the stochastic case is actually *independent* of the dimensionality in accordance to the Central Limit Theorem (CLT) and is instead proportional to the number of samples within the statistical expectation [56]. This defines the integral as,

$$\langle E\rangle \approx \mathbb{E}_{x\sim|\Psi|^2}\left[E_L(X)\right]. \quad (2.35)$$

The many-body positions, X, are random variates sampled in proportion to the Born probability of the wavefunction [54]. Therefore, the local energy of a many-body position, $E_L(X)$, are also random variates distributed in accordance to their own distribution, $\mathcal{P}(E_L)$. A key property of the CLT is that, under the Law of Large Numbers (LLN) this probability distribution will tend to a Gaussian distribution whose mean is defined by Eq. 2.31 and variance by Eq. 2.34 [56]. This remarkable results shows that the integral is purely dependent on the number of points and its variance is proportional to $\frac{1}{\sqrt{N}}$ where N is the number of samples which defines a polynomially scaling method to compute integrals.

2.8 Monte Carlo

Monte Carlo techniques date back to the early 1930s to when Enrico Fermi studied neutron diffusion [57]. This work was unpublished, however, he is credited with the creation of this technique [58]. The technique allows for highly parallelisable, efficient, and most importantly scalable computation of multi-dimensional integrals under samples drawn from a given probability distribution. Now the question arises, how can one sample configurations arbitrarily from a probability density function?

This leads into two branches of sampling domains; direct sampling, and indirect sampling. The former corresponds with sampling from normalised probability density functions

(PDFs) whereas the latter corresponds within unnormalised PDFs via the use of Markov Chains using the Metropolis-Hastings algorithm [58, 59].

2.8.1 Direct Sampling

The most basic form of direct sampling is uniform sampling, or brute force sampling, in which we define a domain and sample configurations with equal probability. A classic example of most textbook is calculating π via 'counting' configurations which lie within a unit diameter circle that is placed inside a unit length square (see Fig. 3.1 of Ref. [57]). The methodology of this pedagogical example doesn't direct 'translate' to general integrals, therefore, we'll use a proxy example with the same integral value,

$$\mathcal{I} = \int_{-5}^{5} dx \int_{-5}^{5} dy \exp\left(-(x^2+y^2)\right) \approx \pi \tag{2.36}$$

where the approximation of π emerges due to truncation of the integral domain. The domain is truncated as the probability of a uniform distribution over $\mathbb{R}^2$ is zero. This integral can be reformatted into an expectation value as previously stated,

$$\mathcal{I} = \int_{-5}^{5} dx \int_{-5}^{5} dy \frac{\exp\left(-(x^2+y^2)\right)}{\mathcal{U}(x,y)} \mathcal{U}(x,y) \tag{2.37}$$

where $\mathcal{U}(x,y)$ is a uniform distribution in range -5 to 5 in all dimensions of the configuration space. In order for the integral to be refactorised as an expectation value, the PDF from which samples are drawn must span the support of the integral, hence the uniform PDF spans -5 to 5 in all dimensions. As the probability of a Uniform PDF is just the volume of the simulation cell, it can be factored outside the expectation and herein lies the problem,

$$\mathcal{I} =\approx \mathbb{E}_{x,y\sim\mathcal{U}(x,y)}\left[\frac{\exp\left(-(x^2+y^2)\right)}{\mathcal{U}(x,y)}\right] \approx 25\mathbb{E}_{x,y\sim\mathcal{U}(x,y)}\left[\exp\left(-(x^2+y^2)\right)\right]. \tag{2.38}$$

This expectation treats *all* configurations equally so configurations near the centre, which have a large function magnitude, contribute as equal as function values near the edge of the integral domain which have small function magnitude. Ideally, we'd want more samples distributed towards the centre of the domain and few towards the edge, i.e. distribute in proportion to the integrand itself. This can be done by changing the sampling PDF to more closely follow the shape of the integrand. In the perfect case, this would be to follow the integrand proportionally thereby perfectly distributing the samples in proportional to the integrand itself. This would lead to zero-variance in the estimator for the integral,

$$\sigma_{\mathcal{I}}^2 = \mathbb{E}_{x,y\sim P(x,y)}\left[\left(\frac{\exp\bigl(-(x^2+y^2)\bigr)}{c\exp(-(x^2+y^2))}\right)^2\right] - \mathbb{E}_{x,y\sim P(x,y)}\left[\frac{\exp\bigl(-(x^2+y^2)\bigr)}{c\exp(-(x^2+y^2))}\right]^2 \tag{2.39}$$

$$\sigma_{\mathcal{I}}^2 = \mathbb{E}_{x,y\sim P(x,y)}\left[\frac{1}{c^2}\right] - \mathbb{E}_{x,y\sim P(x,y)}\left[\frac{1}{c}\right]^2 = 0 \tag{2.40}$$

where c is the proportionality constant. As this method uses direct sampling, the PDF must be strictly positive and normalised to unity. However, when the PDF is normalised to unity the integral is effectively done, thereby invalidating the need to do the integral with which to begin. This main caveat results in it not being possible to choose a PDF which is directly proportional to the integral we wish to solve.

With that being said, this doesn't rule out this method 'directly', it just invalidates a specific subset of PDFs. In practice, we can use a normalised PDF that more closely follows the shape of the integrand than just a uniform distribution where samples are distributed more densely towards peaks and are more sparse towards troughs. This 'smart' method to distribute samples is refereed to as *importance sampling* and can be many orders of magnitude more efficient than brute-force sampling [60].

2.8.2 Indirect Sampling

The aforementioned methods fall under the category of 'direct' sampling methods wherein samples of a given PDF are sampled independently and identically distributed (i.i.d). However, this approach is severely limited in practice to small dimensional problems, functions without sharp peaks, and even can exceed the allocated memory available. Some methods do exist that exploit *importance sampling* alongside *stratified sampling* like VEGAS to deal with sharp integrals [61, 62]. Therefore, this motivates moving pass 'direct' methods to 'indirect' methods where instead of storing the entire state space of a given PDF, we reformulate it as a stochastic process within a phase space. This allows samples to be drawn from arbitrary PDFs without storing the entire state space. In addition, variants of these methods can allow samples to be 'encouraged' to be sampled from peaks of PDFs which can be found automatically thereby leading to increase efficiency [60].

The bulk of these techniques fall under the concept of Markov Chains, named after Russian Mathematician Andrey Markov, in which samples are dynamically evolved over simulation time in order to stochastically sample from a given PDF [58, 59].

The general idea is the an initial configuration is randomly distributed within the state space and subsequently evolved under a stochastic process. This stochastic process depends on the current configuration within the state space and an auxiliary PDF which govern how the current position within the state space relates to all other positions in the state space. This can be interpreted as a PDF which governs moving from one place of the state space to another. This stochastic process can be iteratively performed which results in configurations being distributed in proportion to the Born probability of a wavefunction even though we never normalised it. This methodology is an efficient work around to circumvent the inability to perform 'direct' sampling for a certain PDFs albeit some caveats.

This stochastic Markovian process, $\mathcal{F}$, is defined as,

$$X_{n+1} = \mathcal{F}(X_n, Q(X_n)) \tag{2.41}$$

where X_n is a configuration within the state space with the subscript denoting the iteration of the stochastic process, and $Q(X_n)$ is an auxiliary PDF which proposes the next configuration within the state space given the current configuration, X_n. This auxiliary PDF confines all configurations at X_{n+1} to be conditionally dependent solely on the previous configuration, X_n, as $Q(X_n)$ depends solely on X_n and not previous configurations, e.g. X_{n-1}.

Therefore, we can define a PDF which governs moving between arbitrary pairs of configurations X_{n+1} and X_n. If X_n and X_{n+1} could be independent of each other, which they are not, this PDF could be decomposed into the product of individual PDFs for each configuration. However, due to the Markovian process they are conditionally dependent and hence this joint-probability must be decomposed into a conditional PDF and marginal PDF,

$$\mathcal{P}\left(X_{n+1}, X_n\right) = Q\left(X_{n+1}|X_n\right) P_n\left(X_n\right). \tag{2.42}$$

The conditional PDF is normalised such that,

$$\int_{-\infty}^{\infty} Q\left(X_{n+1}|X_n\right) dX_{n+1} = 1 \tag{2.43}$$

which represents the probability of X_{n+1} given X_n has already occurred. Integrating Eq. 2.42 over X_n results in an updated marginal probability distribution at the next iteration, $n+1$,

$$P_{n+1}(X_{n+1}) = \int_{-\infty}^{\infty} Q\left(X_{n+1}|X_n\right) P_n\left(X_n\right) dX_n. \tag{2.44}$$

This equation is usually referred to as the 'master' equation of the Markovian process as it determines the entire dynamics of the stochastic process by iteratively solving Eq. 2.44 for any value of n. This results in a stochastic process which is defined as the Markov Chain.

This could be applied to the previous example of Eq. 2.36. In this case, the truncation of the configuration space can be ignored (as shown in Eq. 2.37) allowing for an exact calculation of π. Furthermore, this method can generalise very easily to higher dimension and in the case of expectation values of the energy lead to the concept of local operators. In fact, the main advantage is that this method allows for the ability to circumnavigate the calculation of the partition function of a PDF as moves are defined by the Hastings ratio wherein normalisation constants cancel [58, 59].

In order to define this Markov Chain in practice, there are additional conditions our Markov Chain needs to maintain in order for the method to work. One property that is not immediately obvious is how can we be sure that the PDF from which we sample remains constant under the stochastic process? This property is known as *stationarity* or as being in *equilibrium*, which is defined by the detailed-balance condition. This is a required property of any Markov Chain which is referred to as having no 'net flux' between transitioning between configuration, x_n, and x_{n+1}, and is defined as

$$Q\left(x_n|x_{n+1}\right)P_{n+1}\left(x_{n+1}\right)=Q\left(x_{n+1}|x_n\right)P_n\left(x_n\right). \tag{2.45}$$

In practice this has the consequence that if a transition from $x_n \mapsto x_{n+1}$ occurs, the joint probability of such a transition must be equal to the joint probability of the inverse transition from $x_{n+1} \mapsto x_n$. This conditions is required to ensure that the detailed-balance condition is maintained such that the same PDF is sampled throughout the sampling process [55]. This property can be shown by taking the detailed-balance condition of Eq. 2.45, and integrating the left-hand side with respect to x_{n+1} and the right-hand side with respect to x_n. This results in both sides of the detailed-balance having the same PDF between iteration of the stochastic process. This derivation is shown below,

$$\int_{-\infty}^{\infty} Q\left(x_{n+1}|x\right)P_{eq}\left(x_n\right)dx_{n+1} = \int_{-\infty}^{\infty} Q(x_n|x_{n+1})P_{eq}\left(x_{n+1}\right)dx_n, \tag{2.46}$$

$$\underbrace{\int_{-\infty}^{\infty} Q\left(x_{n+1}|x\right)P_{eq}\left(x_n\right)dx_{n+1}}_{=P_{eq}(x_n)\,(\text{Under Eq.}2.44)} = P_{eq}\left(x_{n+1}\right)\underbrace{\int_{-\infty}^{\infty} Q(x_n|x_{n+1})dx_n}_{=1\,(\text{Under Eq.}2.43)}, \tag{2.47}$$

$$\therefore P_{eq}\left(x_{n+1}\right)=P_{eq}\left(x_{n+1}\right). \tag{2.48}$$

This shows that the 'master' equation of the Markovian process maintains stationarity, i.e. the PDF that is sampled under the Markovian process is constant. In order to ensure our Markovian process works in practice there are additional requirements. A fundamental constraint is how the conditional probability, $Q\left(x_{n+1}|x_n\right)$ which dictates proposed moves within the state space, is defined. This conditional probability, or proposal distribution in the languages of PDF samplers, will only work in practice under the following additional requirements; periodicity, irreducibility, and positive recurrence.

As the stochastic process will evolve configurations throughout the state space of the PDF, it's possible that an associated transition can hold periodicity. An example of this is the stochastic process can oscillate between two states, x and x',

$$x_0 \mapsto x_1' \mapsto x_2 \mapsto x_3' \mapsto \ldots \tag{2.49}$$

where the subscript denotes the iteration, n, in the Markov chain. The periodicity is defined as the period, t, which *always* returns to its starting configuration after t-multiple iterations. For example, Eq. 2.49 has a period of 2 as it oscillates between state x and state x'. In the unique case of $t = 1$, it can return to its starting configuration in any multiples of unity (i.e. any number of iterations) and is known as *aperiodic*.

The second requirement is *irreducibility* which states that a stochastic process must be able to reach the equilibrium PDF regardless of its initialisation within the configuration space. For example, let's say we seek to sample from a one-dimensional Gaussian PDF and our stochastic process is initialised at $x = 100$. This configuration within the configuration space has low probability and is far away from the central peak of the PDF. If our Markov chain holds the property of reducibility then our sample should eventually move to $x \approx 0$ where the probability is maximal, and hence converge to the equilibrium PDF. The probability of a given move between x_n and x_{n+1} is defined by the joint probability of Eq. 2.42 which must have a non-zero probability in order for a transition to take place. However, the two aforementioned states don't have to be *directly* connected, they can be connected via any number of intermediate states.

Finally, the property of *positive recurrence* states that a given Markov chain at state x will return to state x with probability of one within a finite number of iterations. If a Markov chain is aperiodic and is positive recurrent it is referred to as *ergodic* [57]. In practice, we'd want our Markov chain to be aperiodic, irreducible, and positive recurrent in order to get a more efficient Markovian process.

The algorithmic implementation in this thesis is the classic Metropolis-Hastings Sampler [58, 59]. The initial configurations, which correspond to the many-body positions of the many-body Hamiltonian, are distributed in accordance to an isotropic Gaussian of unit width centred at the origin. They are subsequently evolved under the aforementioned stochastic Markovian process in order to be distributed in proportion to the Born probability of the wavefunction [54]. In practice, the conditional PDF is decomposed further into a transition PDF and an acceptance PDF. The transition PDF, or so-called proposal PDF, is a Gaussian distribution centred at the current iteration of a given width, σ_n, and 'proposed' moves are defined by,

$$x_{n+1} \sim \mathcal{N}(x_n, \sigma_n) . \tag{2.50}$$

A Gaussian proposal allows for all states within the state space to be connected to one another due to it having support over the entire configuration space, and Ref. [60] states that the proposal distribution can have a significant effect on the efficiency of such sampling methods. Once a proposed move is calculated, we compute the Hastings ratio,

$$\mathcal{R}(x_n, x_{n+1}) = \frac{\mathcal{A}(x_{n+1}|x_n)}{\mathcal{A}(x_n|x_{n+1})} = \frac{P(x_{n+1})\mathcal{Q}(x_n|x_{n+1})}{P(x_n)\mathcal{Q}(x_{n+1}|x_n)} \tag{2.51}$$

which represents the ratio between the joint probabilities of transitioning $x_{n+1} \mapsto x_n$ and $x_n \mapsto x_{n+1}$ respectively. The benefit of using the ratio of proposed moves is that normalisation constants cancel which circumvent the need for the calculation of the partition function. Therefore, the acceptance PDF for moving from x_n to x_{n+1} can be defined as,

$$\mathcal{A}(x_{n+1}|x_n) = \min(1, \mathcal{R}(x_n, x_{n+1})) . \tag{2.52}$$

where the $min(1,\ldots)$ statement emerges from the fact that you can't have a probability of accepting a move greater than one. To evaluate Eq. 2.52, we sample an additional value, $u \in \mathcal{U}[0,1)$ and if $u < \mathcal{A}(x_n, x_{n+1})$ the proposed move is accepted, otherwise its rejected. This step of the algorithmic process is referred to as the *accept-reject* step [53].

The accept-reject step shouldn't be confused with rejection sampling wherein *i.i.d* samples are drawn from an auxiliary normalised PDF and only accepted under a different acceptance criteria. In the case of rejection sampling all samples are eventually accepted with a given number of intermediate rejections which are discarded. These discarded samples actually break reversibility under the detailed-balance condition of Eq. 2.45 as the two accepted samples could have any number of discarded intermediate samples.

In our case, there are two circumstances: either a proposed configuration is accepted or rejected. In the case of rejection, detailed-balance is maintained by definition as both sides of Eq. 2.45 are equal. In the case of accepting a proposed move, the acceptance PDF can be determined by re-arranging Eq. 2.51 with the definition that an accepted moves has an acceptance PDF of one. The acceptance PDF for the reject and accept steps are shown respectively,

$$\mathcal{A}(x_{n+1}|x_n) = 1 \tag{2.53}$$

$$\mathcal{A}(x_n|x_{n+1}) = \frac{P(x_n)\mathcal{Q}(x_{n+1}|x_n)}{P(x_{n+1})\mathcal{Q}(x_n|x_{n+1})} \tag{2.54}$$

Where $P(x)$ is the PDF from which we wish to sample, and $Q(x)$ is the proposal distribution which proposed new configurations for the Markovian process. This has the benefit of circumventing the calculation of the partition function and in the case of a symmetric proposal distribution, i.e. $\mathcal{Q}(x_{n+1}|x_n) = \mathcal{Q}(x_n|x_{n+1})$, the Hastings ratio reduces to the Metropolis ratio. This effectively reduces our stochastic process to a random walk within the configuration space [53].

In the work of this thesis, the proposal distribution is a Gaussian distribution of a given width, σ. In order for our sampler to be efficient for all physical systems studied we follow the motivation of Refs. [57, 63] and set σ to be such that acceptance rate of our set of walkers is ∼50%. In this context, a set of 'walkers' is a set of objects which 'walk' throughout a configuration space and sample configuration in accordance to the aforementioned algorithmic process. How far these walkers moved per iteration, or epoch, is determined by the width of the proposal distribution. Therefore, in order to fix the acceptance rate to 50% we tune σ_n at epoch, n, by following the methodology of

Ref. [64],

$$\sigma_n = \frac{\sigma_{n-1}}{\left(\frac{\alpha_{targ}}{\max(\alpha_{cur},0.05)}\right)} \tag{2.55}$$

where σ_{n-1} is the previous width at epoch, $n-1$, with the first epoch having a width equal, $\sigma_0 = 1$. The term α_{targ} denotes the target acceptance rate which is set to 0.5, α_{cur} is the current acceptance rate of width σ_{n-1} which is determined by measuring the acceptance of the walkers during the accept-reject step. The 0.05 term can be interpreted as a minimum current acceptance which results in a maximum change in step size per epoch. The motivation of the terms within the denominator is that if σ_{n-1} gives the target acceptance, α_{targ}, then the denominator term will equal one, and hence, σ_n equals σ_{n-1}.

One main caveat of such a stochastic process is *autocorrelation* which stems from the fact that our samples are generated via a Markov chain where the current position of the configuration is conditionally dependent on the previous sample. The proposal process inherent to the Markovian process induces autocorrelation between iterations of the sampling process. Therefore, samples within a given chain are dependent on each other but the samples between chains are *independent* except for minimal autocorrelation induced by the tuning of the proposal width from Eq. 2.55. During the energy minimisation process (Sec. 5.1.2) we utilise the i.i.d definition for the standard deviation. Once the DNN has converged to the ground-state, we follow Refs. [56, 60] by calculating a final energy for the ground-state over multiple batches via the 'blocking' method. This method works on the assumption that even though individual samples within our Markov chain are dependent, means over subsets of the Markov chain aren't.

The work of Ref. [60] will now be explained for a single Markov chain. Let's allow a Markov Chain to evolve for N steps (or measurements) and compute the local energy at each step. This results in a set of local energy measurements, $\{E_{L,i}\}$, and the mean, μ, and variance, σ^2, can be computed via an i.i.d estimator as,

$$\mu = \frac{1}{N}\sum_{i=1}^{N} E_{L,i}, \tag{2.56}$$

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}\left(E_{L,i}-\mu\right)^2. \tag{2.57}$$

The equivalent block mean and block variance are defined by,

$$\mu_{B,i} = \frac{1}{L_B}\sum_{j=(i-1)L_B+1}^{iL_B} E_{L,j}, \tag{2.58}$$

$$\sigma_B^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\mu_{B,i}-\mu\right)^2, \tag{2.59}$$

where i is the index for the samples, j is the index for the blocks, and L_B is the number of samples within each block. Now, if the measurements were *i.i.d* then the global variance of Eq. 2.57 would be equal to the block variance, Eq. 2.59, multiplied by the length of the blocks. By comparing the global variance against the block variance (multiplied by the number of samples in a block) we can derive an efficient expression for the autocorrelation length, T_c,

$$T_c = \lim_{L_B \mapsto \infty} L_B \frac{\sigma_B^2}{\sigma^2}. \tag{2.60}$$

In the case of i.i.d sampling, the autocorrelation would equal one as it takes one sampling step to decorrelate the samples. In addition to the autocorrelation length, an additional quantity can be defined by assuming the block length is fixed and global variance is constant. This quantity is denoted the inefficiency quantity, η,

$$\eta := L_B \sigma_B^2 = T_c \sigma^2. \tag{2.61}$$

This can be utilised to compare different sampling methods, and determine which method is superior with the optimum being a method that gives *i.i.d* samples [56]. One simple step to reduce the autocorrelation within a Markov chain is a process called *thinning*. This process entails the use of taking a subset of a Markov chain at evenly spaced intervals. This interval is referred to as the thinning value and is set to 10 for all the numerical simulations in this work unless stated otherwise.

2.9 Wavefunction Ansätze

In principle, the amount of information required to describe a quantum systems is exponential in the number of particles. However, a *physical* many-body system can be represented by an amount of information much smaller than the maximum capacity of the Hilbert space [1, 10]. VMC does allow for a polynomial mapping for integration of a many-body wavefunction, however, the functional form of the wavefunction plays a key role in determining the ground-state wavefunction [65, 11].

The design of the wavefunction ansatz is fundamental in accurately measuring expectation values of observables of a given physical system [66]. The work of this thesis deals with fermions, therefore, the wavefunction must be antisymmetric with respect to particle exchange,

$$\Psi(x_1, x_2, \dots, x_N) = -\Psi(x_2, x_1, \dots, x_N). \tag{2.62}$$

This can be defined via a permutation operator, $\mathcal{P}$, which permute pairs of indices, i, and j,

$$\mathcal{P}_{ij}\Psi(X) = (-1)^P\,\Psi(X) \tag{2.63}$$

with its eigenvalues being the signature of the permutation, $(-1)^P$. The permutation operator can be viewed as a bijective automorphism on the set of indices which represent many-body particles.

In addition to abiding by permutation requirements, the wavefunction amplitude must also be continuous with respect to position. This continuity extends as well to its first derivative with respect to position too [67]. The only exception is at non-analytic points within the potential, for example with Coulombic interactions [67]. The boundary conditions of our physical system are *open* and require that the wavefunction amplitude goes to zero as the distances goes to infinity. This requirement also allows our wavefunction to be purely real [68]. These three requirements enforce restrictions on the type of trial wavefunctions that can be used and define its asymptotic behaviour.

2.9.1 Traditional Ansätze

When designing ansätze for wavefunctions we're effectively constructing an eigenfunction of a Hamiltonian which abides by symmetry constraints. The most basic forms of wavefunctions for fermions (bosons) are antisymmetrised (symmetrised) products of single particle orbitals. For fermions, these are defined as,

$$\Psi(X) = \sum_{\sigma \in \mathcal{S}_n} \left(\text{sgn}(\sigma) \prod_{i=1}^{N} \phi_i(x_{i,\sigma_i}) \right) \tag{2.64}$$

where sgn(σ) is the signature of the permutation, σ which is a set member of the symmetric group, $\mathcal{S}_n$. The set of orbitals, $\{\phi_i\}$, are chosen from a suitable single-particle basis set, and x_i is the one-dimensional position. The antisymmetry requirement is represented by a Slater determinant, however, it does hold limitations in accurately representing the wavefunction of particular interactions [69]. For example, Ref. [70] highlights a scenario in which a Slater determinant is limited to describing a system of hard spheres. The interaction of such a system is defined by,

$$V(x_{ij}) = \begin{cases} \infty, & \text{if } x_{ij} \leq x_0 \\ 0, & \text{otherwise} \end{cases} \tag{2.65}$$

where $x_{ij} = \left|x_i - x_j\right|$ is the distance between fermions i and j, and x_0 is the radius of the hard sphere. If the ansatz of the wavefunction were a Slater determinant it would abide by the antisymmetry constraints but not constraints set by the hard spheres. This interaction has the physical interpretation that two arbitrary fermions cannot be within a radial distance of x_0 from each other, i.e. $|\Psi|^2(x_1, \dots, x_N) = 0 \forall x_{ij} < x_0$. This corresponds to no two fermions overlapping. However, a Slater determinant ansatz doesn't enforce this hard-sphere requirement innately, and hence cannot represent such a physical system accurately. In order for the wavefunction to accurately represent this

Hamiltonian is must abide by,

$$\Psi(x_1,\ldots,x_N) = 0\,\forall x_{ij} < x_0. \tag{2.66}$$

This can be done by simply multiplying the Slater wavefunction of Eq. 2.64 with a function which is 0 if any arbitrary pairs of electrons overlap. This missing factor is referred to as a 'Jastrow' factor [70]. The simplest way of enforcing this property is to perform a product over all pairs of particles which abide by Eq. 2.66,

$$\Psi(x_1,\ldots,x_N) = \left[\sum_{\sigma\in\mathcal{S}_n}\left(\mathrm{sgn}(\sigma)\prod_{i=1}^{N}\phi_i\left(x_{i,\sigma_i}\right)\right)\right]\prod_{i<j=1}^{N} f(x_{ij}) \tag{2.67}$$

where f is arbitrary function that kills the wavefunction if two fermions overlap, and is a constant in the limit of two fermions at large distance,

$$f(r_{ij}) = \begin{cases} 0, & \text{if } r_{ij} \leq r_0 \\ 1 - \frac{\exp(-\beta(r-r_0))}{\frac{r}{r_0}}, & \text{otherwise}\, r_{ij} > r_0 \end{cases} \tag{2.68}$$

with β being a variational parameter to allow the Jastrow factors to adapt the wavefunction. This allows for the wavefunction to realistically represent such physical system. The Jastrow factor isn't of a fixed form like Eq. 2.67 but can be changed to represent the constraint of the physical system of interest. For instance, Ref. [71] has a Jastrow factor that contains one- and two-body pseudopotential terms to lower the expectation of the energy.

One key property of fermionic wavefunctions are nodes [72]. A 'node' in this context corresponds to the manifold of positions where many-body wavefunctions equal zero. In order to accurately represent the ground-state wavefunction of a many-body Hamiltonian the nodes of antisymmetric function must be accurately represented too. In order to calculate the exact ground-state energy, the exact nodal surface is required, however, the exact location of the nodes are generally unknown [72]. Within the aforementioned sections, the nodes of the wavefunction are primarily defined by the Slater determinant component of the wavefunction with some additional nodes emerging from the Jastrow factor. The set of these nodes define a geometric property of the wavefunction denoted the 'nodal surface'. If the many-body Hamiltonian were to have a dimensionality of d with N particles its configuration space would be a $d \cdot N$ space. This set effectively defines a hyperplane within the space of many-body positions and defines the borders of all nodal cells where the nodal cell is a subspace within the space such that all positions within such region have a non-zero amplitude. In the case of this thesis we work with 'spinless' one-dimensional fermions so $d = 1$, therefore, particles can only exchange by passing through one another. This results in an upper bound on the number of nodal cells which for N objects is $N!$ number of permutations.

The concept of nodal cells leads to the *Tiling property*. Hamiltonians which are local in position and have finite matrix-elements hold the Tiling property which has the effect of reducing a many-body wavefunction to being represented as a single nodal cell with all other nodal cells being copies linked via permutation symmetry [72]. Here all nodal cells have the same structure albeit with a differing sign factor which has direct consequences for accurately representing the nodal cells of a many-body wavefunction as all other nodal cells can be modelled by permutation symmetry. A pedagogical example of this is the ground-state wavefunction for the HO Hamiltonian for one-dimensional fermions without spin. This wavefunction can be seen in Fig. 2.1 in which we can clearly see a symmetry along the diagonal of the configuration space. This diagonal line is the aforementioned $d \cdot N$-1 hyperplane which in the $d \cdot N = 2$ space equates to a line. This is shown as a dashed line in Fig. 2.1. This figure visually highlights the aforementioned permutation symmetry of the Tiling property with two nodal cells being identical notwithstanding a sign factor.

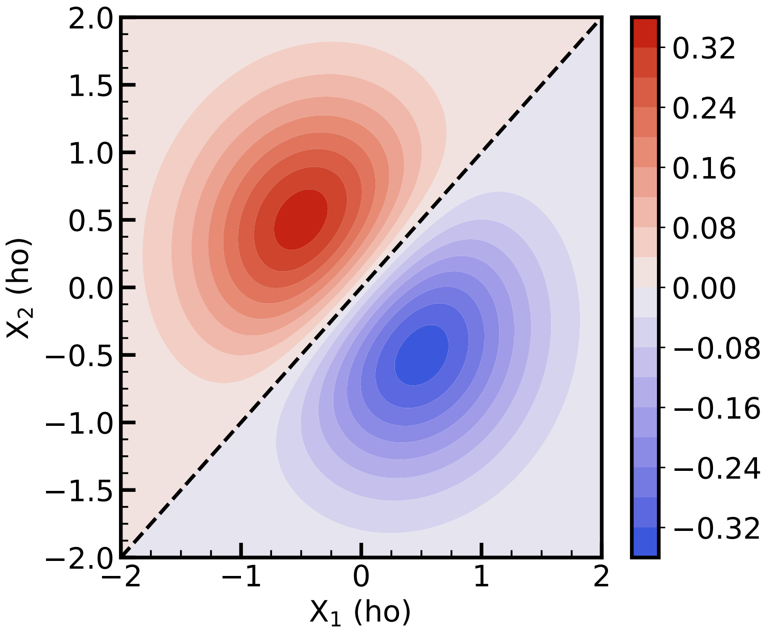


Figure 2.1: The ground-state wavefunction for two fermions in one-dimensional space under the HO Hamiltonian. This contour plot highlights the Tiling property of antisymmetric functions with two identical regions that differ only in sign which emerges from permutation symmetry.

An antisymmetric product of single particle orbitals leads to an antisymmetric wavefunction, and hence, the Jastrow factor that incorporates additional physical constraints to the wavefunction must be symmetric with respect to particle exchange in order for the entire wavefunction to remain antisymmetric [71]. A consequence of this restraint is that the Jastrow factor cannot change the existing nodal surface besides introducing new nodes in the case of two fermions overlapping for the former hard sphere interaction. This is a result of the Jastrow factor purely being a multiplicative factor, therefore, it can only scale the amplitude of the wavefunction and not translate any nodes of the nodal surface. At most, the Jastrow factor can only change the angle through which the amplitude changes sign at the node not where it's located. The knowledge of the nodal surface is vital in achieving accurate ground-state energies and must be incorporated into the wavefunction ansatz [72]. The nodal structure of the wavefunction is purely determined by the Slater component of a Slater-Jastrow wavefunction [73].

As stated the nodal structure is primarily dependent upon the Slater determinant component of the wavefunction, and hence it is directly related to the set of the orbitals used to construct the Slater determinant [71]. Therefore, if the ansatz of the wavefunction cannot accurately represent the ground-state wavefunction, the limitation can reside in the functional form of the orbitals. A proposed solution to this is to enforce higher-order correlations effects within the wavefunction like backflow [74] and three-body [73] correlations. In fact, in particular scenarios the use of backflow correlations are crucial in explaining interactions between quasiparticles with different spins [73].

A Slater-Jastrow wavefunction has the form of,

$$\Psi_{SJ}(x_1,\ldots,x_N) = \left[\sum_{\sigma\in\mathcal{S}_n}\left(\mathrm{sgn}(\sigma)\prod_{i=1}^{N}\phi_i\left(x_{i,\sigma_i}\right)\right)\right]\exp\left(-\sum_{i<j}^{N}u(x_{ij})\right). \tag{2.69}$$

The addition of backflow and three-body correlations are defined by,

$$\Psi_{SJ-B3}(R) = \left[\sum_{\sigma\in\mathcal{S}_n}\left(\mathrm{sgn}(\sigma)\prod_{i=1}^{N}\phi_i\left(x_{i,\sigma_i}\right)\right)\right]\cdot\exp\left(-\sum_{i<j}^{N}\tilde{u}(r_{ij}) - \frac{\lambda_T}{2}\sum_{l=1}^{N}G(l)\cdot G(l)\right). \tag{2.70}$$

where,

$$x_i = r_i + \sum_{j\neq i}\eta(r_{ij})\left(r_i - r_j\right) \tag{2.71}$$

$$G(l) = \sum_{i\neq l}^{N}\zeta(r_{li})(r_l - r_i) \tag{2.72}$$

and the backflow, BF, and three-body, 3B, correlation functions are defined respectively as,

$$\eta(r) = \lambda_B\frac{1+s_B r}{r_B + w_B r + r^{7/2}} \tag{2.73}$$

$$\zeta(r) = \exp\left(-\frac{(r-r_T)^2}{w_T^2}\right). \tag{2.74}$$

where λ_B, r_B, s_B, w_B, w_T are variational parameters of the ansatz. These higher order correlations are also required to abide by the given asymptotics of the Hamiltonian [75].

Eq. 2.71 redefines the fermions positions to quasiparticle positions which are dependent over all positions allowing the nodal surface to be changed, and Eq. 2.72 allows for higher order correlations to be included within the wavefunction. As stated in Ref. [73], it was found that backflow correlations improved ground-state energies at high densities and three-body correlations improved predictions for low densities.

One efficient property of a SJ-wavefunction ansatz, Eq. 2.69, is that changing the position of one particle only changes one row/column of the ansatz, therefore, one can use efficient updates technique to calculate the wavefunction amplitude without having to recompute the rows/columns of the Slater determinant which didn't change. However, when utilising backflow within the wavefunction this efficient property is destroyed in order to more accurately represent the wavefunction. Therefore, when one particle's position is changed all quasi-particle positions changed which results in the aforementioned efficient update being no longer possible [75]. The results of Refs. [73, 75] impose a hierarchy, based on the ground-state energy, of wavefunction ansätze; SJ $>$ SJ+3B $>$ SJ+BF $>$ SJ+3B+BF.

Chapter 3

Machine Learning Theory

Having discussed Nuclear Theory and traditional approaches towards representing many-body wavefunctions, this chapter turns towards the Machine Learning theory of this thesis. It begins with Statistical Learning Theory (SLT) and the UAT to underpin the concepts of learning functions from data. It will subsequently move towards specifically designed neural networks towards QMC calculations, and how modern Machine Learning (ML) libraries can efficiently update such models via Automatic Differentiation (AD). The chapter concludes with theoretical aspects towards convergence being discussed.

3.1 Statistical Learning Theory

The basis of Statistical Learning Theory (SLT) is that the relationship between an input domain, $x \in \mathbb{R}^N$, and an output range, $y \in \mathbb{R}^M$ are defined via some $f : \mathbb{R}^N \mapsto \mathbb{R}^M$ mapping function, $y = f(x)$. This function is unknown, but SLT states that such a function can be learned via testing a hypothesis set, $\mathcal{H}$, which contains all suitable functions we wish to test. An arbitrary function, $h \in \mathcal{H}$ can be found such that it approximates $f(x)$ to a given accuracy. This can be done by defining a dataset, $\mathcal{D}$, over which pairs of data, (x, y), are defined. We can assign a metric to $h \in \mathcal{H}$ under the dataset, $\mathcal{D}$, to access how well such a function performs in reproducing the true function, f. In principle, if this function is learnable from the dataset and our model, h, can accurately predict items within the dataset, it *should*, be able to predict new data outside the dataset by interpolating/extrapolating its learned domain.

When measuring the performance of our numerical model there are three components that define the total error: in-sample error, out-sample error, and statistical noise. The in-sample error is defined as the error the numerical model has from not being accurate enough to represent the data on which it is trained. The out-sample error is defined as the error the numerical model has on unseen data, and can be thought of as akin to the model's ability to generalise to the true function. The statistical noise emerges due to stochastic relationship between the data and the underlying function. In the infinite data limit, a given numerical model will converge to a minimum error denoted the *bias* [76]. This emerges from the two aforementioned errors; if the numerical model lacks the ability to truly represent the underlying function increasing the amount of data will increase the in-sample error. On the other hand the out-sample error will decrease as the amount of data increases due to an ever reducing amount of statistical fluctuations from finite-sampling effects. This phenomena is shown in Fig. 3.1,

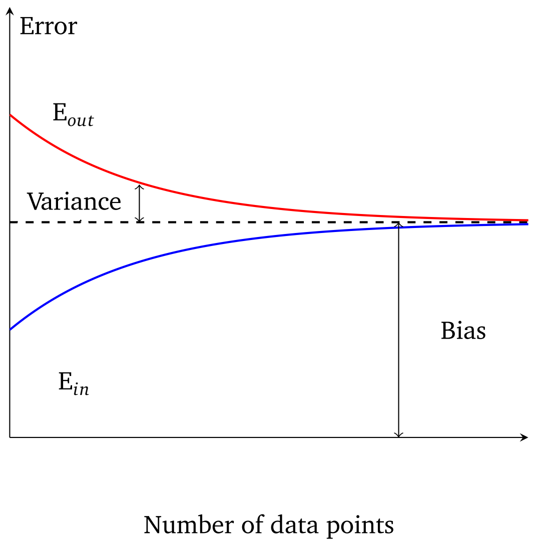


Figure 3.1: The in-sample, E_{in}, vs out-sample, E_{out}, errors as a function of the number of data points which represent the training error, and generalisation error respectively [76].

The bias of a numerical model is directly linked to the model's complexity [76]. This property is quite nuanced and hard to specifically define, but the number of parameters can be used as analogue. As the number of parameters increases the model's bias lowers due to the fact that the 'new' parameters, as compared with a smaller model, can be used to represent the existing model's error such that the bias can always be lowered. For a more detailed explanation, read Sec. 3.3. Due to having a fixed amount of datapoints during training, a more complicated model will have a lower bias but an increased *variance*. This is because as the parameter space of the model increases for a fixed amount data, the model can represent the current dataset in numerous ways. This is referred to as overfitting in the ML literature [77]. For example, visualise some data distributed to a quadratic polynomial with some Gaussian noise. We could fit this model with a quadratic polynomial and see general agreement, subject to noise constraints, however, increasing our polynomial model to a cubic or quartic model would yield inferior results even though it's a more expressive numerical model as the higher order

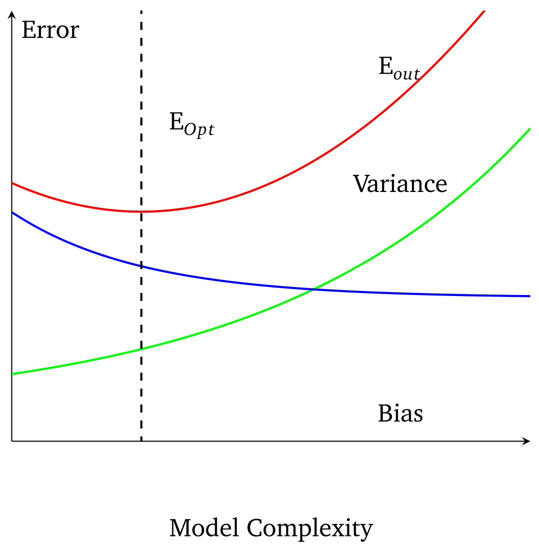


Figure 3.2: The total generalisation error against model complexity (usually defined by the number of variational parameters) for a fixed amount of data. The bias reduces due to arguments made in Sec. 3.3, and the variance increases due to finite-size sampling effects.

components fit to the statistical noise and not the true underlying distribution. This competing concept between lowering a model's bias and variance jointly becomes a trade-off as lowering a model's bias, by defining a larger model, leads to larger variance. And, reducing the variance of a model leads to an increase bias. This is referred to as *bias-variance trade-off* [76]. As working in the infinite data limit is intractable it can sometimes be better to work with a simpler model that holds a higher bias but lower variance. The bias-variance trade-off can be decomposed into constituent terms; the bias, variance, and statistical noise. Let's assume we have some statistical relationship between x and y which is defined over a dataset, $\mathcal{D}$, with associated statistical noise, ϵ. Let's use a numerical model, denoted by $\hat{f}$ with variational parameters θ, to map the input, x, to its predicted output, $\hat{y}$,

$$\hat{y} = \hat{f}(x;\theta) + \epsilon. \tag{3.1}$$

We can define an associated error via a cost function, e.g. mean-squared error, between our prediction, $\hat{y}$, and the true relation, y, which is defined as follows,

$$\mathcal{C}(y,\hat{y}) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[(y-\hat{y})^2\right] \tag{3.2}$$

where the expectation value is calculated over the associated dataset. The bias term can be mathematically written as,

$$\text{bias}^2 = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\mathbb{E}_{x\sim\mathcal{D}}\left[\hat{f}(x_i)\right] - f(x_i)\right]^2 \tag{3.3}$$

and represents how 'far' our numerical model differs from the true distribution under the current cost function which acts as a metric under which training is done. The variance term is mathematically defined as,

$$\text{variance} = \mathbb{E}_{x\sim\mathcal{D}}\left[\left(\hat{f}(x) - \mathbb{E}_{x\sim\mathcal{D}}\left[\hat{f}(x)\right]\right)^2\right], \tag{3.4}$$

and measures the spread of our numerical model in predicting new outputs that were not in the dataset. These terms respectively correspond to the asymptotic performance in the infinite data limit, and finite-size sample effect [76]. Finally, by including the statistical noise, denoted ϵ, we can define a total *generalisation error*,

$$E_{out} = \text{bias}^2 + \text{variance} + \text{noise}. \tag{3.5}$$

Although, this bias-variance decomposition is defined within a linear regression context it generalises to arbitrary learning problems where model complexity and data-samples can severely affect model performance. The work of this thesis focuses on application of VMC with Deep Neural Networks (DNNs) so it more naturally aligns with Reinforcement Learning (RL). This branch of ML focuses on improvement a model's performance via providing reinforcement signals rather than explicitly fitting a target function. In the context of learning many-body wavefunctions it has been referred to as Self-Play RL because the input 'data' is dependent on the network itself due to importance sampling [5].

3.2 Feed-Forward Neural Networks

Feed-Forward Neural Networks (FFNNs) are a subclass of Artificial Neural Networks (ANNs) that are inspired by how the brain work. ANNs consist of processing elements called 'neurons' where all neurons are connected to a subset of the total neurons within the network. These neurons weigh how important signals are from all connected nodes, and allow the network to represent data. These neurons are artificial they hold inspiration from biological neurons within the brain due to their interconnectivity and ability to share information with each other. Within the following subsection, the basics of a FFNN will be described.

3.2.1 Neurons and Linear Layers

Within Feed-Forward Neural Networks, the fundamental computing units is the 'neuron'. Mathematically, the neuron is effectively an $\mathbb{R}^d \mapsto \mathbb{R}^1$ function which takes in the output of other neurons, represented by a vector, and outputs a scalar value denoting the weighted importance of all previous neurons. As all inputs are weighted, the neuron is effectively defined as a dot-product between a learnable weight vector and an input vector. Each neuron has an associated threshold or 'bias' which is the minimal amount of 'signal' strength needed in order to 'fire'. From a mathematical viewpoint, the bias component to a neuron allows for a translation degree of freedom when minimising an error. A single neuron is quite a simple transformation from $\mathbb{R}^d \mapsto \mathbb{R}^1$, however, we can stack multiple neurons in parallel to allow for the creation of a linear map that maps an input vector $\mathbb{R}^d$ to a $\mathbb{R}^H$ where the H denotes the number of hidden nodes within a given layer. This is referred to as the 'width' of a hidden layer [78, 79]. The output of a Linear layer is passed through a non-linear activation function because stacking Linear operations fundamentally simplifies to a Linear operation. By placing an element-wise non-linear activation function between affine transformations, we can allow for highly flexible non-linear models. For example, a one-dimensional model can be defined as,

$$y = m_2\sigma(m_1 x + b_1) + b_2, \tag{3.6}$$

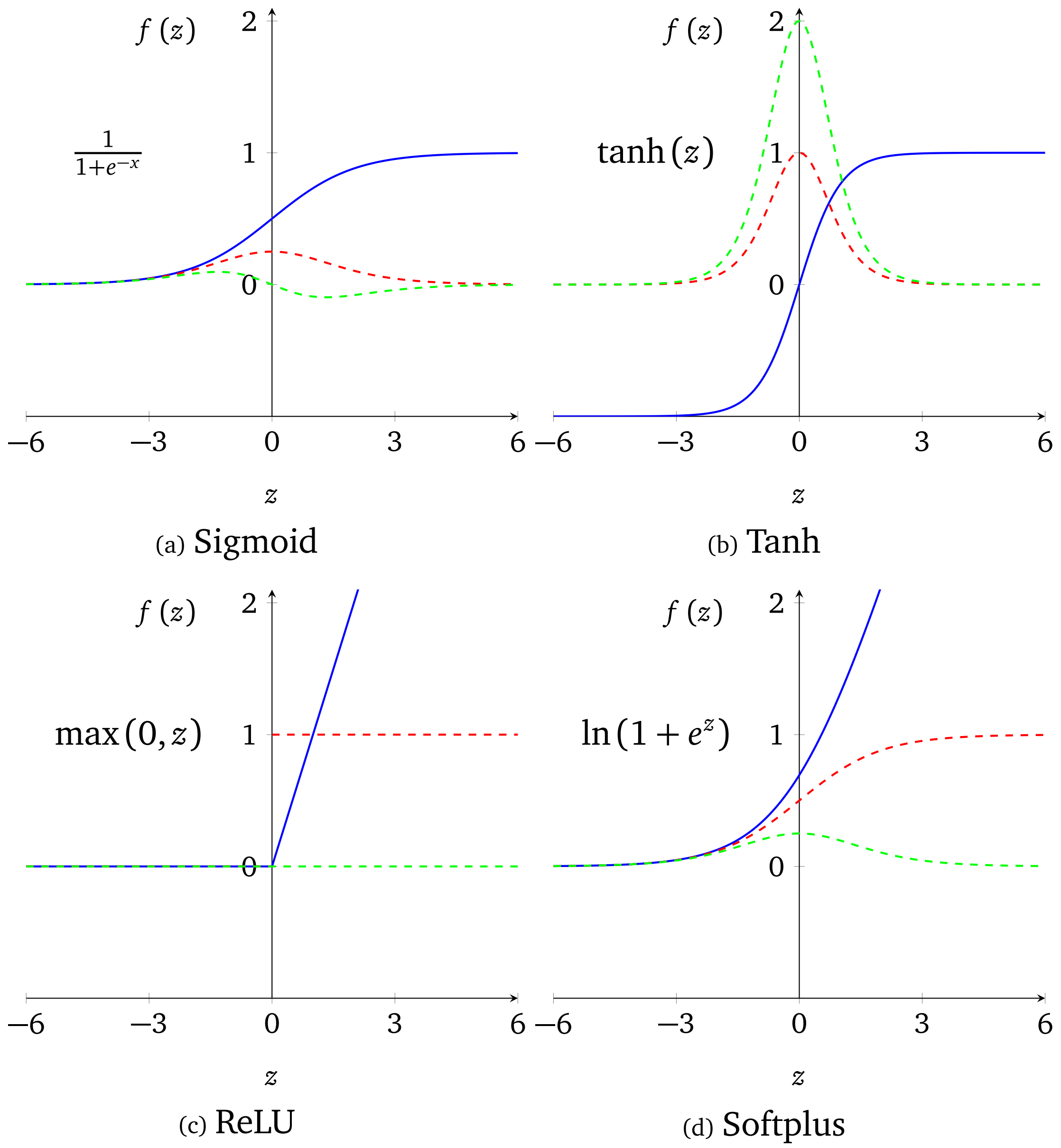
where m_ℓ, b_ℓ, and σ represent the weights, biases, and non-linear activation function respectively.

3.2.2 Activation Functions

The activation function, σ, defines the output range of the neurons and the training can be highly dependent on the particular activation function. As our model learns by some form of gradient descent, the overall derivative of the model is dependent on the gradient of the activation functions. An additional complexity in the setting of this thesis is that our loss function will contain a kinetic energy contributions which is defined by the Laplacian of the wavefunction with respect to the many-body position. Therefore, the second derivative of the activation function is vital in accurately modelling many-body wavefunctions which is not common in traditional ML. In Fig. 3.3, four main activation functions are stated with their output, first derivative, and second derivative highlight in blue, red, and green respectively to show the given properties of each function. The top row of activations functions, Figs. 3.3a, 3.3b, represent bounded activations; Sigmoid and hyperbolic tangent (or Tanh) respectively.

The 'boundedness' of these functions indicate that for large magnitude, $|z|$, they saturate to a fix output which involves a first derivative and, consequently, a second derivative equal to zero. This results in the commonly know *vanishing gradient* problem [80]. As the gradients are calculated by the chain rule, if an intermediate gradient is close to zero then gradients will fail to propagate to the remaining layers. This propagation of vanishing gradient will affect a subset of the gradients within the network and depends on which automatic differentiation mode is used (more information is given in Sec. 3.6.2 and Sec. 3.6.3). This can be mitigated by choosing different activation functions, e.g. ReLU [81], or by using skip-connections to help propagate gradient signals [82].

The bottom row, Figs. 3.3c, 3.3d, represent examples of 'unbounded' activation functions; ReLU and Softplus respectively. The Rectified Linear Unit (or ReLU) rose to prominence



(a) Sigmoid (b) Tanh

(c) ReLU (d) Softplus

Figure 3.3: A selection of popular activation functions: a) Sigmoid, b) Tanh, c) ReLU, and d) Softplus. The top row contain bounded activation functions, and the bottom row contain unbounded activation functions. Each panel shows the activation function (solid line), its first derivative (red lines), and its second derivative (green line).

from Ref. [81] and is a potential solution to the vanishing gradient problem [80]. As the derivative of ReLU(x) with respect to x is unity for $x > 0$, it is immune to the vanishing gradient problem. However, for $x < 0$ its derivative is zero leading to the neuron 'dying' leading to the vanishing gradient problem. In fact, the on/off behaviour of ReLU effectively leads to a sparsity of connections [83]. In our work case, ReLU is a not suitable activation function due to the fact that its second derivative equals to zero. This means that a network containing such a function has no second derivative with respect to its inputs nor parameters. Given that the neural network is representing a many-body wavefunction, its kinetic energy is related to the Laplacian of the wavefunction with respect to its input. If the network has no second derivative then it has no kinetic energy and cannot be used to compute realistic many-body energies.

The continuous form of ReLU is the 'Softplus' activation function, which solves the second derivative issue. Interestingly enough, it's actually the integral of the Sigmoid activation function with respect to its input which might motivate development of future activation function to start from its derivative and find the corresponding activation function. The activation function of this work is dependent upon the physical system being studied. In the case of the deuteron (Chapter 4) both the sigmoid and Softplus activation functions are used and compared. For the more general case of the Harmonic Oscillator (Chapters 5 and 6), the hyperbolic tangent function is used which follows the work of Refs. [5, 64, 6]. Some initial simulations were performed with use of the Softplus activation function, although, Tanh performed better not only in reducing bias but also variance of many-body energies. One potential reason for Tanh performing better than Softplus is due to the boundary conditions of the neural network. Open boundary conditions of a many-body wavefunction require that its amplitude must decay to zero for large distances from the origin [68].

This is done in practice via employing the use of envelopes which enforce such behaviour. As can be seen in Fig. 3.4, we show the output of a single node within a DNN with a given activation function multiplied by an Gaussian envelope function of fixed width. The orbitals of the wavefunction are heavily dependent on the chosen activation function as that directly affects is non-linearity, and this highlights what the shape of the orbitals will look like before the determinant function. The envelope enforces the correct asymptotic behaviour for all activation functions. In the non-interacting case, the exact orbitals are Hermite polynomials within the classically allowed regions, and exponential decays within the classically forbidden regions [50].

All activation functions have the correct asymptotic behaviour, but only tanh holds negative range. When constructing the single-particle orbitals for the HO interaction the exact ground-state solutions are Hermite polynomials within the classically allowed regions and exponential decays within the classically forbidden regions [50]. For a single node, the use of the tanh activation function allows it to have a node at the origin whereas all other functions have nodes at large $|z|$ which forces the equivariant layers to hold weights are larger magnitude.

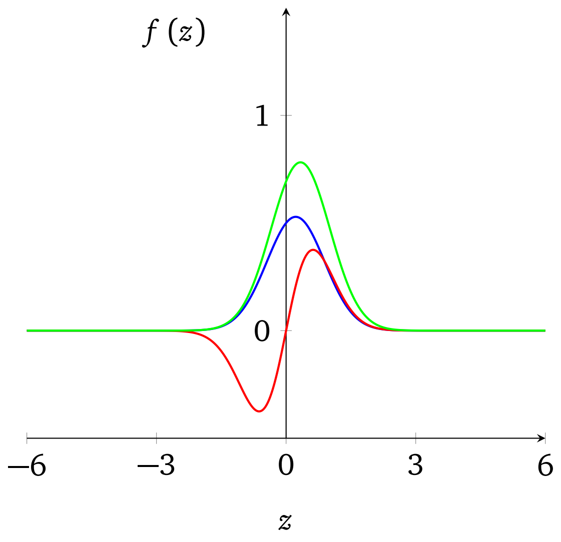


Figure 3.4: Three activation functions with a Gaussian envelope of unity width, the functions are as follows; Sigmoid (blue), Tanh (red), and Softplus (green)

3.3 The Universal Approximation Theorem

The first theoretical work to show that a single layer (or shallow) neural network can, in principle, represent any continuous function was Ref. [13]. It is effectively defined as a weighted linear combination of a bounded non-linear function applied to an affine transformation of the input domain,

$$y_i = \sum_{j=1}^{N} \mathcal{W}_{2,ji}\sigma\left(\mathcal{W}_{1,ji}^T x_j + b_{1,i}\right) \tag{3.7}$$

where $\mathcal{W}$ and b represent the weight and bias respectively, the subscript denotes the 'layer' index, ij represents individual matrix elements of weight/bias between pairs of nodes, and σ represents the activation function. The affine transformation of the input is referred to as 'neural node' within Deep Learning literature [84]. The original form of activation functions where referred to as 'squashing', 'bounded', or 'sigmoidal' functions which have the following properties,

$$\sigma(x) = \begin{cases} 1, & x \mapsto \infty \\ 0, & x \mapsto -\infty. \end{cases} \tag{3.8}$$

The main case for using such a type of function is that sums of Eq. 3.7 are dense in the space of continuous functions as long as σ is a continuous sigmoidal function regardless of the function being approximated [85]. This allows such a network to represent any continuous function to an arbitrary degree as long as the number of hidden nodes are provided ad libitum. In fact, Ref. [86] showed a shallow network can provide a perfect approximation, although, it requires a continuum (i.e. infinite) number of hidden nodes. The Universal Approximation Theorem (UAT) is more of an existence theorem rather than directly solving a given problem.

In the multidimensional case, Kolmogorov's representation theorem states that any continuous N-dimensional function is representable by combination and composition of multiple non-linear one-dimensional functions [87]. The theorem is succinctly shown as,

$$f(x_1, ...x_N) = \sum_{q=0}^{2n} \Phi\left(\sum_{p=1}^{n} \lambda_p \psi\left(x_p + \eta q\right) + q\right) \tag{3.9}$$

where Φ and ψ are one-dimensional functions and the latter is a sigmoidal function of Eq. 3.3c, and λ_p and η are variational parameters that need to be found, or learned in the modern language of ML. However, current neural networks architecture are based upon an iterative nature of a linear mapping function and the *same* non-linear element-wise activation function between all layers to allow such networks to approximate non-linear functions. This results in the theory moving from exactly representing a given multi-dimensional function to a universal approximation theorem where the multi-dimensional function is approximated to an arbitrary degree of accuracy. In practice, this is done via using a neural network with a given number of hidden nodes that can approximate any function in the space of continuous functions. This notation of combination and composition can be shown in Ref. [88] wherein a multi-dimensional function can be approximated as a sum of non-linear functions applied to a linear combination of the input domain. The key underlying point is that multi-dimensional function can be approximated to an arbitrary precision by combination and composition of one-dimensional functions thereby circumventing the need for multi-dimensional function directly.

Numerous early work highlighted this idea via different means. Ref. [89] showed completeness via Radon transformation. Radon transformations represent a function by a set of integrals over a given hyper-plane that can be approximated by a sum of finite terms. It was shown by Ref. [90] that every function in the set of continuous function with a finite limit of its input domain can be uniformly approximated by such a sum. Ref. [91] showed that a shallow network with a cosine squasher function is a special case which yields a Fourier series approximation of the target function. In Ref. [85] it was shown that a monotonic Sigmoidal function in a shallow network is complete in the space of continuous functions and are equivalent to universal approximators. This pins the limitations of such methods on three areas; inadequate learning, too few hidden units, and a stochastic rather than a deterministic relationship in the data. This universal approximation property also has been shown to apply to two-layered networks as well [92, 93, 94]

The main result of Ref. [13] is defined via ensuring that Artificial Neural Networks (ANNs) are dense in the space of continuous functions over a unit hypercube. The degree of accuracy is measured by the supremum norm of the exact function and our approximation, and is proved via functional analysis [95]. The supremum norm is defined as the absolute difference between $G(x)$ and $f(x)$. A function is *dense* in the space of continuous functions if the function evaluation of our approximation, $G(x)$, is arbitrarily close to a function, $f(x) \in \mathcal{C}(\mathcal{I}_n)$, under a given metric. Our metric of choice is the supremum norm, and all inputs x are points within the unit N-dimensional hypercube, $\mathcal{I}_n$.

Ref. [13] shows that Eq. 3.7 is dense within the unit hypercube with respect to the supremum norm. In short, Ref. [13] shows that as long as the activation function is *discriminatory* and continuous then sums of such functions denoted, G(x) as highlighted by Eq. 3.7, are dense in $\mathcal{C}(\mathcal{I}_n)$ and can approximate functions, f(x), to an arbitrary degree of accuracy under the supremum norm, i.e.,

$$\exists\, G(x) s.t. |G(x) - f(x)| < \epsilon \,\forall x \in \mathcal{I}_n. \tag{3.10}$$

Where G(x) is our DNN and f(x) is the function we seek to approximate. The term ϵ denotes a finite scalar that represents a piece-wise tolerance or error between our approximation and the true function. This derivation is proved via the Hahn–Banach theorem and Riesz Representation Theorem [13]. Although, this initial proof requires a discriminatory and continuous function which is bounded, the boundedness requirement can be circumvented in some cases. The Stone-Weierstrass Theorem, even allows for the exp function which is complete in the space of continuous functions [96]. For example, sin and cos can be used to generate all finite trigonometric polynomials [13, 97].

After the initial paper of Ref. [13] established the universal approximation theorem under somewhat strict assumptions on the activation function, some of these assumption were shown to be unnecessary [98]. In fact, the work of Ref. [98] states that the approximating capabilities of Artificial Neural Networks emerge from their innate structure rather than a specific activation function. The previous reference used ‘squashing’ functions which are Sigmoidal in nature and even monotonic. The requirements of the activation functions are furthered reduced to being bounded and non-constant, and its approximating capabilities can be found by an L-norm measure

$$\rho_{\mu,X}(f, G) = \left[\int_{\mathbb{R}} |f(x) - G(x)|^p d\mu(x) \right]^{1/p}. \tag{3.11}$$

A common example of which being the mean-squared error with p equals 2. If the activation function were to be constant, only constant mappings can be learnt which are rare in practice [98]. Having the network approximate the higher-order gradient can be used as an additional metric for validating correctness, and within the quantum mechanical setting of this thesis this is vital when calculating many-body kinetic energies which adds an additional requirement for our activation function that don’t apply in the traditional ML setting.

Although completeness and existence have been proven, such theorems do not state how one can go about finding approximations to continuous functions. Such approximations used by Kolmogorov's original theorem are based upon a set of variational parameters. In practice, we construct a heavily parametrised function that can be varied to approximate a target function under the guidance of a loss function or metric. In the Quantum Mechanical setting, we can learn the ground-state wavefunction by varying the network's parameter such that the expectation value of the energy decreases until it plateaus to the global minimum. Another way of visualising the representation of a DNN is that every position within the parameter space denotes a specific functional form, and the entire parameter space denotes a set of learnable functions. The optimal functional form is found under the guidance of a loss function. In practice, as the number of particles increase so to does the parameter space which leads to the concept of the *curse of dimensionality* in which there is an exponential increase in complexity. The computing of expectation values with such models can be mitigated via MCMC techniques, however, when it comes to efficiently constructing DNNs we can simplify the parameter space by directly embedding symmetries into our DNNs.

3.4 Embedding Symmetries into Feed-Forward Neural Networks

In order to accurately represent wavefunctions via DNNs, we need to ensure that the DNN that's used enforces the antisymmetry requirement of fermionic wavefunctions. The first interest into applying symmetries to network occurred within Ref. [99], wherein the main theorem is the 'Group Invariance Theorem'. This theorem states that if an action of a group of a set of predicates, $\{0,1\}$, is closed then the output of a perceptron, or node in more modern language, is invariant under said action on the condition that the weights associated with such perception are chosen to be preserved by the group. In practice this manifests to restricting the weights of a DNN themselves in order to guarantee that a certain automorphism is respected [100].

In Ref. [100], the concept of a *symmetry network* was first introduced through which the initial work of Ref. [101] was expanded. It is defined as a pair of a FFNN denoted $\mathcal{N}$ (Sec. 3.2), and a group of automorphisms, Γ acting on $\mathcal{N}$, i.e. $(\mathcal{N},\Gamma)$. The weights within a symmetry network are preserved such that for all γ in Γ a given invariance is maintained. The initial proof of Ref. [101] was for a perceptron, with an additional proof in the multi-layer perceptron case, where each perceptron was a predicate defined as follows,

$$\psi(X) = h\left(-\theta + \sum_{i=1}^{n} \alpha_i \phi_i(X)\right). \tag{3.12}$$

Where h is the heaviside function, and θ and α are a set of variational parameters (or weights), and ϕ is predicate that is a function which maps the reals to the set of $\{0,1\}$. It was proven that the weights of the perceptron could be assigned such that the perceptron is invariant to a given automorphism.

An example symmetry network would be a FFNN that represents the *XOR* problem. This network has two input nodes which take predicates (represented by the set of {0, 1}) and would map the corresponding inputs to a single output node that results in unity for an exclusive-or and zero otherwise. This network can efficiently codify the *XOR*-problem via enshrining a symmetry in its weights. For example, if the input receives the input as {0, 1} or {1, 0} the output is the same, that being unity, and likewise zero for {0, 0} and {1, 1}. Now, in this working example for two input nodes there are only four training examples with two target classes so one could easily forgo the symmetry constraints and directly train to the input-output pairs. However, in the general sense enforcing symmetry requirements can lead to significantly simplified training in terms of number of examples, number of parameters, and computational walltime or any combinations thereof. Within Ref. [100] their symmetry network is directly compared to the work of Ref. [102] for the XOR problem and they found that by explicitly incorporating the symmetries into the network reduced the number of parameters from 293 to 21, and due to fewer parameters the number of epochs of training reduced from 40,000 iterations to only 350. This motivates the use of incorporating symmetry directly into a DNN for more efficient training with fewer parameters.

The initial application to enforce symmetry was to assign a given orbit, effectively a sub-group under the action of the group of weights, such that the automorphism was respected. This was later expanded upon under Representation theory [103]. More recent work has moved away from restricting assignment of the weights, to directly enforcing the symmetry under the architecture of the network itself [104, 105]. Initial work on Invariant/Equivariant Neural Networks were first published in Ref. [104] where the analogy between invariant functions and functions acting on sets were drawn. In fact, by designing a function to act on a set it, by definition, acts on an input domain in a permutation-invariant manner [104]. For an arbitrary function, $f(x)$, under a group automorphism, Γ, invariance and equivariance are respectively defined as,

$$f(x) = f(\gamma(x))\,\forall\gamma\in\Gamma \tag{3.13}$$

$$\gamma(f(x)) = f(\gamma(x))\,\forall\gamma\in\Gamma. \tag{3.14}$$

Theorem 2 of Ref. [104] states that this ‘set’ function is valid if and only if it is decomposable into the form,

$$f(X) = \rho\left(\sum_{x\in X}\phi(x)\right) \tag{3.15}$$

where ϕ and ρ are arbitrary transformations which are represented by neural networks which are referred to as DeepSets in this context. For example, in the case of this thesis the inputs are a set of A scalars which denote the one-dimensional position of all particles.

Each set member, or one-dimensional position, is mapped into a H-dimensional space via an embedding function, $\phi : \mathbb{R}^1 \mapsto \mathbb{R}^H$. This function is applied to all set members individually to result in an embedding for all positions, $h : \mathbb{R}^{A\times H}$. Subsequently all embeddings are summed together over the particle index to destroy order information which enshrine the permutation invariance [106].

In the Equivariance case, Lemma 3 of Ref. [104] states that if the equivariant function is a linear map with an element-wise activation function which maps $\mathbb{R}^M \mapsto \mathbb{R}^M$, then its equivariance is maintained if and only if its weight matrix has one weight for the diagonal, and another weight for the off-diagonal. For example, a Linear layer is defined as,

$$y = \sigma(\Theta x) \tag{3.16}$$

where $\Theta \in \mathbb{R}^{M\times M}$, $x \wedge y \in \mathbb{R}^M$, and $\sigma : \mathbb{R} \mapsto \mathbb{R}$ is an element-wise function. By restricting the allowed values of Θ, therefore, enforces the equivariance within the transformation and simplifies a matrix of M^2 weights to be of 2 weights thereby holding exponentially fewer parameters [104, 105]. Reducing the number of parameters allows for more efficient and faster training but can significantly impact the quality of training. To further elucidate this, a comparison can made against UAT. A fundamental tenant is that the number of nodes within a DNN can be expanded allowing the network to fit an arbitrary function. However, in the aforementioned case this is impossible as regardless of the size of M, there are only two parameters. This restriction therefore fundamentally inhibits such a function from arbitrarily fitting the true underlying function.

In short, the invariant DeepSet of Ref. [104] transforms each set member into some representation (or embedding) via a transformation function represented via a neural network. The representations are subsequently summed over to preserve permutation invariance by destroying ordering information [106]. The equivariance case requires a weighted combination of each set member in accordance to Lemma 3 of Ref. [104]. As the transformation, ϕ, is conducted via an invariant quantity, any permutation doesn't change the transformation itself. Therefore, any permutation to the input is propagated through to the output of the transformation as described in Eq. 3.14. In both cases, be it invariant or equivariant, as composition of such functions maintain their respective automorphisms we can stack such function compositionally to create DNNs which respect a given automorphism.

The initial limitation of Ref. [104] was that although equivariance is guaranteed by restricting the weights, there's no universal approximation. This issue was circumvented by Ref. [105]. Within this paper, the group of permutations of n elements, $\mathcal{S}_n$, is the group over which invariant and equivariant neural networks are made. A figure of such a network is shown in Fig. 3.5 below. The set of inputs, X, is passed through a representation function, ϕ, which embeds the each input into a higher dimensional space. As the representation is shared between all set members leads to reduction in the number of parameters for an equivalent mapping under a conventional layer. For

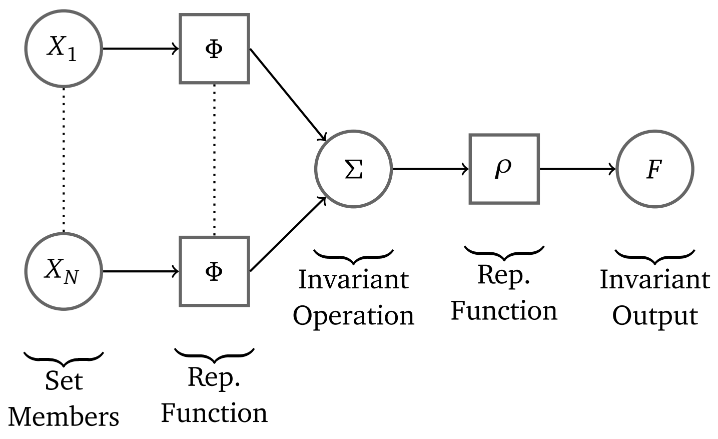


Figure 3.5: An invariant neural network from Ref. [105] which defines a permutation-invariant function $F : \mathbb{R}^N \mapsto \mathbb{R}^1$ with ϕ and ρ being representations function and the summation operation, Σ, enforcing the permutation invariance.

example, a Linear layer of A inputs and H outputs has $A \times H$ parameters, omitting any bias, whereas using a representation function approach leads to only H parameters. All set members are embedded, and subsequently summed over to ensure invariance [106].

For example, within Fig. 3.5 initially the set of values X can be viewed as a vector $X \in \mathbb{R}^{A\times 1}$, and once passed through ϕ becomes $\bar{X} \in \mathbb{R}^{A\times H}$ where the bar denotes transformation. The summation operation, Σ, is applied over the particle index, i.e. $\Sigma_a \bar{X}_{ah} = Y_h$. The transformation is agnostic to each set member any permutation corresponds to permuting rows within $\bar{X}$. And, as we sum over the columns any permutation doesn't affect the summation therefore defining an invariant function which produces an invariant embedding vector of size H. This is subsequently passed through a secondary representation function, ρ, which maps an invariant vector to a scalar output. This easily defines an invariant function,

$$f(x_1, \dots, x_A) = f(g \cdot (x_1, \dots, x_A)) \, \forall g \in \mathcal{S}_n. \tag{3.17}$$

Before expanding to the entirely equivariant case, it is key to discuss a neural network that is partially invariant by borrowing the concepts orbits and stabilizers from group theory.

For example, let's assume we have the set of natural numbers, $\mathcal{N}$, and we create a group of integers under addition mod 7, $\mathcal{G}$, with an associated action of multiplication mod 3[1].

[1] The term 'mod x' here is defined as the modulo operator which returns the remaining of a division of x, e.g 10 mod 4 equals 2.

This example yields,

$$\mathcal{N} = \{0,1,2,3,4,5,6,\ldots\} \tag{3.18}$$

$$\mathcal{G} = \{0,1,2,3,4,5,6\}. \tag{3.19}$$

Two keys concepts for groups are; orbits, and stabilizers which are defined as follows,

$$\text{orbit}\,(x) = \{g \cdot x \,\forall g \in \mathcal{G}\} \tag{3.20}$$

$$\text{Stab}\,(x) = \{g \in \mathcal{G} | g \cdot x = x\} \tag{3.21}$$

where the orbit and the stabilizer are a sub-set and sub-group respectively, with $\cdot$ representing our action, multiplication mod 3. The orbit of x is a sub-set of all group members applied to x. The stabilizer is a sub-group where all group members applied to x remain unchanged. The stabilizer doesn't have to map all group members to themselves, and hence is not equivalent to the identity function.

For the example of Eq. 3.19, the orbit of $x = 0$ is a set containing only 0 because all group members multiplied mod 3 equal to 0. In the case of orbit(1), the set contains 0, 1, and 2 as all group members are multiply by 1 then multiplied mod 3 which results in the set of $\{0,1,2\}$. For the stabilizers, we find group members under the action of x such that they remain unchanged which for Stab(0) leaves the group unchanged as all group member under the action of multiply 0 mod 3 equal 0. In the case of Stab(1), it equals a set of 1 and 4. These results are summarised below,

$$\text{orbit}\,(0) = \{0\} \tag{3.22}$$

$$\text{Stab}\,(0) = \{0,1,2,3,4,5,6\} \tag{3.23}$$

$$\text{orbit}\,(1) = \{0,1,2\} \tag{3.24}$$

$$\text{Stab}\,(1) = \{1,4\}. \tag{3.25}$$

These definitions will be needed when constructing Stab(1)-invariant function. Although abstract, these functions will be required when moving towards defining an equivariant map as defined in Eq. 3.14.

In this thesis, the functions act over the group of n permuting elements, $\mathcal{S}_n$. A Stab(1)-invariant function stabilizes element 1 and is invariant of the remaining $n-1$ elements [105]. This function is shown in Fig. 3.6.

The definition of an equivariant function, Eq. 3.14, is that applying an automorphism to a function's input is equal to applying the automorphism to the function's output, i.e. $\gamma \cdot f(x) = f(\gamma \cdot x)$. This equivariant map can be defined by creating n Stab(x)-invariant functions for each element of the set of indices, $x \in X$. As each individual function is naturally Stab(x)-invariant with respect to a given set member, mapping over all members of the set can ensure equivariance [105].

As ϕ and ρ are represented as neural networks here, the entire equivariant map becomes a universal approximator. Interestingly enough, Ref. [105] indirectly borrows from

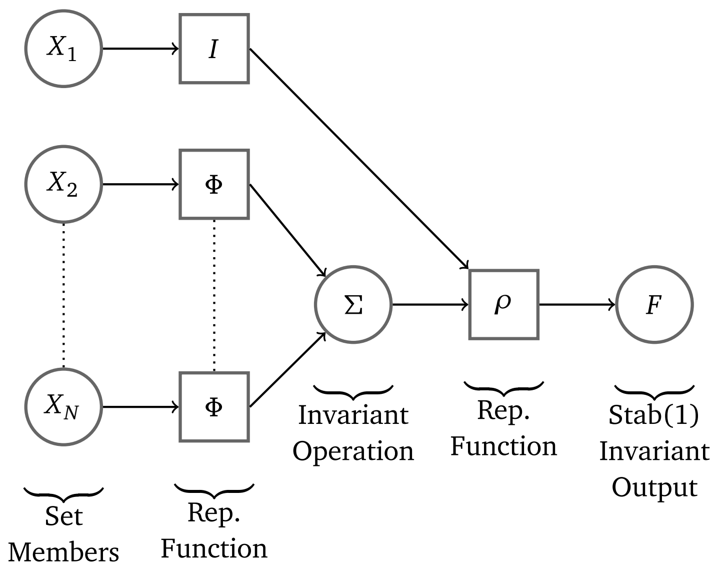


Figure 3.6: A Stab(1) invariant neural network - modified figure of Ref. [105]. The circle denote input, intermediate, and output variables that being set members, representational embeddings, and the function's output respectively. The squares denote representation functions and the identity function respectively.

the original concepts of UAT with composition and combination [87]. As there is a Stab(x)-invariant for each set member there are numerous redundant quantities that are repeated. For a given set member there is 1 identity transformation, and $N-1$ ϕ-representation functions, therefore, for the total N members of the set there are N identity transformations and $N(N-1)$ ϕ-representation transformations. However, we only need N ϕ-representations as the remaining are just repeats. Therefore, naïvely repeating this Stab(x)-invariant function N times leads to N^2-2N redundant ϕ-representations calculations. This can be efficiently defined as the concatenation of the input with its ϕ-representation. This concatenated feature, f, can then be passed through the ρ-representation function to define the equivariant embedding [105]. In practice, we follow the work of Ref. [5] and define our ϕ-representation as the identity function which when defined with the summation operation can be defined as a mean position over all particles. This is the invariant feature which is concatenate to our feature due to its simplicity and efficiency. The ρ-representation is a Linear layer which maps our invariant embedding to an equivariant embedding by exploiting the structure of the equivariant map rather than restricting the weights of Linear layers. This allows for a universal approximator to embed our inputs in an equivariant manner which is vital for quantum many-body simulations.

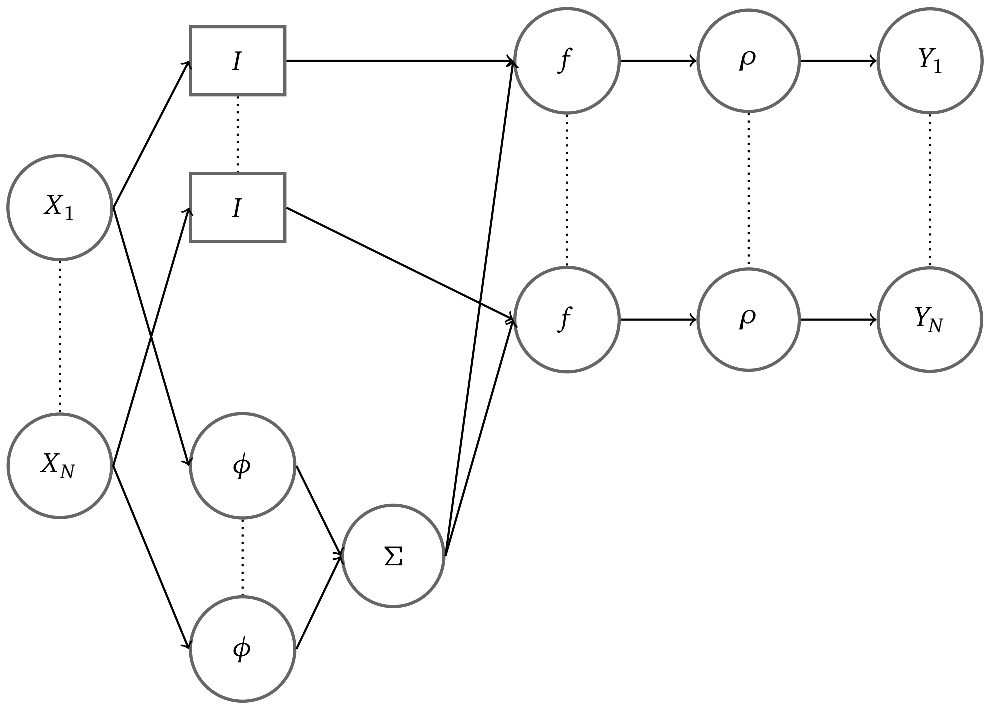


Figure 3.7: A compressed version of an equivariant map from Ref. [105] with repeated computations removed. This equivariant map is defined via a pair of neural networks which equivariantly embed input features.

3.5 Creating Neural Ansätze for Variational Monte Carlo Simulations

The ‘traditional’ ways in which wavefunction ansätze are constructed were discussed within Sec. 2.9. In this section, we extend the aforementioned ML theory towards constructing ‘neural’ ansätze for many-body wavefunctions for fermionic systems in VMC calculations. The first application of fermionic neural networks emerged within Refs. [66, 5]. In this thesis a simplified model of Ref [5] is used as our DNN ansatz. The wavefunction can effectively be broken down into four compartments which are fundamental in accurately modelling one-dimensional fermions. The first is the permutation equivariant layer which ensures that equivariance is propagated towards the Slater matrices. Next is the generalised Slater determinants (GSD) which are constructed from these matrices and create an arbitrarily correlated single-particle orbitals with backflow. In order to ensure our wavefunction abides by open boundary conditions, log-envelopes are used to enforce the correct asymptotic behaviour in a numerically stable manner. Finally the Stable Summed Signed Log Determinant (3SLD) layer is utilised. This layer takes a set of D GSDs and merges them together into a single signed-log determinant to represent the antisymmetric function. The following sections will discuss in detail each layer's contribution towards the neural ansatz.

As previously stated, in order for a wavefunction to represent fermionic systems it needs to be antisymmetric with respect to permutation of particle index. This can easily be done via the use of a determinant. However, a determinant of independent orbitals is only fully descriptive of fermionic systems in the case of non-interacting Hamiltonians. Therefore, in the work of this thesis with a finite-range interaction additional correlation is required via the use of Jastrow functions or backflow [67, 74]. By following the initial work of Ref. [104] a linear map can be constructed that is equivariant with respect to permutation which is shown in Fig. 3.8.

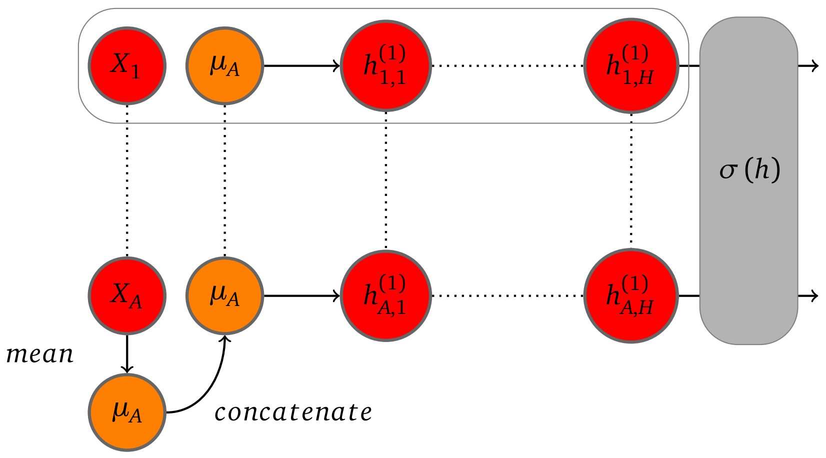


Figure 3.8: The input equivariant layer used within the Ansatz. The input positions (shown left) are concatenated with the mean position, μ_A, over all A positions. They are subsequently mapped into an embedding vector via a linear layer, denoted via an arrow, which is shared over all input positions. The entire linear layer process is encircled by a solid line. This outputs a set of A embedding vectors of length H which is denoted $h^{(l)}_{i,j}$ where i and j represent the indices particle index and hidden nodes index respectively. These intermediate features are passed through an element-wise non-linear activation function, $\sigma(h)$, to introduce non-linearity to the transformation.

This equivariant map allows for each particle's position to be embedded into a higher dimensional space of size H. It also allows for the ability of the embedding space to be systematically increased by exploiting the UAT of Sec. 3.3. The question then arises what representation functions are used for ϕ and ρ of Fig. 3.7? In Ref. [5], an equivariant layer is used and defines its ϕ-representation function as an identity function and redefines the summation operation, ρ, as a mean operation. The only difference between summation and mean is dividing by the number of particles which normalises the features against the dimensionality of the physical space. This ensures that the invariant feature is independent to the number of particles.

This is concatenated to each particle and passed through a linear layer (Sec. 3.2.1) and defines the ρ-representation function $\rho : \mathbb{R}^2 \mapsto \mathbb{R}^H$ where 2 denotes a vector of the particle's one-dimensional position concatenated with the mean position over all particles of the physical system, e.g. see the left two column of Fig. 3.8. This permutation invariant property allows for an expandable function, in terms of hidden nodes, to map the one-dimensional position into a higher-dimensional space while respecting equivariance. This is respected because by using an invariant quantity in the mapping ensures that each particle is transformed in the same way. As a permutation is performed to the ordering of the particle indices, the embedding function is invariant but the ordering isn't thereby leading to the permutation propagating through to the output [105]. By stacking multiple equivariant layers together, deep equivariant representations can be built which allow for more flexible equivariant maps.

In summary, the entire non-linear equivariant embedding takes an input vector, $X \in \mathbb{R}^{A\times 1}$, and maps it to an output feature, $h \in \mathbb{R}^{A\times H}$. For all intermediate layers, the 'invariant' feature is the mean-body position over all particle in the embedded hidden space. This defines the equivariant layer as $l \colon \mathbb{R}^{A\times 2H_{in}} \mapsto \mathbb{R}^{A\times 2H_{out}}$ where the factor of 2 emerges from the use of the mean many-body position over the entire hidden space. The terms H_{in} and H_{out} denote the dimensionality of the input and output spaces of the layer. For the first input layer $H_{in} = 2$ and $H_{out} = H$, and for all remaining layers $H_{in} = 2H$ and $H_{out} = H$. In order to ensure strong gradient flow through the embedding, we utilise residual connections between all equivariant layers [107]. The use of the mean many-body position to use to enforce equivariance within the embedding, however, it also acts as a form of generalised backflow when constructing the generalised Slater determinants. The output features, $f \in \mathbb{R}^{A\times H}$, are mapped to a set of D matrices via a set of Linear layers, i.e. $f \in \mathbb{R}^{A\times H} \mapsto M \in \mathbb{R}^{D\times A\times A}$. This transformation is defined by the following layer of the DNN ansatz which is shown in Fig. 3.9,

This set of D matrices, referred to as generalised Slater matrices (GSMs), can be antisymmetrised by calculating their determinants. Now, this has immediate similarities with the Configuration Interaction (CI) wavefunction due to the fact that the wavefunction is defined as a weighted sum of a set of Slater determinants [108, 69]. The noted improvement of these matrices is that they are completely general under the width of the hidden layer, H. The generality emerges from the mean operation within the equivariant embedding which encodes the position of all particles thereby converting the single-particles into quasi-particle orbitals that depend on *all* particles. In the case of a standard Slater determinant, the orbitals are defined as a function of a single orbital acting on a *single* particle, i.e. $\phi_i\left(x_j\right)$ where i and j denote the orbital and particle index respectively. However, under the generality of backflow the orbitals become a single orbital acting on *all* particles, i.e. $\phi_i\left(x_j; \{x_{\setminus j}\}\right)$ where $\{\setminus j\}$ denotes the set of all particles *except* particle j. This induces higher-order correlations between all particles within the physical system.

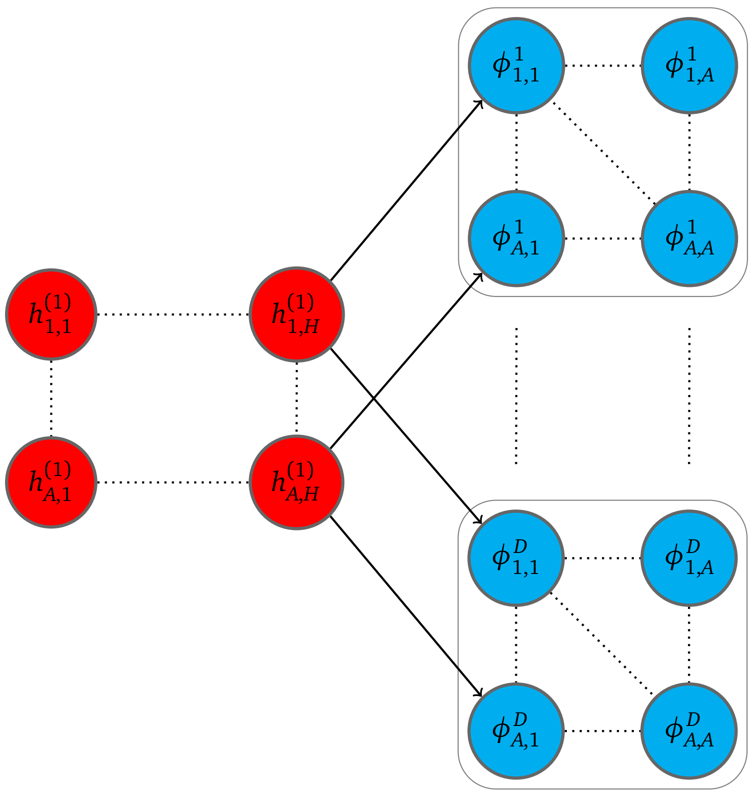


Figure 3.9: The Generalised Slater Matrices (GSMs) of the DNN ansatz. The input features (left) are the outputs of the final permutation equivariant layer. This is mapped to D GSMs (right) via a set of D linear layers. These matrices are completely general under the width of the hidden layer, H.

Indeed, in Ref. [109] it is stated that just one of these GSDs can represent any physical system, although, with some fundamental caveats of which one is discontinuity of the wavefunction. This caveat would break a main property of our fermionic wavefunction and hence a few determinants are required in practice to *exactly* represent *any* antisymmetric wavefunction. However, exponentially fewer number of determinants are required when backflow is used via these equivariant layers rather than standard Slater determinants [5]. This type of determinant is referred to as *Generalised* Slater Determinant (GSD) due to the inherent use of backflow via the aforementioned permutation equivariant layers.

In order to impose asymptotic behaviour to our single-particle orbitals, and by extension the wavefunction itself, the amplitude of the wavefunction must tend to zero as the distance increases from the origin, $x \mapsto \infty$. This can be efficiently done via the use of envelope functions,

$$e_i(x_j) = \exp\left(-b_i^2 x_j^2\right), \tag{3.26}$$

where b_i is a variational parameter that defines how rapidly the asymptotic behaviour of the envelope kicks in for the i^{th} orbital, and j denotes the j^{th} particle. These functions are required in order to represent the open boundary conditions of the physical system [68]. These are combined with the GSMs to create a highly flexible ansatz to represent the wavefunction. The implementation of this layer is performed within the log-domain for added numerical stability. This can be effectively computed as an outer product of a vector containing the variational parameters for the envelopes, b_i, and the position of the fermions, x_j.

Once the GSMs and the log-envelopes have been established the final part of the network can be defined, the Stable-Summed-Signed-Log Determinant (3SLD) function. The network contains a set of D GSMs which are antisymmetrised via a determinant call in order to represent a fermionic system. This results in the network producing D determinant values that need to be summed together to represent a total wavefunction amplitude. However, naïvely taking the determinant of such matrices can be numerically unstable and result in divergences, especially when taking second derivative which are required for the kinetic energy. Therefore, we follow and expand upon Ref. [5] and define a fused operation which takes a set of D matrices, computes a signed-log determinant for each matrix and combines all values together via logsumexp function to compute a global signed-log determinant function. This is defined as,

$$\mathrm{sgn}(\Psi(x)) = \mathrm{sgn}\left(\sum_{d=1}^{D} \mathrm{sgn}(\psi_d)\exp(\ln|\psi_d|)\right) \tag{3.27}$$

$$\ln|\Psi(x)| = \ln\left|\sum_{d=1}^{D} \mathrm{sgn}\left(\psi_d\right)\exp\left(\ln|\psi_d|\right)\right| \tag{3.28}$$

where ψ_d is defined as,

$$\psi_{d,ij} = \det\left(\mathcal{A}_{d,ij} \odot \exp\left(-\left(b_{d,i}x_j\right)^2\right)\right). \tag{3.29}$$

Where $\mathcal{A}_{d,ij}$ denotes the GSD matrix for matrix, d, orbital, i, and particle, j, with $\odot$ denoting element-wise multiplication, and $b_{d,i}$ is the log-envelope weights, and x is the many-body position. This function maps a set of determinant values, from the log-domain, into a tuple of reals that represent the global sign and global log abs value of the sum. The global log abs value of the wavefunction is effectively a log-sum-exp function, albeit with a sign factor [110]. For a CI wavefunction, all Slater determinants are summed together with an associated weight which denotes relative importance of each Slater determinant [108]. In the original work of Ref. [5], this methodology was followed by taking a weighted sum of GSDs. However, it was noted in Ref. [6] that these are redundant parameters as the GSDs are calculated via a weighted transformation this determinant *weight* can be absorbed into the function which maps the equivariant embedding of the many-body positions to the set of GSDs. This is effectively a repeat of Eq. 3.6 where the introduction of such a coefficient would effective constitute redundant parameters that negatively affect performance of such a model, we therefore follow Ref. [6] and omit such coefficients. For the sake of brevity, we omit the full derivation of this function here including the first and second derivative terms with respect to the input matrix. This is required for numerical stability in the case of singular matrices. For the full derivation, we direct the reader to App. A.

This section is concluded by defining the DNN ansatz of this work. The DNN contains L Equivariant layers of H hidden nodes each with an associated activation function, $\sigma(h)$. This can be viewed as non-linear equivariant embedding of our many-body position in a H-dimensional space which innately incorporates backflow. These embeddings are mapped to a set of D GSDs to define our orbitals. In parallel, the many-body positions are mapped to a set D log-envelope to enforce the open boundary conditions of our Hamiltonian. The D GSDs and D log-envelopes are merged towards to define a set of D matrices which are antisymmetrised via a determinant. This determinant is calculated within the log-domain for numerical stability and summed via a max-subtracted signed-logsumexp function which results in a global sign and global log abs wavefunction for the total wavefunction. The complete structure of the DNN ansatz is shown in the diagram of Fig. 3.10.

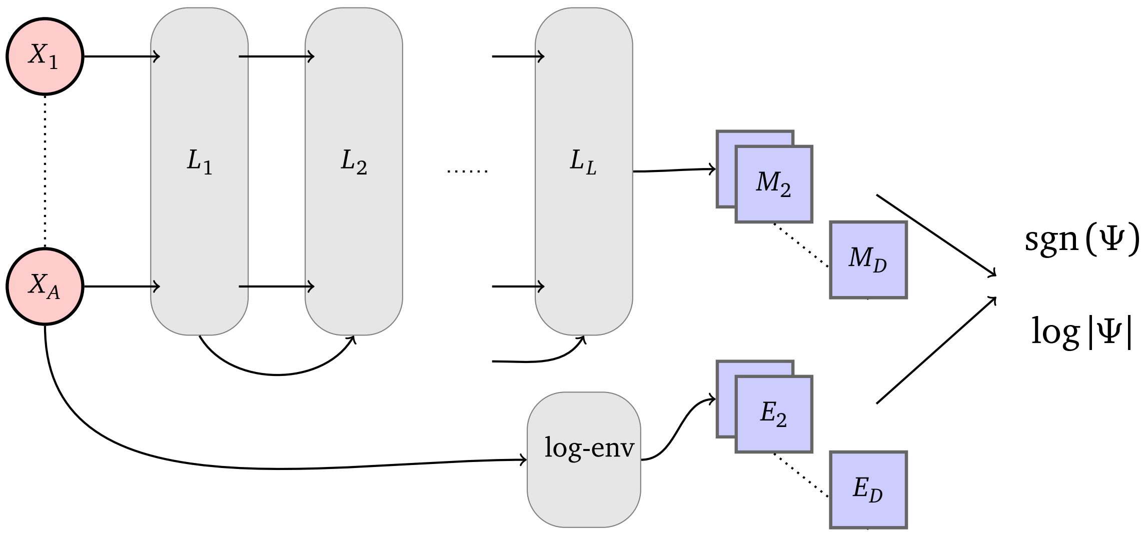


Figure 3.10: The DNN ansatz of this thesis for L equivariant layers of H hidden nodes each, D Generalised Slater Determinants (GSDs), the log-envelope layer which creates D log-envelope matrices, and summed determinant function which takes the D GSDs and computes the sign and log-absolute of the wavefunction via a custom `torch.autograd.Function` object.

3.6 Automatic Differentiation

As previously mentioned we model our wavefunction based on DNNs architectures which are randomly initialised and subsequently optimised (or 'trained' in the language of Machine Learning) under the governance of a cost function. As DNNs can have numerous number of parameters in a hierarchical-based structure ranging from the thousands to the trillions [111], we seek to find an efficient yet scalable algorithm that can easily calculate accurate derivatives for our DNNs.

There are 4 traditional methods to compute derivatives: Analytical, Numerical, Symbolic, and Automatic. All of these methods have their associated pros and cons, yet the latter will be our method of efficient derivative calculation. Analytical differentiation gives exact derivatives, however, these derivatives are restricted to the original function of choice and require hard-coded solutions for each function which exacerbate inefficiency of using such a method for DNNs.

For numerical differentiation, calculating derivatives is relatively simple to implement via forward-difference or centred-difference methods, however, this is impractical for DNNs due to the shear amount of parameters. To evaluate a derivative of N parameters requires on the order of $\mathcal{O}(N)$ operations. In addition to the linear scaling for calculating derivatives, the accuracy of numerical differentiation is affected by its step-size which for small step-sizes is dominated by *round-off* error and for large step-sizes *truncation* error occurs. In practice, numerical differentiation is never exact but only an accurate approximation.

For symbolic differentiation, a direct formula for the derivative is derived via a symbolic integrator and can give exact results. This approach has been implemented in numerous languages and even python packages [112, 113, 114]. The primary issue with symbolic differentiation is *expression swell* which can lead to exponential increase in computation walltime for evaluating derivatives. This becomes the main obstacle for calculating derivatives of DNNs via symbolic differentiation.

3.6.1 The History and Development of Automatic Differentiation and Computational Graphs

The exact inception of AD is not definitely pinned down to a specific date but initial research began in the 1950s and 1960s with Refs. [115, 116, 117] to the discovery of Forward-mode AD (discussed in Sec. 3.6.2) in 1964 [118]. Subsequently, interest in AD fell on the back-burner for research groups for unknown reasons [119] until in the 1980s, the group at Argonne National Lab (ANL) introduce Reverse-mode AD (discussed in Sec. 3.6.3) and revitalised the field with numerous conference proceeding and books [120, 121, 122, 123]. This lead to improvement in computational languages/tools, the most important of which is the *computational graph* [119]. This concept, although initially straight-forward, allows one to visualise a computational program, or algorithm, as a graph with the nodes representing operations and the edges connecting different operations together to form the algorithm. The neural networks used throughout this thesis are directed acyclic graphs (DAG) meaning that all connections between nodes flow from input to output, i.e. there are no cycles between nodes, and information flows in one direction; input to output [123]. By defining an algorithm, via a computational graph, the derivatives between particular nodes within the graph can be seen as modifications of the original DAG [124]. The first algorithmic implementations of AD were introduced by Refs. [125, 126]. In fact, numerous groups independently discovered these methods [127]. The usage of AD was popularised primarily by the Parallel Distributed Processing (PDP) group [128], and in fact they only became aware of Ref. [127] after publishing their own work.

An example computational graph would be that of Ref. [129] in which we can visualise,

$$f(x_1, x_2) = \ln(x_1) + x_1 \cdot x_2 - \sin(x_2) \tag{3.30}$$

as a DAG which can be subsequently used to compute derivatives with respect to intermediate or input variables via AD. These operations are recorded within a Wengert List which defines how all nodes within the graph relate to one another. Within the PyTorch Framework, this use of ‘tape’ is deliberately avoided and only a subset of the graph is

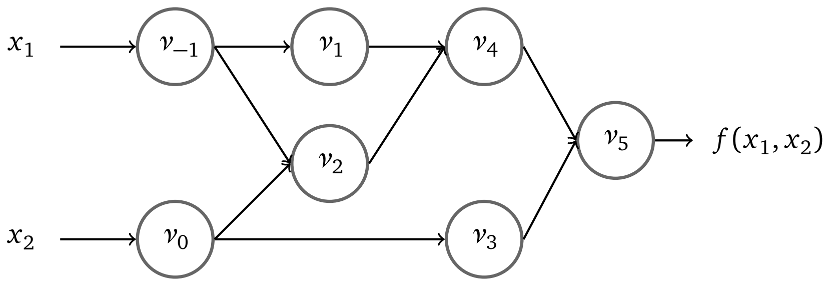


Figure 3.11: A computational graph representing the function $f(x_1, x_2) = \ln(x_1) + x_1 \cdot x_2 - \sin(x_2)$. All intermediate values, v_i, follow the notation of Ref. [131] - modified figure of Ref. [129]

stored for a given node. This allows the model to be more memory efficient as there's no point in storing parts of the graph which have no use [130]. There are two main version of AD; forward-mode, and reverse-mode. Both AD methods will be explained pedagogically with their associated pros and cons.

3.6.2 Forward-mode Automatic Differentiation

Forward-mode AD is based upon the concept of primal and tangent traces which effectively represent the output and directional derivative of a given function. The directional derivative is defined as the sum of partial derivatives of a function with respect to its inputs, i.e. $\sum_{i=1} \frac{\partial f}{\partial x_i}$. The idea is to apply the chain rule to the each operation of the primal (output) trace, in order to generate the corresponding tangent (directional derivative) trace [129]. This allows for the evaluation of the function as well as the first derivative in tandem, i.e. simultaneously in the same forward-pass.

Forward-mode AD (or forward accumulation mode) initially defines all intermediate values of a derivative. All intermediate values, v_i, follow the notation of Ref. [131] that for an $\mathbb{R}^n \mapsto \mathbb{R}^m$ function all $v_{i-n} \, \forall i \in \{1, \ldots, n\}$ represent the input variables; $v_i \, \forall i \in \{1, \ldots, \ell\}$ represent all intermediate nodes; and $v_{l-i} \, \forall i \in \{m-1, \ldots, 0\}$ represent all output variables. These intermediate derivatives values are referred to as the *tangent trace*, and are denoted as follows,

$$\dot{v}_i = \frac{\partial v_i}{\partial x_1} \tag{3.31}$$

and we can use the chain rule to multiply all tangents together to get the derivative of the function with respect to a given input. Let's compute the output of the function at a given input $(x_1, x_2) = (2, 5)$. We set the initial values of the graph as the inputs,

$$v_{-1} = 2, \tag{3.32}$$

$$v_0 = 5, \tag{3.33}$$

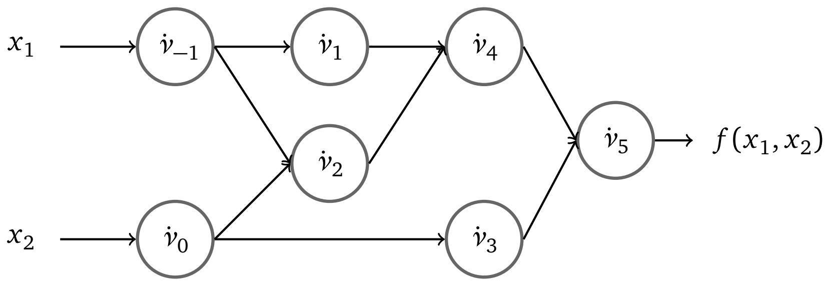


Figure 3.12: A computational graph representing the function $f(x_1, x_2) = \ln(x_1) + x_1 \cdot x_2 - \sin(x_2)$. All intermediate values, $\dot{v}_i$, follow the notation of Ref. [131] and denote the intermediate derivative with respect to the given initialisation - modified figure of Ref. [129]

and compute the function of Eq. 3.30 by using the following set of operations, all of which are shown in Fig. 3.11,

$$v_1 = \ln(v_{-1}) \approx 0.693\ldots, \tag{3.34}$$
$$v_2 = v_{-1} \cdot v_0 = 2 \cdot 5, \tag{3.35}$$
$$v_3 = \sin(v_0), \tag{3.36}$$
$$v_4 = v_1 + v_2 = 0.693 + 10, \tag{3.37}$$
$$v_5 = v_4 - v_3 = 10.693 - -0.959, \tag{3.38}$$

where v_5 equals $f(x_1, x_2)$. This is completely equivalent to passing (2, 5) into Eq. 3.30 and recording all intermediate results. Now, let's say we seek to find the derivative of $f(x_1, x_2)$ with respect to x_1. First, we duplicate the graph of Fig. 3.11 but instead of representing the primal (output) it represents the tangent (derivative) following the convention of Eq. 3.31. The input tangents are by definition equal to unity and zero respectively because we seek the derivative with respect to x_1 from the directional derivative. This is derived mathematically from differentiating our inputs with respect to the node we wish to seek the directional derivative of,

$$\dot{v}_{-1} = 1 \tag{3.39}$$
$$\dot{v}_0 = 0. \tag{3.40}$$

Next, we calculate the intermediate derivative values of the graph via the chain rule,

$$\dot{v}_1 = \dot{v}_{-1} \cdot \frac{\partial v_1}{\partial v_{-1}} = \frac{1}{2} \tag{3.41}$$

$$\dot{v}_2 = \dot{v}_{-1} \frac{\partial v_2}{\partial v_{-1}} + \dot{v}_0 \frac{\partial v_2}{\partial v_0} = 1 \cdot 5 + 0 \cdot 2 \tag{3.42}$$

$$\dot{v}_3 = \dot{v}_0 \cdot \frac{\partial v_3}{\partial v_0} = 0 \cdot \cos(5) \tag{3.43}$$

$$\dot{v}_4 = \dot{v}_1 + \dot{v}_2 = 0.5 + 5 \tag{3.44}$$

$$\dot{v}_5 = \dot{v}_4 - \dot{v}_3 = 5.5 - 0. \tag{3.45}$$

with the output derivative being defined as equal to $\dot{v}_5$ given that the output is defined by an Identity function. And, $\dot{v}_5$ equals the derivative we computed, i.e. $\frac{\partial f}{\partial x_1}$. Notice that when calculating the derivative we also used the input values (x_1=2, x_2=5) from the primal trace (Fig. 3.11) and not purely the tangent trace (Fig. 3.12). So, ideally we would like to merge both the primal trace and tangent trace into a single graph where we can utilise this arithmetic to efficiently compute primal and tangents simultaneously. This can be done efficiently via the use of Dual Numbers [132].

When visualising Dual numbers, it helps to start from the familiar viewpoint of complex numbers. Complex numbers can be viewed as a tuple of reals that define a real and imaginary component bound by their own arithmetic, such that,

$$z = a + bi \tag{3.46}$$

where i^2 is defined as -1. It should be noted that the real component $\mathrm{Re}(z) = a$ and $\mathrm{Im}(z) = b$ are orthogonal to one other and define the complex plane. Dual numbers follow a similar take of being a tuple of reals and are defined as,

$$z = a + b\epsilon \tag{3.47}$$

where ϵ is a nilpotent number meaning ϵ^2 equals null but $\epsilon \neq 0$. These pairs are referred to more broadly as Argand pairs [133]. Like the components of complex numbers, the components of Dual numbers are orthogonal. The dual component, ϵ, can be viewed as an infinitesimal number. This is because the real component can be viewed as in the basis of real numbers (1, 0), and its dual is orthogonal to that basis vector, (0, 1). As it is a non-zero quantity smaller than any real number it is by definition an infinitesimal number [120]. The arithmetic of Dual numbers follows in a similar manner to that of Complex number; e.g. adding Dual numbers is associative, and multiplication distributive with the exception of $\epsilon^2 = 0$.

These useful properties of Dual numbers can be used to compute functions, as well as their derivatives simultaneously. The evaluation of a function, $f(v)$, on its input represented as a dual can be defined as

$$f(v + \dot{v}\epsilon) = f(v) + f'(v)\,\dot{v}\epsilon. \tag{3.48}$$

which emerges from the definition of a Taylor expansion coupled with the nilpotent property. For example, let's define a function, $f : \mathbb{R}^1 \mapsto \mathbb{R}^1$ with its derivative,

$$f(x) = 5x^2 + 3x + 4 \tag{3.49}$$

$$\frac{\partial f}{\partial x} = 10x + 3 \tag{3.50}$$

and compute the evaluation and derivative for $x = 2$. This results in $f(x) = 30$, and $\frac{\partial f}{\partial x}$ $= 23$ respectively. Let's repeat this example with dual numbers, i.e. $x = 2 \mapsto 2 + 1\epsilon$,

$$\begin{aligned} f(x) &= 5(2+1\epsilon)^2 + 3(2+1\epsilon) + 4 \\ &= 5\left(4 + 2\epsilon + 2\epsilon + 1\epsilon^2\right) + 3(2+1\epsilon) + 4 \\ &= 30 + 23\epsilon \end{aligned} \tag{3.51}$$

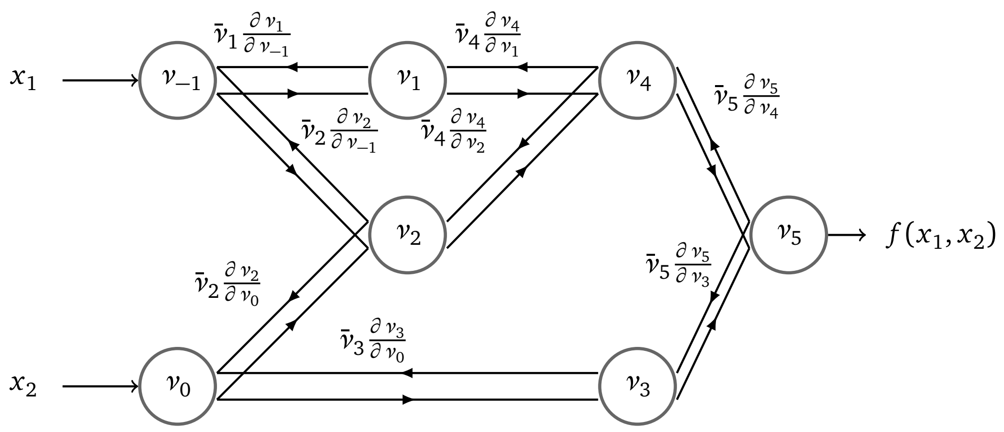
The output is also a dual number whose real component is equal to the function evaluation and the dual component is equal to the directional derivative. This can be implemented efficiently by constructing a dual datatype which will allow a DNN to compute its function evaluation and directional directive simultaneously. It also has the added benefit of merging the DAGs of Fig. 3.11 and Fig. 3.12 into a single graph which represents both the primal *and* tangent simultaneously.

This results matches the analytical expressions of Eq. 3.50. This can be implemented via constructing a dual datatype with real and dual components, and allows for neural networks to efficiently compute its function evaluations and directional derivatives simultaneously. This allows for Fig. 3.11 and Fig. 3.12 to be merge into a single graph

3.6.3 Reverse-mode Automatic Differentiation

Reverse-mode AD (or reverse accumulation mode) is another AD method to computing derivative of computational graphs. This method holds similarities with Forward-mode AD, but instead of computing primals and tangents simultaneously, it is performed in two phases commonly referred to as the *forward pass* and *backward pass* respectively. Within the forward pass the primals are computed, and intermediate variables are cached and their place within the computational graph is saved. Then in the backward pass, all derivatives are calculated via propagating adjoints in reverse, hence the name *backward*.

The entire computational graph of reverse-mode AD can be seen in Fig. 3.13 which shows both the forward pass and backward pass which are denoted by their respective edges. Let's follow the forward-mode AD example from Eq. 3.30 and repeat it for reverse-mode AD. The inputs are initialised in the same manner as Eqs. 3.32 and 3.33 with the same intermediate values of Eqs. 3.34 and 3.38 which results in $f(x_1, x_2) = 11.652$. Now, when computing the derivatives for each intermediate node the chain rule is used once again to propagate adjoints in reverse to populate the derivative of all nodes. For reverse



x_1
x_2
v_{-1}
v_0
v_1
v_2
v_3
v_4
v_5
$f(x_1, x_2)$
$\bar{v}_1 \frac{\partial v_1}{\partial v_{-1}}$
$\bar{v}_4 \frac{\partial v_4}{\partial v_1}$
$\bar{v}_2 \frac{\partial v_2}{\partial v_{-1}}$
$\bar{v}_4 \frac{\partial v_4}{\partial v_2}$
$\bar{v}_5 \frac{\partial v_5}{\partial v_4}$
$\bar{v}_2 \frac{\partial v_2}{\partial v_0}$
$\bar{v}_5 \frac{\partial v_5}{\partial v_3}$
$\bar{v}_3 \frac{\partial v_3}{\partial v_0}$

Figure 3.13: A computational graph representing the function $f(x_1, x_2) = \ln(x_1) + x_1 \cdot x_2 - \sin(x_2)$. All intermediate values, v_i, as well as their Adjoints, $\bar{v}_i$, follow the notation of Ref. [131] and denote the intermediate derivative with respect to the given initialisation. Note that two graphs are merged in one, like with the use of Dual numbers for forward mode AD except one graph goes input to output, and the other goes output to input. This figure is modified from Ref. [129].

mode, the notation $\bar{v}_i$ is used to represent the adjoint derivative, or derivative of the output with respect to the parameter, v_i, i.e. $\bar{v}_i = \frac{\partial y}{\partial v_i}$. The node $\bar{v}_5$ is initialised to unity as we're differentiating the output with respect to all nodes. Mathematically this emerges from differentiating v_5 with respect to v_5 which is equal to unity by definition. The connected nodes to v_5 are v_4 and v_3. The derivatives of these nodes can be found by multiplying the adjoint of $\bar{v}_5$ by the derivative between v_5 and the respective node, e.g. v_4. So, the derivative of v_4 equals,

$$\frac{\partial y}{\partial v_4} = \underbrace{\frac{\partial y}{\partial v_5}}_{adjoint} \cdot \frac{\partial v_5}{\partial v_4} \tag{3.52}$$

For the current computational graph this can defined for all nodes as follows,

$$\bar{v}_5 = 1 \tag{3.53}$$

$$\bar{v}_4 = \bar{v}_5 \frac{\partial v_5}{\partial v_4} \tag{3.54}$$

$$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_4} \tag{3.55}$$

$$\bar{v}_2 = \bar{v}_4 \frac{\partial v_4}{\partial v_2} \tag{3.56}$$

$$\bar{v}_1 = \bar{v}_4 \frac{\partial v_4}{\partial v_1} \tag{3.57}$$

$$\bar{v}_0 = \bar{v}_2 \frac{\partial v_2}{\partial v_2} + \bar{v}_3 \frac{\partial v_3}{\partial v_0} \tag{3.58}$$

$$\bar{v}_{-1} = \bar{v}_1 \frac{\partial v_1}{\partial v_{-1}} + \bar{v}_2 \frac{\partial v_2}{\partial v_{-1}} \tag{3.59}$$

Eqs. 3.53 through 3.59 denote a hierarchical structure of propagating adjoints from the output node to all input nodes in one pass. By using the chain rule, we can multiply all adjoints together to calculate the exact derivative [2].

3.6.4 Forward-mode, Reverse-mode, or both?

When comparing forward-mode AD vs reverse-mode AD it helps to see how both algorithms compare for an arbitrary function, $\mathbb{R}^n \mapsto \mathbb{R}^m$. The Jacobian of a given function is a matrix of partial derivatives between all output nodes and all input nodes. For a $\mathbb{R}^n \mapsto \mathbb{R}^m$ function, its Jacobian, $\mathcal{J}_f$, would be a $m \times n$ sized matrix,

$$\mathcal{J}_f = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} \cdots \frac{\partial y_1}{\partial x_n} \\ \vdots \ddots \vdots \\ \frac{\partial y_m}{\partial x_1} \cdots \frac{\partial y_m}{\partial x_n} \end{bmatrix} \tag{3.60}$$

Using forward-mode AD we construct the Jacobian 1 column at a time by setting an input node to unity, with the remaining input nodes to zero, so as to effectively create a unit vector. For reverse-mode AD we construct the Jacobian 1 *row* at a time by setting an output node to unity, with the remaining output nodes to zero. This creates a unit vector acting in the opposite direction. This results in the walltime for calculating the *entire* Jacobian matrix being dependent on which AD mode is used. The walltime forward-mode AD it's proportional to $n \cdot \mathrm{ops(f)}$ and for reverse-mode AD it's proportional to $m \cdot \mathrm{ops(f)}$. Therefore, for $n \ll m$ forward-mode AD will incur fewer operations than the reverse-mode AD approach. Likewise for $m \ll n$ reverse-mode AD will hold fewer operations than forward-mode AD.

[2] subject to floating point arithmetic

Also, initialising the tangent/adjoints to an arbitrary vector allows one to construct Jacobian-vector or vector-Jacobian products in a matrix-free manner, i.e. without constructing the full matrix and storing it in memory [129]. A key advantage of forward-mode AD is calculating the function evaluation with its directional derivative simultaneously. For reverse-mode AD, its key advantage is computing the derivative for all input nodes simultaneously for a scalar output which is the most common output shape for most loss functions. However, this does require more memory to store the intermediate variables as they are not computed on the fly like with forward-mode AD.

In this thesis this discussion holds relevance with the calculation of the kinetic energy of a wavefunction,

$$\langle \hat{T} \rangle = \mathbb{E}_{x\sim|\Psi|^2}\left[\frac{1}{\Psi}\frac{\partial^2\Psi}{\partial x_i^2}\right]. \tag{3.61}$$

This is the trace of the Hessian of the wavefunction with respect to its input and requires that the wavefunction is differentiated twice. The Hessian could be computed by using forward-mode AD twice (forward-over-forward) or reverse-mode AD twice (reverse-over-reverse). However, it was shown by Ref. [134] that it is actually more efficient to merge forward-mode AD and reverse-mode AD together into forward-over-reverse AD. This can be done by computing the directional derivative for each input node and then computing the reverse-mode derivative for the respective node. This process can then be vectorised to efficiently compute the Laplacian (the trace of the Hessian) without directly computing all Hessian elements, and hence is a matrix-free way, in principle, of computing the Laplacian. As we have discussed how to efficiently calculate exact gradients of a given function. We will now turn to how to optimise a set of variational parameters via the use of gradient-descent based algorithms.

3.7 A Theoretical Perspective on Convergence

Let's assume we seek to find the minimum of some arbitrary one-dimensional function which depends on a single weight, θ, where the corresponding loss is arbitrarily defined as $\mathcal{L}(\theta)$. An example could be linear regression for a single weight. In this pedagogical example it will be assumed that the cost function is purely quadratic and the loss landscape in vicinity of the global minima is shown in Fig. 3.14.

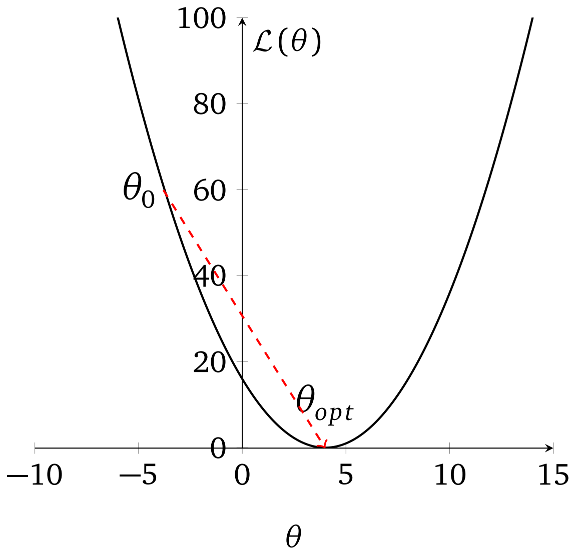


Figure 3.14: The loss, $\mathcal{L}(\theta)$, landscape of a purely quadratic cost function with a global minimum at $\theta_{opt} = 4$. The dashed red arrow denotes the update of the initial weight, θ_0, towards the global minimum in a single update.

In this example the optimal weight is chosen to be θ_{opt} equals 4 which corresponds to a loss value of zero, i.e. $\mathcal{L}(\theta = 4) = 0$. In the gradient descent setting we randomly initialise our weight(s) in accordance to some probability distribution which has the same dimensionality as our weights. The initial weight, θ_0, will have a corresponding loss value greater than the global minima. In the optimal case, we would use the gradient information of the current loss to update our weight value such that the weight sits at the global minimum, i.e. $\theta_0 \mapsto \theta_{opt}$, as shown in Fig. 3.14. The optimal step towards the global minimum is directly linked to the Hessian of the loss which incorporates the curvature of the loss landscape. This can be seen by taking a Taylor series expansion of the loss around the initial point, θ_0,

$$\mathcal{L}(\theta) = \mathcal{L}(\theta_0) + (\theta - \theta_0)\frac{\partial \mathcal{L}(\theta_0)}{\partial \theta} + \frac{1}{2}(\theta - \theta_0)^2\frac{\partial^2 \mathcal{L}(\theta_0)}{\partial \theta^2}, \tag{3.62}$$

and differentiating the Taylor series with respect to θ we find,

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta} = \frac{\partial \mathcal{L}(\theta_0)}{\partial \theta} + (\theta - \theta_0)\frac{\partial^2 \mathcal{L}(\theta_0)}{\partial \theta^2}. \tag{3.63}$$

By defining θ as θ_{opt}, Eq. 3.63 can be rearranged to highlight how the Hessian directly relates to the optimal step-size,

$$0 = \frac{\partial \mathcal{L}(\theta_0)}{\partial \theta} + \left(\theta_{opt} - \theta_0\right) \frac{\partial^2 \mathcal{L}(\theta_t)}{\partial \theta^2} \tag{3.64}$$

$$\theta_{opt} = \theta_0 - \underbrace{\left(\frac{\partial^2 \mathcal{L}(\theta_t)}{\partial \theta^2}\right)^{-1}}_{\text{optimal step-size}} \frac{\partial \mathcal{L}(\theta_0)}{\partial \theta} \tag{3.65}$$

The optimal step is equal to the inverse of the Hessian. This will update the initial weight, θ_0, to the optimal weight, θ_{opt}, in one step. In the case of more than one parameter, the inverse of the Hessian scales quadratically in the number of matrix elements. However, the associated matrix inversion brings additional factors and the total number of operations is cubic in the number of matrix elements. If the eigendirections of the Hessian could be aligned with the co-ordinate axes of the parameter space, the weights within the Hessian would be decoupled[3]. This would effectively convert an N-dimensional parameter space to a set of N one-dimensional parameter spaces. In practice, the Hessian's eigendirections never aligns with its co-ordinate axes indicating that the weights are coupled. The weights can be decoupled via diagonalising the Hessian. One way of quantifying the importance of each direction in the parameter space is to calculate the eigenvalues of the Hessian which physically represents the curvature along a given eigendirection. Let us take a Taylor series approximation around the global minimum, θ_{opt},

$$\mathcal{L}(\theta) = \mathcal{L}\left(\theta_{opt}\right) + \frac{1}{2}\left(\theta - \theta_{opt}\right)^2 \frac{\partial^2 \mathcal{L}\left(\theta_{opt}\right)}{\partial \theta^2} \tag{3.66}$$

which, when differentiating with respect to the weight, θ, equals,

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta} = \left(\theta - \theta_{opt}\right) \frac{\partial^2 \mathcal{L}\left(\theta_{opt}\right)}{\partial \theta^2}. \tag{3.67}$$

Therefore, we can define an update under gradient descent as,

$$\theta_{t+1} = \theta_t - \alpha\left(\theta - \theta_{opt}\right) \frac{\partial^2 \mathcal{L}\left(\theta_{opt}\right)}{\partial \theta^2}. \tag{3.68}$$

By subtracting θ_{opt} from both side of Eq. 3.68, and factorising we find,

$$\underbrace{\theta_{t+1} - \theta_{opt}}_{\substack{\text{Distance from local}\\\text{minima at time, t+1}}} = \underbrace{\left(\mathcal{I} - \alpha \frac{\partial^2 \mathcal{L}\left(\theta_{opt}\right)}{\partial \theta^2}\right)}_{\substack{\text{Eigenvalues' magnitude are}\\\text{less than unity}}} \underbrace{\left(\theta_t - \theta_{opt}\right)}_{\substack{\text{Distance from local}\\\text{minima at time, t}}} . \tag{3.69}$$

As we shall see below, in order for an update to converge the eigenvalues must be less than unity which corresponds to reducing the magnitude of the input vector, $\left(\theta_t - \theta_{opt}\right)$

[3] An eigendirection is defined as the space of scalar multiples of a given eigenvector.

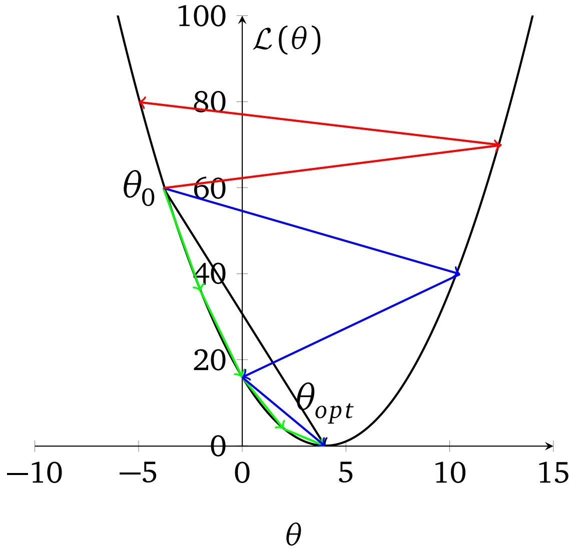


Figure 3.15: Shows the convergence behaviour dependent on the size of the learning rate (step-size) in reference to the optimal learning rate, α_{opt}. With black ($\alpha = \alpha_{opt}$), green ($\alpha \approx 0$), blue ($\alpha_{opt} < \alpha < 2\alpha_{opt}$), and red ($\alpha > 2\alpha_{opt}$) representing the different regions of behaviour.

to the output vector, $\left(\theta_t - \theta_{opt}\right)$. This will eventually reduce to a vector of zero magnitude which indicates that the parameters have converged, i.e. $\theta_{t+1} = \theta_t = \theta_{opt}$.

In the one-dimensional case, or the multidimensional case wherein the Hessian is aligned with its eigendirections, Eq. 3.69 links the condition of convergence to the eigenvalues of the Hessian itself. It will converge under the condition $|1-\alpha\lambda_i| < 1$ where λ_i is the i^{th} eigenvalue of the Hessian and α is the step-size or learning rate. This relation gives us an acceptable range for α, and different regions of α will exhibit different behaviour. Expanding out $|1-\alpha\lambda_i| < 1$ results in,

$$0 < \alpha < \frac{2}{\lambda_i}. \tag{3.70}$$

For $\alpha \approx 0$ the convergence towards the nearest local minima will take an impractical amount of updates. For $\alpha = \alpha_{opt}$, the optimal step-size, it will converge to the local minimum in a single update and is directly related to the reciprocal of the corresponding eigenvalue, i.e. $\alpha_{opt} = \frac{1}{\lambda_i}$. For a value of $\alpha_{opt} < \alpha < 2\alpha_{opt}$, convergence towards the local minimum is guaranteed albeit with oscillations. In the case of $\alpha > 2\alpha_{opt}$, convergence is not guaranteed and in fact it will diverge. This can be visualised in Fig. 3.15. In fact the optional learning rate is the reciprocal of the eigenvalue, $\frac{1}{\lambda_i}$. This shows mathematically that one should not use a global learning rate when minimising a set of variational parameters but should assign each weight its own learning rate which ideally should equal the reciprocal of its corresponding eigenvalue. However, doing this exactly is impractical and other methods need to be used that can efficiently estimate the optimal learning rate. In the multi-dimensional case, the eigenvalues of the Hessian are impossible to calculate for models with a large number of parameters due to the it scaling as $\mathcal{O}\left(N^2\right)$ with its inversion scaling as $\mathcal{O}\left(N^3\right)$. If this were possible, the optimal learning rate would lead the model to converge in only a few steps [135].

3.8 First-Order Numerical Optimisation

3.8.1 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is an optimisation algorithm which stochastically computes estimates of the gradient of the loss and uses that information to minimise the loss. This is done by iteratively updating the parameters of a DNN while using the aforementioned gradient,

$$\theta_{t+1} = \theta_t - \alpha \frac{\partial \mathcal{L}}{\partial \theta_t}, \tag{3.71}$$

where $\mathcal{L}$ is the loss function we seek to minimise, θ is the set of parameters at a given time-step, t, with learning rate, α. This process is repeated for a fixed number iterations, or *epochs* in the ML community, until $\theta_{t+1} \approx \theta_t$ which indicates that the weights are in a local minimum. As mentioned in Sec. 3.7, the optimal learning rate is related to the eigenvalues of the Hessian of the loss function. In practice utilising Hessian information is intractable due to its scaling. An example of this is the pioneering Convolutional Neural Network (CNN) AlexNet which contains over 60 million parameters and significantly improved the state-of-the-art (SOTA) performance on the ImageNet database [12]. The Hessian for a network would contain contain $\sim 10^{15}$ matrix elements, and cost on the order of 10^{23} operations which is impractical with current hardware.

This motivates moving away from first-order methods like SGD to 'higher'-order methods which precondition the gradient to further improve convergence. The most notable of such schemes are diagonal approximations which deal solely with the diagonal of the Hessian and therefore scale linearly with the number of parameters. The two most common optimisers that use only the diagonal of the Hessian are Root-Mean-Square propagation (RMSprop) [17], and Adaptive Moment Estimation (Adam) [19]. Other approximations can expand upon the diagonal approximation to include block-diagonal information which exploits the inherent structure of a DNN in order to efficiently approximate the Hessian. A clear example of this is Kronecker-Factored Approximate Curvature, KFAC, which will be discussed in detail with novel expansion in Appendix B.

3.8.2 Root-Mean-Squared Optimization

One way to introduce higher-order effects without significantly introducing any overhead is to record the history of the gradient and use that to speed-up convergence if the variational parameters continually move in the same direction. This optimisation algorithm has one additional step before updating the variational parameters. An exponential moving average (EMA) is used to compute a 'velocity' of each parameter,

$$\mathcal{V}_{t+1} = \beta \mathcal{V}_t + (1-\beta)\left(\frac{\partial \mathcal{L}}{\partial \theta_t}\right)^2 \tag{3.72}$$

where $\beta \in [0,1)$ is a 'hyperparameter' which dictates the importance of the history of the square of the gradient[4]. The physical interpretation of this is that smaller values of β focus more on the current gradient, and higher values focus more on the history of recent gradients. The reason for the use of the EMA is to efficiently store the history without having to allocate memory to store the n most recent gradients, by accumulating these values and updating them in-place can allow for all the history to be stored as a single variable thereby being much more efficient. Once the 'velocity' has been calculated we can include it with the standard definition of SGD,

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\mathcal{V}_t} + \epsilon} \frac{\partial \mathcal{L}}{\partial \theta_t} \tag{3.73}$$

The use of dividing the learning rate by a prefactor of '$\sqrt{\mathcal{V}_t} + \epsilon$' allows for the learning rate to be 'adapted' locally for each variational parameter. The velocity is initialised to zero at the initial time-step, i.e. $\mathcal{V}_t = 0$, and ϵ is a numerical stability constant used to avoid a divide-by-zero error which has a default value of 10^{-8}. It should be noted that PyTorch's RMSprop implementation differs from Tensorflow's implementation by excluding the ϵ term from the square root whereas TensorFlow includes ϵ within the square root [136].

If the velocity is large this indicates that the history of gradients are in a steep-region of the loss landscape, and therefore one should use a small learning rate to stop overshooting a local minima. In the other case, if the velocity is small then this indicates that the parameter is in a flat-region of the loss landscape and by including previous gradient we can add an 'inertia' which helps convergence across relatively flat regions of the loss landscape.

3.8.3 Adaptive Moment Estimation

The EMA approach of RMSprop allows for significant speed-up by including an approximation to curvature information to regulate the learning rate of our optimisation algorithm.

[4] A hyperparameter is a parameter that is not adapted by the optimisation scheme but determines its behaviour, e.g. the learning rate is also a hyperparameter

Further information can also be included with the first-moment of the gradient, not only the second-moment like with RMSprop. Adam, or Adaptive Moment Estimation, is a first-order stochastic gradient descent algorithm which efficiently uses EMAs to adapt the learning rate for each variational parameter via the use of the first-moment, $\mathcal{M}_t$, and second-moment, $\mathcal{V}_t$, of the gradients. These moments are stored efficiently as EMAs (as discussed previously in Sec. 3.8.2) and have their own associated hyperparameters β_1 and β_2 which control the importance of previous gradients,

$$\mathcal{M}_t = \beta_1 \mathcal{M}_{t-1} + (1-\beta_1)\frac{\partial \mathcal{L}}{\partial \theta_t}, \tag{3.74}$$

$$\mathcal{V}_t = \beta_2 \mathcal{V}_{t-1} + (1-\beta_2)\left(\frac{\partial \mathcal{L}}{\partial \theta_t}\right)^2. \tag{3.75}$$

The true value for the first- and second-moments of the gradient is not equal to zero, and hence initialising them as zero biases any future estimates as they are calculated via EMAs. The explicit form of the moments for a given number of time-steps, t, are defined as,

$$\mathcal{M}_t = (1-\beta_1)\sum_{i=1}^{t} \beta_1^{t-i}\frac{\partial \mathcal{L}}{\partial \theta_t}, \tag{3.76}$$

$$\mathcal{V}_t = (1-\beta_2)\sum_{i=1}^{t} \beta_2^{t-i}\left(\frac{\partial \mathcal{L}}{\partial \theta_t}\right)^2. \tag{3.77}$$

The use of an EMA for the first- and second-moments of the gradients allows the gradient to be preconditioned via incorporating a form of gradient inertia from the first-moment EMA and an estimate of curvature from the second-moment EMA. The moments of Eqs. 3.74 and 3.75, although numerical efficient to calculate, will be zero-biased. This can be found by taking the expectation of Eq. 3.74 (Eq. 3.75) and comparing with $\frac{\partial \mathcal{L}}{\partial \theta_t}$ ($\frac{\partial \mathcal{L}}{\partial \theta_t}$). If both terms equal each other then the expected value of the EMA will equal the *true* moment at the t epoch. However, this isn't the case as the factor will be biased by $\left(1-\beta_1^t\right)$ ($\left(1-\beta_2^t\right)$). Therefore, by dividing Eq. 3.74 by $\left(1-\beta_1^t\right)$ and Eq. 3.75 by $\left(1-\beta_2^t\right)$ will remove the bias of such EMAs,

$$\mathcal{M}_t^{cor} = \frac{\mathcal{M}_t}{\left(1-\beta_1^t\right)} \tag{3.78}$$

$$\mathcal{V}_t^{cor} = \frac{\mathcal{V}_t}{\left(1-\beta_2^t\right)}. \tag{3.79}$$

This bias-correction is not performed with RMSprop, but is performed with Adam, often leading to improvements. Once the corrected moments have been estimated, we can follow a similar method to RMSprop and modify SGD to take into account the first- and second-moment of our gradient via,

$$\theta_{t+1} = \theta_t - \alpha\frac{\mathcal{M}_t^{cor}}{\sqrt{\mathcal{V}_t^{cor}} + \epsilon}. \tag{3.80}$$

In the work of this thesis, both RMSprop and Adam will be used in solving for the ground-state wavefunction of given Hamiltonians. In chapter 4, RMSprop will used to solve for the ground-state of the deuteron, and in chapters 5 and 6 Adam will be used to solve for a many-body system of A one-dimensional fermions.

Chapter 4

The Deuteron

This Chapter of the thesis details the first of two fermionic schrödinger equations solved within this work. This Chapter is a merger of previous work detailed within the Confirmation report and an extension of the Author's own work that was published within Ref. [137]. The work entails solving the deuteron as an effective one-body problem and comparing with exact diagonalisation [4].

4.1 Ansatz

In this thesis we uses a shallow neural network to represent both the S- and D-state wavefunctions for the deuteron under the Entem-Machleidt N^{3}LO interaction [4, 137]. The matrix elements of this interaction are easily available in a partial wave basis in momentum space, so we choose to solve the problem in this basis for convenience. This leads to the problem being solved within momentum space. The kinetic energy becomes a continuous function thereby avoiding costly second derivatives calculations. In addition the centre-of-mass and relative motion contributions can be separated allowing for the problem to be effectively solved with relative momentum, q. By combining all these aspects the deuteron can be solved as an effective one-body problem with two coupled angular momentum states.

The N^{3}LO interaction realistically represents the deuteron with both S- and D-state contributions and therefore, when solving for the groundstate, we need to resolve two state wavefunctions simultaneously. This motivates the construction of an $\mathbb{R}^1 \mapsto \mathbb{R}^2$ function that can predict the wavefunction contributions for both states of the deuteron simultaneously for a given value of relative momenta, q.

The main focus of this thesis was to design a network which was minimalist in approach and both states of the deuteron are represented by the same internal hidden state which can be interpreted as compressing the information of both state wavefunctions within a single embedding vector [137]. The network used was a shallow, single layered, feed-forward neural network as can be seen within Fig. 4.1. The input node, q, represents the input of a given value of the relative momentum of the deuteron, and the output nodes, S and D, represent the partial wavefunctions for the deuteron for that value of q. Mathematically, the network is equivalent to,

$$\psi^{L}_{VANN}(q) = \sum_{i=1}^{H} \mathcal{W}^{(2)}_{i,L}\sigma\left(\mathcal{W}^{(1)}_{i}q + b^{(1)}_{i}\right) \tag{4.1}$$

where $\mathcal{W}^{(l)}$ and $b^{(l)}$ denote the weights and bias of the l-th layer respectively. These weights connect nodes of different layers together via an linear transformation, or affine if a bias node is used. Connections with the superscript (1) denote weights/biases that connect q to the hidden nodes, and (2) connects the hidden nodes to the output nodes which physically represent the amplitude of the S- and D-state respectively. The term, $\sigma(x)$, represents the non-linear activation function used to inject non-linearity, and is applied element-wise along each node of the hidden layer. Within this work, two activation functions were used; Sigmoid, and Softplus. The motivation was to use a bounded (Sigmoid) and unbounded (Softplus) activation function and compare how the boundedness of the activation function dictates network performance. There is no activation function applied to the connection between the hidden layer and the output nodes, and therefore each partial wavefunction is a weighted-linear combination of the same hidden nodes. The network initially holds an arbitrary admixture between both states. The total set of weights (and biases) is denoted by θ which is the concatenation of all connections, $\theta = \{\mathcal{W}^{(1)}, b^{(1)}, \mathcal{W}^{(2)}\}$. For the given architecture, the number of weights (and biases) is four times the number of hidden nodes, $4H$. Where each node has 2 parameters for the initial affine transformation of the relative momenta, and 1 parameter for each state wavefunction.

The weights of these connections are initialised with uniform distributions; $\mathcal{W}^{(1)} \in [-1, 0)$, $b^{(1)} \in [-1, 1)$, and $\mathcal{W}^{(2)} \in [0, 1)$. This differs from traditional initialisation scheme like Xavier's or He's initialisation [138, 107] which are symmetrically distributed via a uniform or Gaussian distribution. These initialisation schemes were chosen to dictate the initial shape of the starting wavefunction.

In this work, we use a Gauss-Legendre quadrature in all integrations that are required due to its high accuracy and the low dimensionality of the physical system at play. The relative momenta space, q, is discretised between $q_1 = 0$ fm^{-1} and $q_N = 500$ fm^{-1} for $N = 64$ points. The spacing of these points is non-equidistant and is governed by a tangential map applied to the initial spacing of the mesh under the Gauss-Legendre method. The computational implementation thereof is conducted via the SciPy library's `roots_legendre` function [139].

The N discretised values of momentum are calculated by solving for the roots of the n^{th} ordered Legendre Polynomial, $P_n(k')$ which holds roots in $[-1, 1]$. These values are

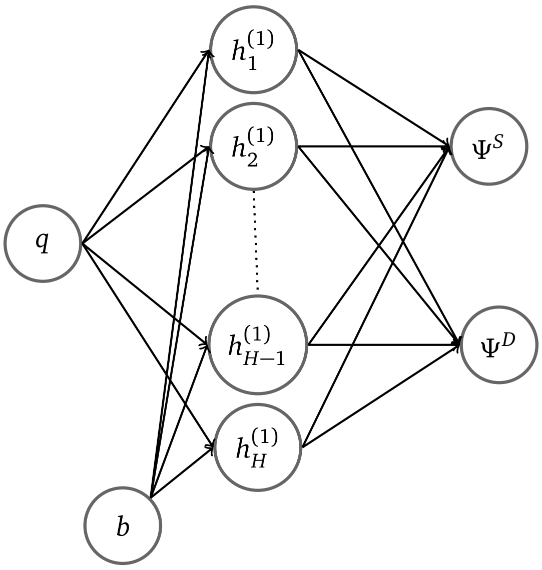


Figure 4.1: The Neural Network ansatz of Ref. [137] for a neural network with a given number of hidden nodes, H. The network maps the relative momenta, q, to the S-state and D-state wavefunction via an intermediate hidden state of H nodes with each node being an affine transformation with bias, b. The activation function is explicitly assumed to occur within the hidden nodes, and is omitted due for brevity.

re-scaled by a linear transformation and subsequent tangential map to update the initial values, $k' \in [-1, 1]$ to $k \in [0, 500]$ via,

$$k_n = c \tan\left(\frac{\pi}{2} k'_n\right), \tag{4.2}$$

where c is the cut-off parameter defined by,

$$c = \frac{k_{max}}{\tan\left(\frac{\pi}{2} k_N\right)}, \tag{4.3}$$

where k_{max} is chosen to be 500 fm^{-1}. The integrals are calculated via Gauss-Legendre quadrature method,

$$\int_0^\infty f(k) dk = \sum_{i=1}^{N} \omega_i f(k_i), \tag{4.4}$$

where ω_i denotes the weights of the quadrature method. The original weights, ω', are derived from the Legendre polynomials such that they allow the integration of polynomials of order 2n-1 over the interval, $[-1, 1]$ where n is the number of samples [140]. In order to integrate over the domain of $[0, 500]$ fm^{-1}, the weights are transformed via a tangential map to,

$$\omega_i = \frac{\omega_i' c\pi}{2\cos\left(\frac{\pi}{2}k_i\right)}. \tag{4.5}$$

with ω denoting the re-scaled roots of the Legendre polynomials.

4.2 Methodology

The methodology of Ref. [137] is separated into three sections; initialisation, pre-training, and energy minimisation. The initialisation scheme will be omitted as it was previously discussed in Sec. 4.1, however, the pre-training and the energy minimisation will be discussed in full.

4.2.1 Pre-training

The network is initialised by a uniform distribution, as stated in Sec. 4.1, which sets the initial shape of each state of the deuteron. We follow the methodology of Ref. [2] by pre-training our network to a physically sound target wavefunction rather than the initially random wavefunction. The target wavefunction has the following functional form,

$$\psi^L_{targ}(q) = q^L \exp\left(-\frac{\xi^2 q^2}{2}\right), \tag{4.6}$$

where $\xi = 1.5$ fm is an arbitrary parameter that dictates the shape of the target wavefunction. The q^L pre-factor enforces origin behaviour for different states; $\psi^S(0) \neq 0$ and $\psi^D(0) = 0$. The overlap between two arbitrary wavefunctions is defined by,

$$\begin{aligned} \mathcal{K}^L &= \frac{\langle \psi^L_{targ} | \psi^L_{VANN} \rangle^2}{\langle \psi^L_{targ} | \psi^L_{targ} \rangle \langle \psi^L_{VANN} | \psi^L_{VANN} \rangle}, && (4.7) \\ &= \frac{\left[\int_0^\infty dq q^2 \psi^L_{targ}(q) \psi^L_{VANN}(q)\right]^2}{\int_0^\infty dq q^2 \psi^L_{targ}(q) \psi^L_{targ}(q) \int_0^\infty \psi^L_{VANN}(q) \psi^L_{VANN}(q)}. && (4.8) \end{aligned}$$

A pre-training cost function is the sum of the difference in overlap with the exact result for both states, i.e. $\mathcal{C} = \mathcal{C}^S + \mathcal{C}^D$ where,

$$C^L = \left(\mathcal{K}^L - 1\right)^2, \tag{4.9}$$

and is minimised with use of the RMSprop optimisation algorithm with the hyperparameters, $\alpha = 10^{-2}$, $\beta = 0.9$, and $\epsilon = 10^{-8}$. The large learning rate, α, is possible due to the fact that the normalisation constant of each partial wave is larger than unity and hence acts as regulator to the learning rate [137]. Once the pre-training has finished, all neural networks obtain a pre-training overlap of 0.95 to 0.99 independent of the number of hidden nodes with the desired value of $\mathcal{K}^L$ equal to 1. The unphysical admixture of 50% between the S- and D-state does not impact the pre-training as the current model has no knowledge of the physical system.

4.2.2 Energy Minimisation

Once the network has been pre-trained it is able to begin minimising the energy directly towards the ground-state. This can be done via computing an expectation value of the energy under the N^{3}LO interaction and directly differentiating it with respect to the weight of the network, $\mathcal{W}$. This differentiation is performed automatically by PyTorch's `autograd` engine [130]. The expectation value of the energy is defined as,

$$\langle E \rangle = \frac{\langle \Psi_{VANN} | \hat{H} | \Psi_{VANN} \rangle}{\langle \Psi_{VANN} | \Psi_{VANN} \rangle} = \langle T^S \rangle + \langle T^D \rangle + \langle V^S \rangle + \langle V^D \rangle + \langle V^{SD} \rangle + \langle V^{DS} \rangle + \langle V^D \rangle \quad (4.10)$$

where $\langle T^L \rangle$ and $\langle V^L \rangle$ is the kinetic energy and potential energy for state L respectively. The superscript of SD denotes the admixture between the S- and D-state wavefunctions due to the tensor component of the nuclear force [141]. The convergence of the energy isn't impacted by the initially large admixture of 50%. For the first 10,000 iterations, the network shows a rapid binding energy convergence as shown in Fig. 4.2. Past 10,000 iterations fluctuations emerge in the convergence, although, the overall convergence is unhindered. Within 50,000 iterations the binding energy is within 5% of the exact groundstate. The exact groundstate is computed by direct diagonalisation on the same numerical mesh. Finally, we conclude the convergence at 250,000 iterations where the binding energy of the deuteron has fluctuations of the order of 2-3 keV. Once the network has finished minimising at 250,000 iterations, we can reconstruct both partial wavefunctions by performing a forward pass through the network. The state wavefunctions for a network of $H = 10$ are shown in Fig. 4.3. The performance of the network holds excellent agreement with the exact ground-state wavefunction (solid red line) for a wide range of momenta. The exact ground-state wavefunction is found by numerical diagonalisation in momentum space with the same mesh structure. The only discrepancy is for low-momenta values wherein a clear mismatch between the network's prediction and the exact ground-state wavefunctions can be clearly seen. In Fig. 4.3, the results of the post-minimisation are shown with both Sigmoid (dashed blue line), Softplus (dotted green line), and the exact result (solid red line). There is excellent agreement between the network result and the exact results for both states regardless of the activation function. In fact, these results are repeated for 50 different pseudorandom seed initialisation and the standard deviation is shown as an errorband around the prediction to indicate initialisation effects on the convergence of the network. This errorband is negligible

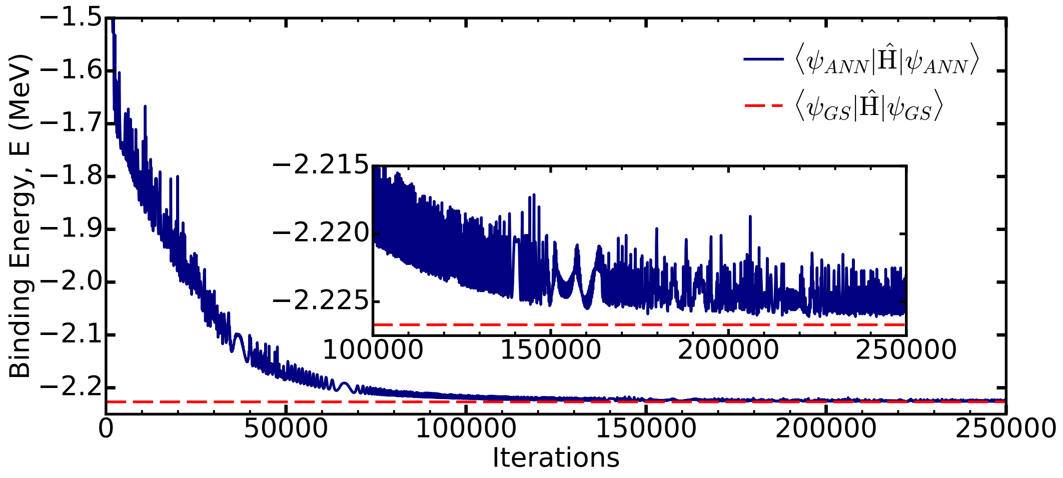


Figure 4.2: The binding energy of the deuteron as a function of iteration number for a network with $H = 10$ and a Softplus activation function.

within the figure which indicates a robust methodology for solving Schrödinger's equation with a shallow neural network.

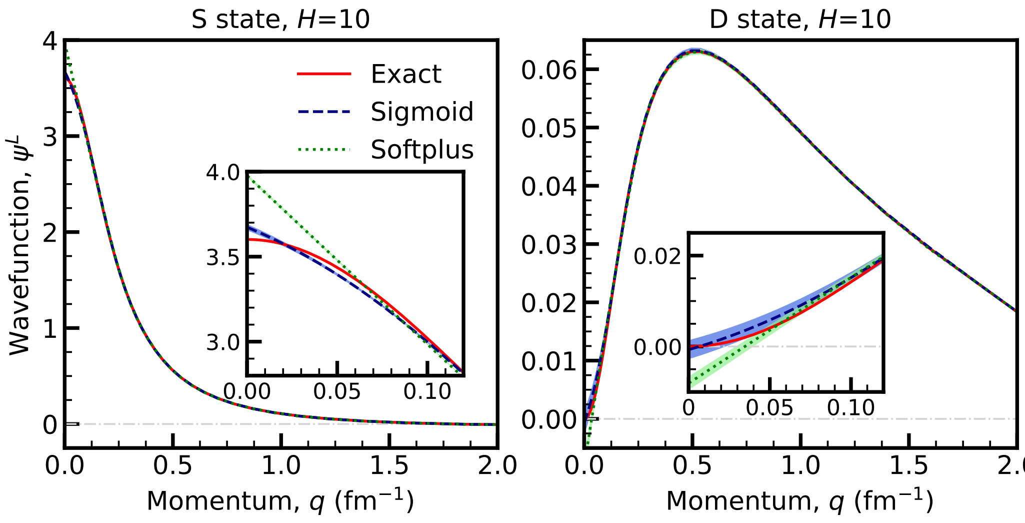


Figure 4.3: Left (right) panel: the S (D) state wavefunction as a function of momentum, q. The exact wavefunction (red) is compared against the neural network ansatz of 10 hidden nodes with sigmoid (blue) and Softplus (green) activation function.

4.3 Results

The results of this network are broken down into the following subsections. Firstly, properties of the deuteron are measured and compared against direct diagonalisation in order to determine the accuracy of this methodology. These properties are benchmarked against the exact results and the network's performance is varied twofold; firstly with the number of hidden nodes and activation function, and secondly with the initial seed configuration in order to quantify weight initialisation effects.

4.3.1 Binding Energy and Fidelities

A summary of all the results obtained for the deuteron can be seen in Fig. 4.4. The first row contains the results of the binding energy of the deuteron as a function of the number of hidden nodes from 2 all the way up to 100 hidden nodes. The central value is the mean of final binding energies obtain over all 50 seed configurations, and the error-band is the associated standard deviation. For $H = 2$ the results are not shown for brevity and clarity. The deuteron is bound by 0.8MeV although with significant variance. This does improve with the $H = 4$ case where the binding energy is within 5% of the binding energy benchmark and 2% of the fidelity. As the width of the network increases all three properties improve. For $H = 6$, the network can achieve a binding energy with 10keV of the exact binding energy and a fidelity of 99.99%. In Fig. 4.4, both

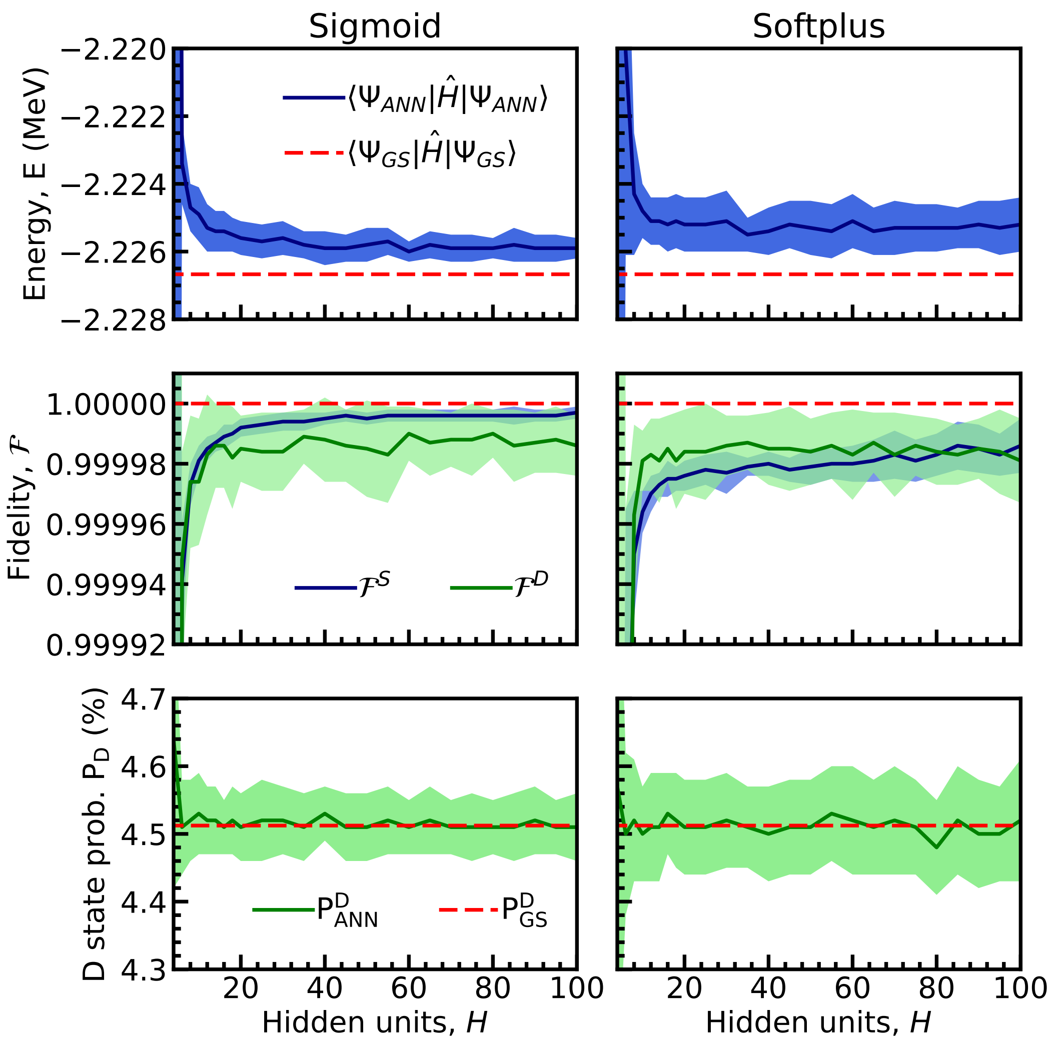


Figure 4.4: The results of the neural ansatz for the binding energy (top row), fidelities to exact ground-state wavefunction (middle row), and D-state probability (bottom row). The results of exact diagonalisation are denoted in red (dashed) with S-state wavefunction (blue) and D-state wavefunction (green). The left (right) column being the neural network ansatz with a Sigmoid (Softplus) activation function. The errorband represents the standard deviation of 50 measurements with different pseudorandom initialisation and shows how initialisation affects the performance of the network in determining the ground-state wavefunction [137].

activations function are shown with Sigmoid (Softplus) on the left (right). The binding energy results between both activation function performs well with sigmoid performing slightly better in not only its bias (from the exact ground-state) but also its variance. This reduction in variance will be directly related to the output range of the activation function. Sigmoid is limited to $(0, 1)$ rather than Softplus which is limited to $(0, \infty)$, and hence the intermediate values of the hidden nodes will have more variance which will propagate through to the output nodes for each state wavefunction. The superior performance of Sigmoid over Softplus seems to initially contradict the vanishing gradient problem by which the Sigmoid activation function is more affected. However, this issue primarily affects DNNs of 10s to 100s of layers so this behaviour is not untoward. As the network used in this chapter is a shallow network this issue has negligible effect on performance [80].

The fidelity, $\mathcal{F}^L$, of the state wavefunction, L, is denoted by the overlap between the network wavefunction, $|\Psi^L_{VANN}\rangle$, and the *exact* wavefunction, $|\Psi^L_{gs}\rangle$. This can be related by Eq. 4.8 except by replacing $|\Psi^L_{targ}\rangle$ with $|\Psi^L_{gs}\rangle$. An exact replication of the exact ground-state wavefunction would be defined by $\mathcal{F}^L$ equal to unity. When measuring the fidelities of the network against unity we see excellent performance for the fidelities. The middle row of Fig. 4.4 shows the fidelities for the both state wavefunctions as a function of the number of hidden nodes. In fact a more stark contrast can be seen here, firstly comparing the S-state wavefunction between sigmoid and Softplus activation functions shows a clear superiority for the sigmoid case. This improvement is, again, not only limited to its bias but also its variance. The variance is negligible towards the higher limit of hidden nodes and the fidelity approaches a value of over 99.999%. The Softplus case is more impacted by an increase in nodes, as can be seen in the subfigure, however, the bias still reduces; due to the UAT [86]. However, it can be clearly seen that the variance of the results increases too; potentially hinting at the bias-variance trade-off starting to impact convergence [76]. Regardless, the results for the S-state wavefunction are within excellent agreement of the exact diagonalisation results. The results of the D-state wavefunction fidelities hold a higher variance than the S-state wavefunction, regardless of activation function, with this increase of variance being consistent for both activation functions. This is most likely due to two turning points with the structure of the D-state wavefunction; the origin $q = 0$ fm^{-1}, and the maximum at $q \approx 0.5$ fm^{-1}. Within Fig. 4.3 it can be seen that both activation functions overshoot the low-momenta components of the D-state wavefunction subject to the non-linearity of the activation function but it can also be seen that there's noticeable variance at the maximum of $q \approx 0.5$ fm^{-1} which bulges at the turning point. This bulge in the prediction will increase the variance for all properties discussed as the network struggles to exactly predict the turning point. However the high momenta component of the wavefunction is an exponential decay over 5 points within the mesh, and is less prone to higher variance predictions.

4.3.2 D-state Probability

The final row within Fig. 4.4 contains the D-state probability across activation functions and number of hidden nodes. As previously mentioned, there is a direct correlation between the D-state probability and the strength of the Tensor force [141]. Within just 6 hidden nodes, both network structure reconstruct the exact value of P_D within 0.01%. The fact the central value is consistent across activation functions for the same number of hidden nodes indicates that a kind of independence of activation function for such property of the deuteron exists and the current methodology is robust. Although the central value remains constant as the number of hidden nodes increases, the variance does not. In fact, by increasing the number of hidden nodes, we find a decrease in the standard deviation. In addition, there's a clear difference between neural networks with different activation functions here. The unbounded softplus holds a larger standard deviation as the number of hidden nodes increases, whereas the bounded sigmoid activation function remains below across all number of hidden nodes.

4.3.3 Wavefunction Variance Analysis

Increasing the number of hidden nodes leads to a near constant variance regardless of the number of hidden nodes which again seems to contradict the bias-variance trade-off. The trade-off dictates the variance should continually increase as the parameter space of the neural network increases [76]. The origin of this constant variance emerges from the deuteron being solved within relative momenta. As can be seen with Eq. 4.8, in the case of calculating the overlap of two arbitrary wavefunction, all integrals carry a q^2 pre-factor. This results in all integral components for low-momenta, i.e. $q < 1\ \text{fm}^{-1}$, contributing a near negligible amount towards a particular integral. In fact, one could arbitrarily change the wavefunction for either state at low-momenta and the binding energy, fidelities, and D-state probability would be negligibly affected. This emerges from the use of a global loss function rather than a more 'local' loss function that doesn't penalise mismatches for low-momenta and take into account the local structure of the state wavefunction. In fact, Fig. 4.5 clearly highlights this behaviour. As the number of hidden nodes increase the variance at low-momenta increases, as does the bias. The asymptotics of each activation functions seem to indicate the asymptotics of the state wavefunctions here too. For example, the state wavefunctions with a sigmoid activation functions experience curvature, and as H increases too much curvature. For a Softplus activation functions the opposite occurs, it overshoots linearly in all cases albeit with an increase in variance. In the future, the use of specially designed activation functions that incorporate physical attributes of a given physical system, and secondly directly incorporating boundary conditions to the model.

Another interesting takeaway is that the variance can sharply change as a function of H, a clear example of this the softplus activation function for the S-state where there

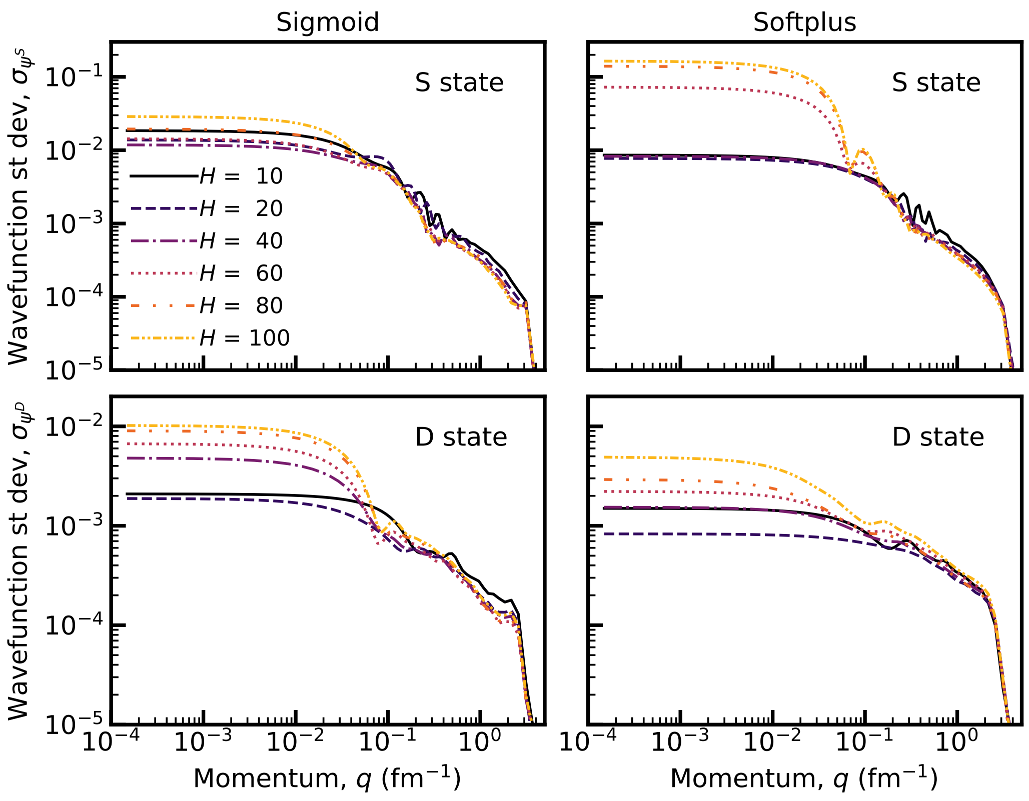


Figure 4.5: The standard deviation of the initial 50 seed configurations for a given state wavefunction. The top (bottom) row shows the $S(D)$-state wavefunction standard deviation as a function of momentum with the left (right) columns denoting the activation function sigmoid (softplus) respectively.

are effectively two regions of variance, $H \leq 40$ and $H \geq 60$ which differ by over an order of magnitude. An additional conclusion from these results indicates that different activation functions perform better, in terms of variance, for different states: the sigmoid (softplus) performs better for the S-state (D-state) wavefunctions. However, this ties in with the previous discussion directly incorporating physical knowledge into the ansatz via envelope function or Jastrow factors [68, 70].

In Fig. 4.5 the standard deviation of the wavefunction is shown as a function of momenta as a log-log plot which highlights how the network's prediction of the wavefunction varies for a given activation function and number of hidden nodes, H. The standard deviation, σ_{ψ}, at finite momentum is near zero for both states and both activation functions indicating consistent predictions and robust methodology. However, for low-momenta there is a clear structure to the standard deviation in relation to H. As the number of hidden nodes increases from 10 to 20 the variance decreases. However, any additional increase in the number of hidden nodes increases the variance. The exception being the S-state for a Sigmoid activation function where the minimum variance occurs with $H = 40$. The high-momentum components of the wavefunction are indifferent to the variance. This would indicate a local interpretation to the bias-variance trade-off where the q^2 factor of the integrands effectively acts as a 'sink' absorbing any variance of higher-order terms to low-momenta components of the wavefunction.

The initial hypothesis was that the bias in the calculation of the binding energy emerged solely as a result of low-momenta mismatching. This hypothesis was tested by replacing the low-momenta components of the wavefunction with the *exact* ground-state wavefunction and measuring how the ground-state energy changes. However, it was noted that this hypothesis was *not* correct and there was minimal change in the ground-state energy. This emerges because the loss function is the expectation value for the energy which has a q^2 phase factor within its integral. Therefore, any contribution towards the integral will be weighted by q^2 for low values of q. This results in low-momenta values being negligible to the calculation of the loss, and hence, will have negligible impact on the updating of the network's weights. This results in the initial hypothesis being wrong, and correctly concludes that the current error within the methodology is due to mismatches as *finite* momentum rather than low-momenta values.

4.3.4 Wavefunction Bias Analysis

In Sec. 4.3.3 the variance of the network is analysed with the conclusion that the q^2 pre-factor is primary culprit in affecting the convergence of the network towards the ground-state wavefunction. In this section, we perform analysis of the wavefunction directed towards the network's bias, i.e. compare the explicit difference between the exact wavefunction and the network's prediction. One way of quantifying this is to compute the difference between the exact wavefunction and the network's wavefunction for each state of the deuteron.

In Fig. 4.6, we show the difference, $\sum_i \omega_i \left(\psi^{L,2}_{ANN,i} - \psi^{L,2}_{GS,i}\right)$, for the S-state (panel a), D-state (panel b), sum of S- and D-states (panel c), and the cumulative sum of S- and D-states for the kinetic energy (panel d). This measure defines a scalar difference between the network's predicted state wavefunction, $\psi^{L,2}_{ANN,i}$, and the exact state wavefunction, $\psi^{L,2}_{GS,i}$. In the ideal case, this measure equals to zero when both state wavefunctions are identical. This highlights the effect of a *global* loss function at *local* momentum values. These results are for a network of $H = 20$ with all 50 different seed configurations shown in the same plot with the average denoted in bold. Results for other values of H are qualitatively similar. Both activation functions are shown in the same plot to highlight the effect of how changing the activation function affects the performance of the network in representing the exact ground-state wavefunction. The results are shown in a log-log plot to explicitly highlight the large change of magnitude of the error against momentum. Within the figure, all 50 pseudorandom runs can be seen in shades of blue and green with the solid blue and green representing the mean value for Sigmoid and Softplus respectively.

In the top panel, the S-state difference is shown. We effectively see the error in the network's prediction of the ground-state wavefunction at different values of momentum. We can see that the differences in the wavefunction aren't entirely restricted towards low-momentum values as initially shown within Fig. 4.3. In fact the bulk of the error actually occurs at intermediate momentum values. In addition, the error for both activation function are both positive and negative indicating that the network can indeed represent the wavefunction, although, it suffers from a bias-variance trade-off which indicates it struggles to deal with some of the finite-momentum structure. The same analysis can be stated for panel b which shows the D-state wavefunction difference. The 'spread' of the error as a function of momenta is restricted to the region between $q = 0.2$ fm^{-1} and $q = 3$ fm^{-1}, and is smaller for the D-state than the S-state. This could occur due to the fact that low-momentum values of the D-state are near zero, and hence any difference will also be near zero. The errors are located at intermediate values of momentum with the maximal error occurring at the turning point in the D-state wavefunction. This can be also seen in Fig. 4.3 with the noticeable increase in the variance at the turning point. In both the S- and D-state errors here, the large momentum values are near exact with negligible error. This emerges due to the fact that the wavefunction should equal zero as $q \mapsto \infty$ which will result in the difference being near zero too.

Panels c and d show the cumulative sum of the (c) squared wavefunctions and (d) the kinetic energy of both state wavefunctions. The third subfigure highlights how the total error varies with q. The main takeaway here is that the main component of the error, over both state wavefunctions, is situated at intermediate values of the momentum with the difference going to zero at high momentum values of the grid. The sum goes to zero as $q \mapsto 500$fm^{-1} because at $q_N = 64$ the cumulative sum equals the integral, and hence the difference in these normalised integrals equals zero. Direct inspection of the state wavefunctions in Fig. 4.3 shows very little disagreement of the functional form of the wavefunction. However, it must be stated that each value of q has an associated weight ωq^2 which will affect how the error at a given value of q affects the error in the overlap integral. In the bottom panel, we can see the cumulative sum of the S and D state for the kinetic energy contributions. In this case, each part of the numerical mesh is weighted by ωq^4 and hence is extremely sensitive to error for intermediate momentum, and effectively negligible to low momentum. This subfigure highlights the error contributions within the kinetic energy with the final momentum value representing the integral for the kinetic energy. A value of 10^{-3} MeV is highlighted in a bold dashed grey line within Fig. 4.6. For each seed, the final difference in kinetic energies is of the order of the error in the kinetic energy. This error is on the order of ± 0.1 MeV which, more importantly, is relatively constant regardless of the network initialisation. Again, this error is both positive and negative indicating that variance dominates the error. This indicates that a sub-set of the results will overestimate the ground-state wavefunction and a different sub-set of the results underestimate the ground-state wavefunction which

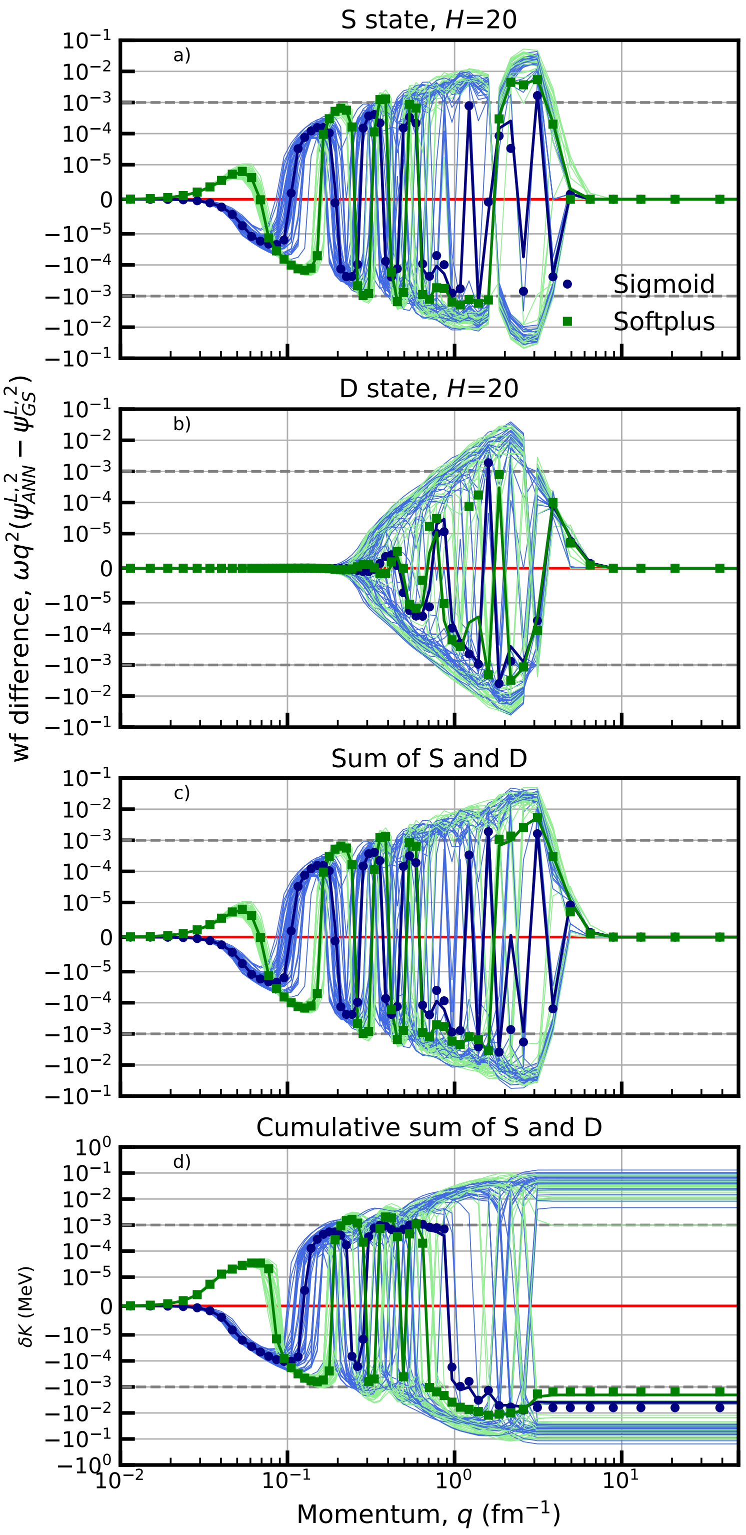


Figure 4.6: The difference in wavefunction amplitude between the network and the exact ground-state wavefunction for a Sigmoid (Softplus) activation function in blue (green). The difference is shown for the S-state, D-state, sum of both states, and its cumulative sum in each panel respectively.

results in the mean value having a *reduced* error on the order of $\sim 10^{-3}$. Interestingly enough, the spread across all 50 runs is extremely consistent for both of the bottom panels in Fig. 4.6, then at $q \approx 0.1$ fm^{-1} there is a widening in the error for the kinetic energy contribution which by $q \sim 1$fm^{-1} separates into the aforementioned behaviour of some runs overestimating the kinetic energy contributions and other runs underestimate it.

To conclude, the bias analysis of the wavefunction highlights some key points about a solving for the ground-state wavefunction with a neural network. Primarily the use of a global loss function without directly encoding boundary conditions into the wavefunction can lead to increase variance when predicting the shape of the wavefunction. In addition, the use of solving for the ground-state wavefunction as a function of relative momentum can penalise the loss function by effectively weighting particular values of momentum more importantly than others[1]. This allows for low momentum values to be effectively ignored during the minimisation process. A positive aspect of this methodology is that the network's performance acts independent of the initialisation indicating a robust methodology. For further statistical analysis, as well as more complicated network architectures, we direct the reader to Ref. [142].

[1] This can become particular problematic within two-dimensional and three-dimensional settings.

Chapter 5

Non-interacting Fermions

In this chapter, we will study a system of A trapped spinless fermions in one dimension. The Hamiltonian, $\hat{H}$, in ho units, is defined as,

$$\hat{H} = \sum_i h_i, \tag{5.1}$$

$$h_i = -\frac{1}{2}\nabla_i^2 + \frac{1}{2}x_i^2, \tag{5.2}$$

which represent the kinetic and potential energy contributions respectively. The eigenvectors of h_i are the Hermite polynomials and the single-particle levels are equidistantly separated by an energy of $\left(n + \frac{1}{2}\right)$ in units $\hbar\omega$, where n is the principal quantum number. For this Hamiltonian, the ground-state wavefunction can be found analytically allowing for exact comparison. It is also available via a variety of numerical methods. In this thesis the method of choice is VMC which is broken down into two phases; pre-training, and energy minimisation. The methodology of the pre-training and energy minimisation phases will be described in Secs. 5.1.1 and 5.1.2 respectively. The results of the vANN methodology will be described within Sec. 5.2 through Sec. 5.3.

5.1 Methodology

As stated the numerical method of this work is VMC and the methodology is formed of two phases; pre-training, and energy minimisation. The extension to VMC within this thesis entails the use of representing the wavefunction as a DNN which has been referred to as Deep VMC within the literature [66]. The pre-training phase trains the initial network to represent some 'training' function so that when the energy minimisation phase begins with an initial expectation value of the energy it is not only reasonable but also numerical stable. The energy minimisation phase is where the ground-state energy is calculated via a standard VMC algorithm. A wavefunction is represented by a DNN ansatz from which samples are drawn in proportion to its Born probability. The expectation value of the energy is calculated from these samples, and by the use of AD techniques the derivative of the expectation of the energy with respect to the variational parameters of the ansatz are calculated. These gradients are used within gradient-descent-based algorithm to stochastically optimise the parameters such that the expectation value of the energy is minimised.

5.1.1 Pre-training

The pre-training methodology of Chapter 4, which involves maximising an overlap between Ψ_{vANN} and Ψ_{targ}, failed when applied towards a more general A-body DNN. Therefore, the pre-training scheme used within this chapter is that of Ref. [5] where we seek to maximise an overlap between between individual orbitals of the DNN and a set of target orbitals. The pre-training formula that was used within this work does differ from Ref. [5]. The most notable differences are from the lack of spin index due to the fact we aim at describing spinless fermions. The more subtle difference is within the sampling distribution, P_{vANN}. In Ref. [5], the sampling distribution for the pre-training was spilt 50-50 between the network's Born PDF and the target wavefunction's Born PDF this was done to ensure that the walkers were distributed equally between the network and the target wavefunction [5].

In this work, this 50-50 procedure was initially tried but significantly slowed down runtime of the pre-training with no improvement in the pre-training loss; possibly due to the use of a simpler Hamiltonian. Therefore, the sampling distribution was restricted to just the network's Born PDF. By using the Born PDF, allows for an efficient form of importance sampling when computing many-body expectation values for the energy of the network with a smaller variance than naïvely using brute-force sampling methods. The pre-training loss is defined as,

$$\mathcal{L}^{Pre}(\theta) = \int \left[\sum_{ijk} \left(\phi^k_{vANN,i}\left(x_j; x_{\setminus j}\right) - \phi_{exact,i}\left(x_j\right)\right)^2 \right] P_{vANN}(x_1, \ldots, x_A)\, dx_1 \ldots dx_A \tag{5.3}$$

where the sampling distribution is defined as,

$$P_{vANN}\left(x_1, \ldots, x_A\right) = \frac{|\Psi\left(x_1, \ldots, x_A\right)|^2}{\int |\Psi\left(x_1, \ldots, x_A\right)|^2 dx_1 \ldots dx_A}. \tag{5.4}$$

Here $\phi^k_{vANN,i}\left(x_j; \{x_{\setminus j\}}\right)$ is the i^{th} orbital for the j^{th} fermion in the k^{th} determinant of the DNN. The term $\phi^k_{exact,i}$ are the single-particle orbitals of the analytical solution to the non-interacting ground-state defined as,

$$\phi_{exact,n}(x) = \mathcal{N}_n \exp\left(-\frac{x^2}{2}\right) H_n(x) \tag{5.5}$$

with $\mathcal{N}_n = (2^n n! \sqrt{\pi})^{-1/2}$ and $H_n(x)$ being the nth order Hermite polynomial. The loss as measured between individual orbitals of the k determinant is a squared difference between the exact orbital and the DNN's i^{th} orbital. The total loss is the sum over all individual orbital losses and all determinants. The pre-training phase is defined by four separate sub-phases; sample generation, pre-training loss, loss gradient calculation, and parameter updates.

The pre-training loss of Eq. 5.3 can be approximated via the statistical expectation of,

$$\mathcal{L}^{Pre}(\theta) = \mathbb{E}_{x \sim P_{vANN}(x_1,\ldots,x_A)} \left[\sum_{ijk} \left(\phi^k_{vANN,i}\left(x_j; x_{\setminus j}\right) - \phi_{exact,i}\left(x_j\right) \right)^2 \right]. \tag{5.6}$$

where θ are the variational parameters of the DNN ansatz we seek to optimise. The initial samples are drawn from P_{vANN} via the use of the Metropolis-Hastings algorithm (Sec. 2.8.2) with a thinning value of 10. With the assumption that the samples have decorrelated we calculate a 'local' pre-training loss (quantity defined within square brackets of Eq. 5.6) for each sample, we calculate the expectation to define the pre-training loss. The gradient of the pre-training loss with respect to the variational parameters, θ, is calculated via PyTorch's autograd engine [130]. This is subsequently used to update the parameters via the Adam optimiser (Sec. 3.8.3). This process is repeated for 10^4 epochs which updates the wavefunction from an initially random structure from Xavier or He initialisation [138, 107] towards a more 'physical' wavefunction. This pre-training allows for a more numerically stable wavefunction when performing the energy minimisation phase which results in more stable convergence. If pre-training was not performed before the energy minimisation phase then the likelihood of the convergence breaking down increases as already indicated in Refs. [5, 6, 64].

As discussed in Sec. 3.5, the DNN ansatz is constructed of H number of hidden nodes, L number of layers, and D number of GSDs. Therefore, when pre-training the DNN ansatz these values need to be selected to construct a DNN of a given architecture. And, as stated by the UAT (Sec. 3.3) the architecture will affect the performance of the DNN ansatz on representing the target wavefunction. Here, we explore the dependencies of width, H, and depth, L, in the DNN. The values are varied as H = 32, 64, and 128, and L = 1, and 2. This showcases how the number of parameters can affect performance from the ~160 (H=32, L=1) to ~34,000 (H=128, L=2) parameters. The number of GSDs is fixed to one to highlight the network's compressibility. The results of the pre-training loss (Eq. 5.3) as a function of epochs against different DNN architectures is shown in Fig. 5.1 for A = 2 to 6 fermions.

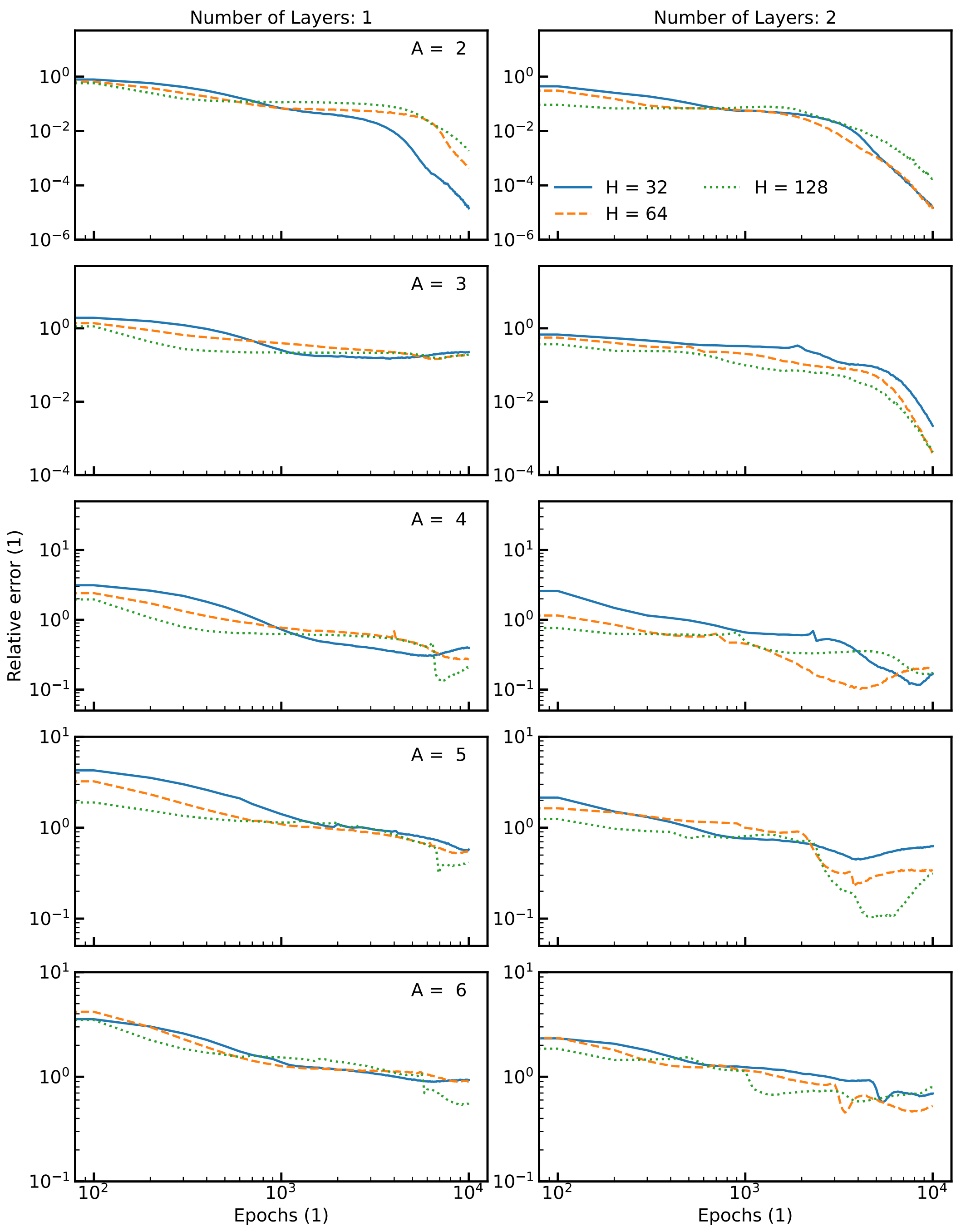


Figure 5.1: The expectation value of the pre-training loss for varying hidden width, H, and number of layers, L, for a single generalised Slater determinant, $D = 1$, as a function of epochs. Different rows correspond to increasing values of A, from 2 to 6. The y-axis is fixed within a given value of A, but changes from row to row.

The pre-training loss has a global minimum at $\mathcal{L}^{Pre}(\theta) = 0$, and hence any difference indicates a mismatch in the functional form of the exact ground-state orbitals. As the exact target wavefunction is comprised of the ground-state orbitals, this pre-training loss effectively falls under the ML concept of supervised learning. In this scenario the network is effectively just performing a modified form of linear regression to a pre-determined functional form with samples that are distributed in proportional to $P^{Pre}_{vANN}(X)$. In this use case the target wavefunction coincides with the exact ground-state wavefunction, due to it consisting of the ground-state orbitals. The results of the pre-training are shown in Fig. 5.1 and highlight how the width, measured by H, and the depth, measured by L, affect the pre-training of the DNN ansatz.

Overall, we can see that increasing the depth, L, of the DNN ansatz can help reduce the pre-training loss, especially for $A \leq 3$ where the use of an additional layer allows the final 1000s of epochs to improve the pre-training loss by a few orders of magnitude. Furthermore, at the final epoch the final pre-training loss, $\mathcal{L}^{Pre}(\theta)$, gradually increases as A increases. This is to be expected as the DNN ansatz remains a fixed size yet the Hilbert space of the target wavefunction increases exponentially with respect to A. Interestingly enough, the size of H doesn't seem to greatly impact the performance of the pre-training with the final pre-training being somewhat consistent between $H = 64$ and 128. This indicates that DNN ansatz is able to give consistent results as long as it has *enough* hidden nodes to represent a given system. On the other hand, the depth of the DNN ansatz does not seem to hugely affect the pre-training either. The rate of convergence for $A = 2$ is initially quick with $P^{Pre}(\theta)$ falling on the order of 10^0 within the first 10^3 epochs which allows for the energy minimisation phase to being in a stable manner. For $A \leq 3$ significant improvement in $P^{Pre}(\theta)$ is found in the post 10^3 epochs with more minimal improvement for $A \leq 4$.

5.1.2 Energy Minimisation

Once the pre-training phase of the DNN has finished, the DNN represents a more 'physical' wavefunction that differs from the exact target wavefunction by the pre-training loss, $\mathcal{L}^{Pre}(\theta)$. This more 'physically sound' wavefunction also has a lower associated energy than the initial wavefunction. The same methodology of the pre-training phase can be modified to find the ground-state wavefunction.

The algorithm used for finding the ground-state wavefunction is near identical to that of Sec. 5.1.1. The difference between these two phases is the different loss function that is used. In Sec. 5.1.1, Eq. 5.6 acts as a modified form of linear regression where a functional form is shown to the DNN and its goal is to fit this functional form by minimise a mean-squared error. The use of a mean-squared error works as we have a functional form to which we train, however, in the energy minimisation we do not know the functional form of the groundstate. If we were to know it, then the entire phase would be redundant as we'd already have the answer. Therefore, the loss function moves from a 'supervised' task of linear regression to an exact target value to a 'reinforcement' task of minimising an expectation via the Variational principle. The loss function for this phase is changed to the expectation value of the energy,

$$\langle E_{vANN} \rangle = \frac{\langle \Psi_{vANN} | \hat{H} | \Psi_{vANN} \rangle}{\langle \Psi_{vANN} | \Psi_{vANN} \rangle} \approx \mathbb{E}_{x \sim |\Psi|^2} \left[\frac{\hat{H} \Psi_{vANN}}{\Psi_{vANN}} \right]. \tag{5.7}$$

The gradients of the energy with respect to the variational parameters of the DNN are calculated via AD, and updated via the Adam optimiser (Sec. 3.8.3). The algorithm is repeated for 10^5 epochs to ensure convergence. If the convergence is achieved before the final epoch, then all remaining epochs will effectively reduce the variance in the expectation value of the energy which leads to a refinement of the ground-state energy value.

In order to further improve the numerical stability on the estimate of the derivative, we follow Ref. [5] and construct an auxiliary loss function whose output differs from the expectation value of the energy but its derivative is equal. This is implemented via exploiting the 'detach' function. The detach function is defined as,

$$\perp(x) = x, \tag{5.8}$$

$$\frac{\partial \perp(x)}{\partial x} = 0. \tag{5.9}$$

This abstract function is best viewed from the viewpoint of DAGs (Sec. 3.6.1) rather than an explicit mathematical function. A DAG is comprised of nodes which are connected via edges which are differentiable mathematical operations. The detach function removes edges between specific nodes so that they are 'detached' from the DAG. Therefore, when computing the derivative with respect to this detached variable leads to zero as it has no history of operations [143]. This trick allows for a more numerically stable computation of the energy derivative. The expectation value of the energy already contains a second derivative term, the kinetic energy. Therefore, when differentiating the energy would result in a third derivative term which will compound floating point errors at every order of the derivative. Hence, utilising this trick to keep the maximum derivative at second order will aid numerical stability. The auxiliary loss function is defined as,

$$\mathcal{L}^E(\theta) = 2\mathbb{E}_{x \sim |\Psi_{vANN}|^2} \left[\perp \left(E_{Loc} - \mathbb{E}_{x \sim |\Psi_{vANN}|^2} [E_{Loc}] \right) \ln |\Psi_{vANN}| \right] \tag{5.10}$$

whose derivative with respect to the variational parameters, θ is,

$$\frac{\partial \mathcal{L}^E(\theta)}{\partial \theta} = 2\mathbb{E}_{x \sim |\Psi_{vANN}|^2} \left[\left(E_{Loc} - \mathbb{E}_{x \sim |\Psi_{vANN}|^2} [E_{Loc}] \right) \frac{\partial \ln |\Psi_{vANN}|}{\partial \theta} \right]. \tag{5.11}$$

This is mathematically equivalent, under statistical expectation, to the derivative of the energy with respect to the variational parameters. An additional trick to enhance numerical stability is to utilise a clipping based on an ℓ_1-norm to stop outlier values of local energy of Eq. 5.10 from significantly skewing the mean [5]. The variation, ξ, over a batch of local energies is defined as,

$$\xi = \mathbb{E}_{x\sim|\Psi_{vANN}|^2}\left[\left|E_L - \tilde{E}_L\right|\right] \tag{5.12}$$

where $\tilde{E}_L$ is the median local energy. Eq. 5.12 is an ℓ_1-norm equivalent to the usual, ℓ_2-norm standard deviation. Using this, the local energy values are clipped within a window of $\tilde{E}_L \pm c\xi$. If a local energy value lies outside the range $\left[\tilde{E}_L - c\xi, \tilde{E}_L + c\xi\right]$ it is replaced with the respective outer boundary of the window, i.e.,

$$\mathrm{clip}(E_L) = \begin{cases} \tilde{E}_L - c\xi & \text{if } E_L \leq \tilde{E}_L - c\xi \\ \tilde{E}_L + c\xi & \text{if } E_L \geq \tilde{E}_L + c\xi \\ E_L & \text{otherwise.} \end{cases} \tag{5.13}$$

where c is the clipping constant which is kept to $c = 5$ for all simulations [5]. This clipping is only applied to the local energy values, and not the other quantities like $\frac{\partial \ln|\Psi|}{\partial \theta}$. The clipped local energies (in accordance to Eq. 5.13) are passed into Eq. 5.10 to compute a 'clipped' form of the loss. The value of $\mathcal{L}^E(\theta)$ is differentiated via AD and the variational parameter is updated via the Adam optimiser (Sec. 3.8.3).

The convergence of the energy minimisation phase is shown in Fig. 5.2 for a wide variety of network architectures and number of fermions. All network architectures were pre-trained to a target wavefunction in accordance to Sec. 5.1.1 and subsequently minimised via the algorithmic process of VMC (Sec. 2.7). The behaviour of the convergence of each architecture can highlight an optimal architecture of the DNN which translates into better observable calculations. In this case, the global minimum for the non-interacting case is known analytically (Eq. 2.14), however, the training of this network doesn't known the global minimum. The DNN ansatz is working within a reinforcement setting where the DNN effectively estimates the expectation value of the energy and then adjusts its parameters under a gradient descent algorithm without knowing the 'target' ground-state energy. This will converge, in the large limit of epochs, to the nearest local minimum, relative to its initial position, within the parameter space of the DNN ansatz.

In Fig. 5.2, the network's expectation value of the energy is compared against the analytical ground-state energy, and is shown for $A = 2$ to 6 fermions for all aforementioned DNN architectures. The number of epochs for the energy minimisation is an order of magnitude larger in order to ensure the DNN has converged to the ground-state as well as following the methodology of Refs. [5, 6, 64]. The DNN ansatz is optimised via the Adam optimiser of Sec. 3.8.3 with a learning rate, α, of 10^{-4} with the remaining hyperparameters being their default values.

The convergence of all DNN architectures show that they can represent the ground-state wavefunction of the respective physical system to at least 1% accuracy. However, in some situations DNNs can achieve a relative error on the order of 10^{-4} to 10^{-6} (0.01% to 0.0001%) [1]. One main difference when analysing the convergence is the effect of including an additional hidden layer. In the $L = 1$ case (left panels), increasing the number of hidden nodes seems to improve the rate of convergence for larger values of A or at least give similar performance to the $H = 32$ and 64 cases. However, when moving towards the $L = 2$ case (right panels) we can see that the $H = 128$ results have a higher expectation value of the energy than narrower DNN ansätze. This seems to indicate a bias-variance trade-off of H with larger L that impacts the final stages of convergence and introduces a source of statistical noise. In the case of $A = 6$, we see the $H = 64$ slightly improves upon the energy for $H = 32$ but both improve upon $H = 128$.

To conclude, the DNN ansatz performs best with $H = 64$ and $L = 2$ for both the pre-training procedure and energy minimisation due to its lower expectation value of the pre-training loss and energy respectively.

[1] Some visual artefacts do exist within the convergence plots for some network architectures (e.g. $A = 5$, $L = 2$) with linear convergence followed by highly stochastic oscillations, examples like this are artefacts of the datafile (e.g. missing epochs)

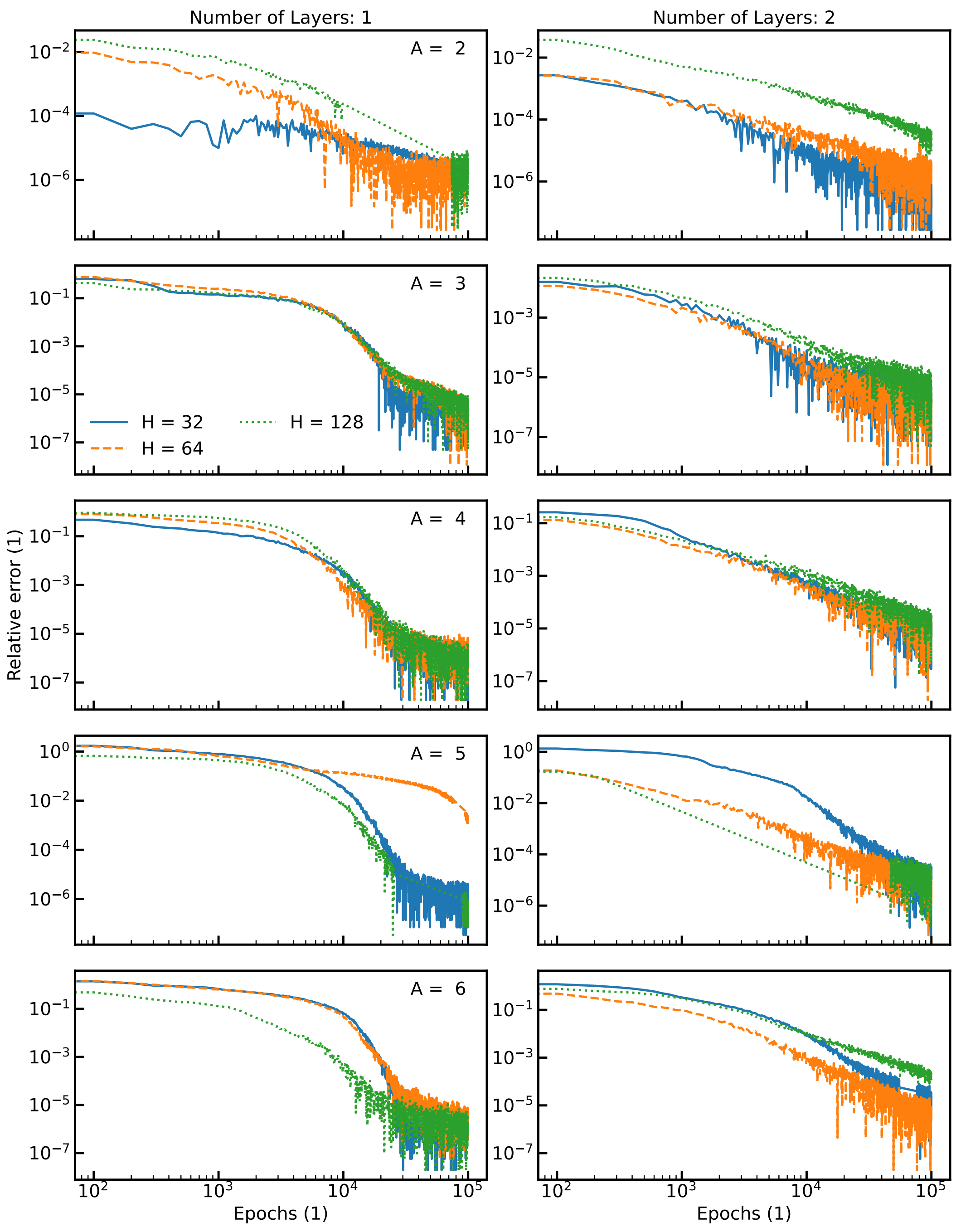


Figure 5.2: The convergence of the DNN ansatz for the expectation value of the energy for varying hidden width, H, and number of layers, L, for a single generalised Slater determinant, $D = 1$, as a function of epochs.

Having discussed the methodology used within the VMC algorithm to compute the ground-state wavefunction, we will now move towards discussing the results of the DNN ansatz in the non-interacting case. The results of this section are determined by the following properties; 64 hidden nodes per layer ($H = 64$), 2 equivariant layers ($L = 2$), a 1 generalised Slater determinant ($D = 1$).

5.2 One-body Density

The one-body density (OBD) is a positive-semi definite function of one-dimensional position, x. It has the physical interpretation of $\rho(x)dx$ being the number of particles between x and $x + dx$ with all others being at arbitrary positions [144]. The one-body density is defined as,

$$\rho(x) = \int_{-\infty}^{\infty} \delta\left(x - x_1\right) |\Psi(x_1, \ldots, x_A)|^2 \, dx_1 \ldots dx_A. \tag{5.14}$$

The OBD is computed stochastic via sampling sampling $x \sim |\Psi|^2$, and taking only the x_1 values from x, which are subsequently plotted in a histogram that's normalised to the number of particles, A.

The OBD is determined via a stochastic methodology. By sampling, $x \sim |\Psi|^2$ we find many-body positions distributed in proportion to the wavefunction. The position value for x_1 is selected from these values and subsequently represented as a histogram which is normalised to the number of particles, A. The act of plotting the OBD as a histogram allows a stochastic method like VMC to efficiently compute such a property. In addition, the bin width of the histogram effectively discretises the OBD and allows for a point-wise comparison with other numerical and exact methodologies. For the one-body density we compare the vANN results against the exact one-body densities which are shown in Fig. 5.3. The vANN results have excellent agreement for the one-body densities over wide range of x for all number of particles studied. The peaks and troughs of these densities are accurately calculated at the correct densities and the asymptotics are also equally well defined with the exact results. This indicates that the methodology of the vANN results can accurately, and efficiently, reproduce the one-body density.

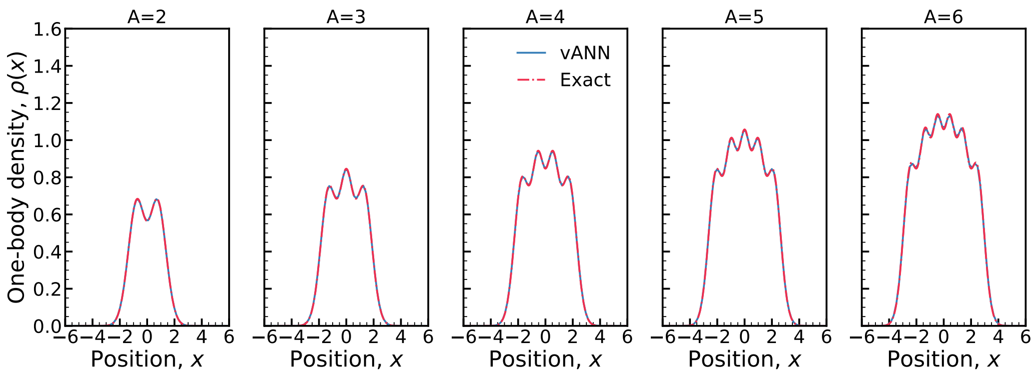


Figure 5.3: The one-body density for the ground-state wavefunction for $A = 2$ to $A = 6$ particles for the vANN results (blue), and exact results (red).

5.3 One-body Density Matrix

The one-body density matrix (OBDM) is a function of one-dimensional positions, x and x' which defines how two positions correlate to one another for a given wavefunction. This matrix is Hermitian and Positive Semi-Definite (PSD) which results in strictly positive eigenvalues. The eigenvalues and eigenvectors of the OBDM are defined as the occupation numbers and natural orbitals respectively [144]. The analytical solution for the HO potential is also a Slater determinant and hence the vANN results can be directly compared against an exact solution. The one-body density matrix is defined as,

$$\rho(x,x') = \int_{-\infty}^{\infty} \delta(x-x_1)\,\delta\left(x'-x_2\right)|\Psi(x_1,\dots,x_A)|^2\,dx_1\dots dx_A. \tag{5.15}$$

This is calculated via the algorithm of Ref. [18], in which Eq. 5.15 is calculated as a statistical expectation rather than a deterministic integral. This allows for the density matrix for an arbitrary number of particles to be calculated efficiently [18]. The algorithm is defined as follows. Firstly, sample a set of many-body positions from the wavefunction using a Metropolis-Hastings (MH) sampler, $X \sim |\Psi|^2$. Then sample a 'ghost' particle from a uniform distribution, $s \sim \mathcal{U}[a,b)$ where a and b are the extrema of the uniform distribution. Next calculate the sign and log-absolute wavefunction from the wavefunction for X; $\operatorname{sgn}(\Psi(X))$, $\ln|\Psi(X)|$. Now define a new set of many-body positions, X', where $x_1 \in X$ is replaced with the ghost-particle, s, i.e. $X' = \{s, x_2, \dots, x_A\}$. Then calculate the sign and log-absolute wavefunction for the wavefunction with the new many-body position, X'; $\operatorname{sgn}\left(\Psi\left(X'\right)\right)$, $\ln\left|\Psi\left(X'\right)\right|$. The OBDM can be created via histogramming the x_1 and s sampled positions by weighting the i^{th} contribution towards the histogram by,

$$w_i = \frac{\operatorname{sgn}(\Psi(X_i))\operatorname{sgn}\left(\Psi\left(X_i'\right)\right)}{\mathcal{U}(s_i)} \exp\left(\ln|\Psi(X_i)| - \ln\left|\Psi\left(X_i'\right)\right|\right). \tag{5.16}$$

In practice we run the MH sampler with 4096 walkers in parallel for 10^4 batches of walkers, and then concatenate all walkers over all batches together into a single superset of approximately 4.1×10^7 samples which we assume are *i.i.d* under the thermalisation of the Markov chain. Each sample in the superset has an associated weight defined by Eq. 5.16. These results are fed into a 2D histogram which creates a stochastic approximation to the one-body density matrix on a discretised grid, determined by the histogram's bin width, which is subsequently trace-normalised to the number of particles. The OBDMs calculated via the method of Ref. [18] for $A = 2$ to 6 are shown in the bottom panels of Fig. 5.4.

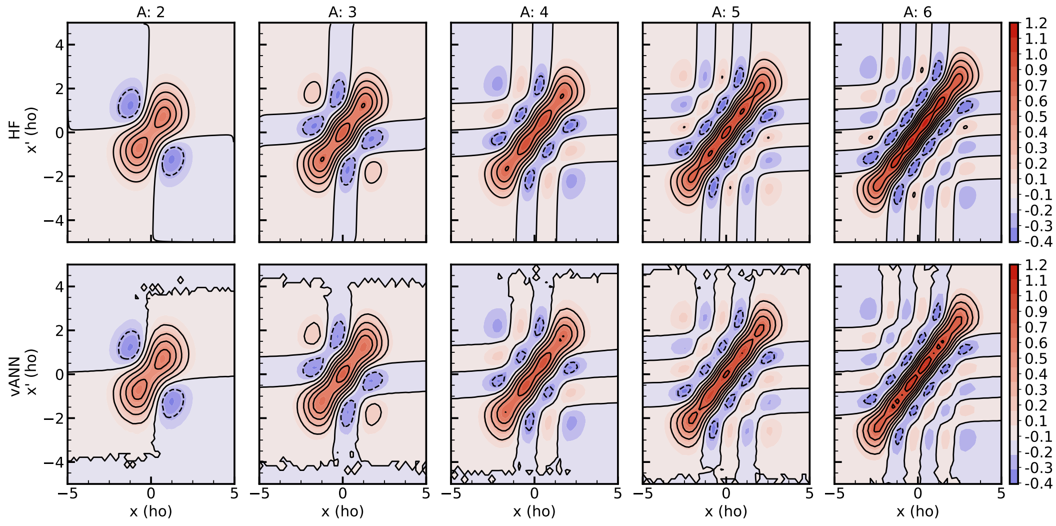


Figure 5.4: The one-body density matrix (OBDM) for $A = 2$ to 6 is shown for both the exact results (top row) and the ML approach of vANN (bottom row).

The OBDM can adequately and succinctly describe the internal structure of many-body wavefunctions [144], and by comparing the exact results with the vANN results can quantify how effective this new methodology is. The overall comparison provided by Fig. 5.4 shows a good benchmark between methodologies albeit with some minimal visual artefacts due to the stochasticity of the ML approach. The methodology used to calculate the OBDM was the 'ghost'-particle statistical method of Ref. [18]. The central bands, that emerge from the PEP, within the OBDM hold a similar structure with the peaks and troughs having similar magnitude and functional forms regardless of methodologies. This indicates the DNN used to represent the ground-state wavefunction can efficiently reproduce the exact results for the non-interacting case. The range of the ghost particle distribution could be fixed to a pre-determined range, however, it was favoured to allow the algorithm to adapt this range on the fly in proportion to the number of particles so that the samples are distributed more densely to ensure the histogram has sufficient statistics.

The discrepancies between methodologies seem to emerge purely from the statistical method used to calculate the OBDM from the vANN results. For example, the OBDM truncates at $x' \sim 4$ ho which is more prominent for smaller A but diminishes as A increases. This mismatch in support over domains emerges from the use of multiple batches when computing the OBDM. The first batch of many-body positions, which are sampled from $X \sim |\Psi|^2$, determine the range of the ghost particle distribution, $\mathcal{U}[a, b)$, where a and b denote the lower and upper bound of the uniform distribution respectively. The x_1 coordinate is selected from X, and its absolute maximum is used to the limits of $\mathcal{U}[a, b)$ with the values of a and b being set to $-\max|x_1|$ and $+\max|x_1|$ respectively. This ensure that a OBDM with equal lengths in x and x' can be created. These limits, for the ghost-particle distribution, could be fixed across values of A but it was chosen to adapt these values in order for the histogram to have sufficient statistics. Furthermore, the exact results for the OBDM were constructed on a discretised grid via an exact methodology. This grid size was 200×200 for $A \leq 4$ and a 240×240 for $A \geq 5$. For the vANN results, the discretised grid was a sparser 100×100 grid which was motivated by ensuring that each bin had sufficient statistics. However, this does lead to some artefacts within the OBDM with poorer resolution for the smaller peaks and troughs for larger many-body systems.

The physical interpretation of the OBDM depends differently on the on-diagonal and the off-diagonal matrix elements. The on-diagonal elements represent the one-body density of the wavefunction [144]. The off-diagonal matrix elements are more abstract but can be interpreted as a sum over the 'joint-amplitudes' for a pair of natural orbitals. For example, in the two fermion case (A=2) there are two single-particle natural orbitals, $\phi_n(x)$,

$$\phi_0(x) = \pi^{-1/4} \exp\left(-\frac{x^2}{2}\right), \tag{5.17}$$

$$\phi_1(x) = 2^{1/2}\pi^{-1/4} x \exp\left(-\frac{x^2}{2}\right). \tag{5.18}$$

where x is the one-dimensional position of an arbitrary fermion, and n is the orbital index.

In Fig. 5.5, both natural orbitals for the A=2 case are shown in the left most panel. The centre two panels are the outer products, $\phi_n^*(x)\phi_n(x)$, of the n=0 and n=1 orbitals respectively. The right most panel is the sum of the centre two panels, i.e. the sum of outer products for the n=0 and n=1 orbitals. These outer products of a given orbital denote a form of 'joint-amplitude' of how the amplitude changes between two points within the one-dimensional space. In total, the OBDM can be defined as the sum over all outer product of orbitals, i.e. $\rho(x, x') = \sum_n \phi_n(x)\phi_n(x')$. The ground-state orbital is a Gaussian and the first-excited orbital is x multiplied by a Gaussian. This results in contribution of the OBDM from the ground-state as a Gaussian, and the first-excited states contribution is a Gaussian with an xx' pre-factor. This pre-factor acts as a boundary which separate each quadrant of the 1st component of the OBDM.

For the $A = 2$ case, the structure of the OBDM is a two-mode Gaussian along the diagonal which represents the number of particles within the system at hand. Along the off-diagonal components of the OBDM there exists a single, much smaller, peak. This peak emerges from the joint-amplitude of the first excited state which dominates the contribution of the OBDM at the off-diagonal quadrants.

In the $A \geq 2$ case, the on-diagonal component of the OBDDM will have A modes representing each particle within the system. As for the off-diagonal components of the OBDM, a ripple-like effect emerges from the sum over all natural orbitals. As one moves further away from the on-diagonal component of the OBDM, contributions from higher orbitals will begin to dominate over lower orbitals in the same case of $A = 2$. The number of bands above and below the off-diagonal is one less than the number of particles in the system. Within each band, the number of peaks/troughs decreases by one within each successive band. For example, in Fig. 5.5 there are two peaks within the central band which lies along the diagonal, and there is 1 trough off the diagonal. In the $A \geq 3$ case, this behaviour is more clearly shown with the central band holding A peaks with the first off-diagonal band holding A-1 trough then A-2 peaks for the next band etc. These results are highlighted in Fig. 5.4. As for the orbitals themselves, the number of nodes is directly tied to the orbital itself. The n^{th} orbital will have n nodes which directly affects the orbital's contribution towards the OBDM.

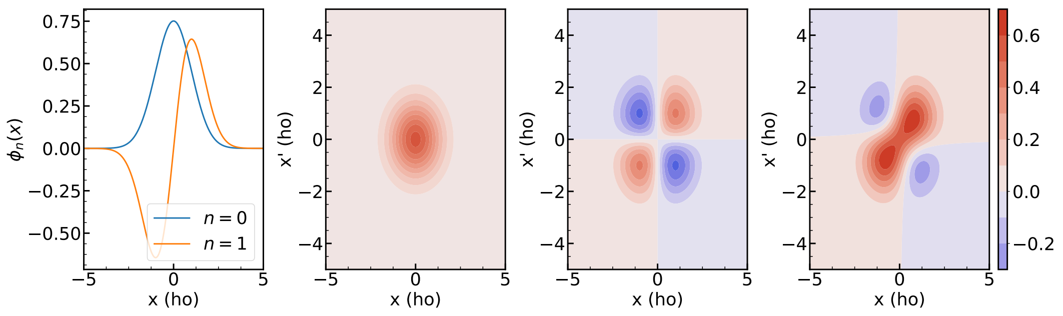


Figure 5.5: The natural orbitals for the ground- and first excited-state of the 2-body 1D Harmonic Oscillator with the associated density matrices of each natural orbital. The one-body density matrix for the ground-state wavefunction of 2 particles is shown in the furthest right panel. This is equal to the sum of the joint amplitudes for the ground- and first-excited states of the physical system.

5.4 Accelerating Convergence via a Novel KFAC Optimiser

The energy convergence results of Sec. 5.1.2 highlight how the architecture of the DNN affect its convergence towards the ground-state wavefunction. However, the convergence of the network is also affected by the use of a given optimiser. Within this section, brief results will be discussed which focuses on a novel optimiser that has been developed as a results of this thesis. This novel optimiser is called QN-KFAC, and is an extension to the existing KFAC optimiser of Ref. [20]. In short, this modifies the KFAC optimiser by including a Quadratic Model (QM) which closely follows the Hessian of the expectation value of the energy with respect to the parameters. In practice, KFAC preconditions gradient descent based on the Fisher Information Matrix (FIM) and yields a new 'preconditioned' gradient which will have a learning rate, α, and momentum constant, μ, in the same way as standard stochastic gradient descent (SGD) as described in Sec. 3.8.1. However, the learning rate needs to be arbitrarily chosen or set in accordance to a learning rate scheduler to perform well in practice [5, 6, 64]. Additionally, the use of a momentum constant works well with first-order optimisers like RMSprop (Sec. 3.8.2) or Adam (Sec. 3.8.3) but in the case of KFAC using momentum actually degradates performance [5, 6, 64].

Therefore, we follow the original methodology of Ref. [20] and utilise a re-scaling technique for the KFAC preconditioned gradient. This re-scaling technique employees a Quadratic Model (QM) to act as an efficient PSD approximate to the Hessian of the expectation value of the energy with respect to the energy. This QM allows for the KFAC preconditioned gradient to be re-scaled within a given region of the parameter space such that it finds an optimal learning rate (and momentum constant) to minimise the loss at the current epoch. This allows for a heuristic approach to defining the optimal learning rate (and momentum constant), and can *significantly* improve convergence over KFAC. In this section we compare Adam, KFAC, and QN-KFAC against each other in solving the ground-state of the non-interacting HO. For Adam the default hyperparameters are used with a learning rate of 10^{-4}. For KFAC, the learning rate is also 10^{-4} with a damping strength $\lambda = 10^4$ and all amortisation constants equal to 5, i.e. $T_1, T_2, T_3 = 5$. In the case of QN-KFAC, it shares all the same hyperparameters as KFAC *except* its learning rate is calculated on-the-fly via a heuristic method. Additionally, QN-KFAC also utilises a form a momentum to further improve its performance which is also heuristically chosen on-the-fly to efficient reduce the loss. A full discussion on the methodology, theory, and pseudocode for QN-KFAC can be found in appendix. B.

A comparison of Adam (blue, Sec. 3.8.3), KFAC (orange), and QN-KFAC (green, this thesis) is shown in Fig. 5.6. In order to highlight the quality pf each optimisers, we perform a weaker pre-training of only 10^3 epochs, and then begin energy minimisation. This reduced pre-training starts the DNN with a higher expectation value for the energy. By showing a comparison between optimisers, we can highlight how by simply including a QM can further accelerate KFAC's convergence. In Fig. 5.6, the relative error of the energy is shown as a function of epochs for all optimisers for A=2 to 6 particles. The results are shown as a log-log plot to emphasise convergence and accuracy of the optimiser. The DNN ansatz for these results is fixed between optimiser with H=64, L=2, and D=1 in order to purely measure the effect of the optimiser. The convergence of Adam is consistently slow across all values of A studied. In the first 10^3 epochs, the convergence of Adam does not differ that much from its initial energy. However, by the order of 10^4

epochs we can see that convergence is starting to take place albeit much slower than all other optimisers. The convergence for Adam is generally quite smooth without any drastic oscillations, except for minimal oscillations for A=3. For KFAC and QN-KFAC, these optimisers are ran an order of magnitude fewer epochs in order to test whether they can convergence to similar or better performance with fewer epochs.

In the case of KFAC, the optimisation begins in the same position within the parameter space as Adam. Therefore, any difference in convergence is due to the different methodology of the optimisers and not a result of initialisation effects. In the A=2 case, KFAC performs as the best optimiser with convergence effectively resolved in only 10^2 epochs. The convergence is initially quite quick with minimal oscillatory behaviour, although, once it has reached the ground-state oscillations occur as it struggles to settle within a local minimum. For QN-KFAC in the A=2 case, similar behaviour is present with generally smooth convergence, albeit slower than KFAC, with QN-KFAC reaching the ground-state within 10^3 epochs. However, it does seem that the use of re-scaling allows QN-KFAC to be less subject to oscillatory behaviour near the ground-state for A=2. As we increase the number of particles, we noticed that the performance of Adam is somewhat consistent, however, in the case of KFAC its improvement in performance is mixed. In the case of A=3, its performance is similar to Adam until $\sim$ 500 epochs wherein its convergence begins to noticeable improve. For A=4, the initial convergence for KFAC is significantly fastest with the loss deviating from Adam with the first 10 epochs and the final error being on the order of 1% by 10^3 epochs. Unfortunately in the case of A=5 and A=6, KFAC's performance only minimally improves over Adam. The relative error of KFAC is approximately the same as Adam, although its about an order of magnitude quicker in doing so.

Now we turn our attention to the work of this thesis, QN-KFAC, in which we effectively take KFAC and find an optimal learning rate (and momentum constant) within a given region of the parameter space. This allows for the update to be used for efficiently and results in improved convergence. A clear example is the A=3 case, KFAC's initial convergence mirrored Adam until $\sim$ 500 epochs wherein a sudden improvement is seen. However, in the case of QN-KFAC the use of the QM allows significantly improved convergence which occurs at $\sim$ 50 epochs. Furthermore, the convergence is strong with minimal oscillatory behaviour until it reaches the ground-state. This results in the final error of QN-KFAC being $\sim 10^{-4}$ which is at least 2 orders of magnitude improved over KFAC, and 4 orders of magnitude over Adam. For the A=4 case, minimal improvement is seen over KFAC with a slightly improved ground-state energy and similar oscillatory behaviour. However, in the cases of A=5 and A=6 we can see significantly improved convergence. The convergence of QN-KFAC begins to differ from KFAC at $\sim$ 30 epochs wherein one can see strong convergence. The performance of KFAC did not greatly differ from Adam here, however, by the use of re-scaling KFAC's updates to a QM, i.e. QN-KFAC, the updates can be significantly improved. This results in the relative error of A=5 and A=6 being $\sim 10^{-4}$ which is approximately 4 orders of magnitude better than KFAC in a similar number of epochs.

To summarise, by utilising a QM to re-scaling our gradients we can significantly improve convergence for solving the many-body Schrödinger equation via VMC methods. The walltime of KFAC to Adam is a factor of ~ 1.7, and QN-KFAC's walltime to Adam is ~ 3.4. Therefore, by approximately doubling the walltime of KFAC we can improve its convergence by a relative error of 4 orders of magnitude for a similar number of epochs. These results open the possibility of further improving VMC methods with DNNs by simply utilising more powerful optimisers *without* changing the architecture of the DNN.

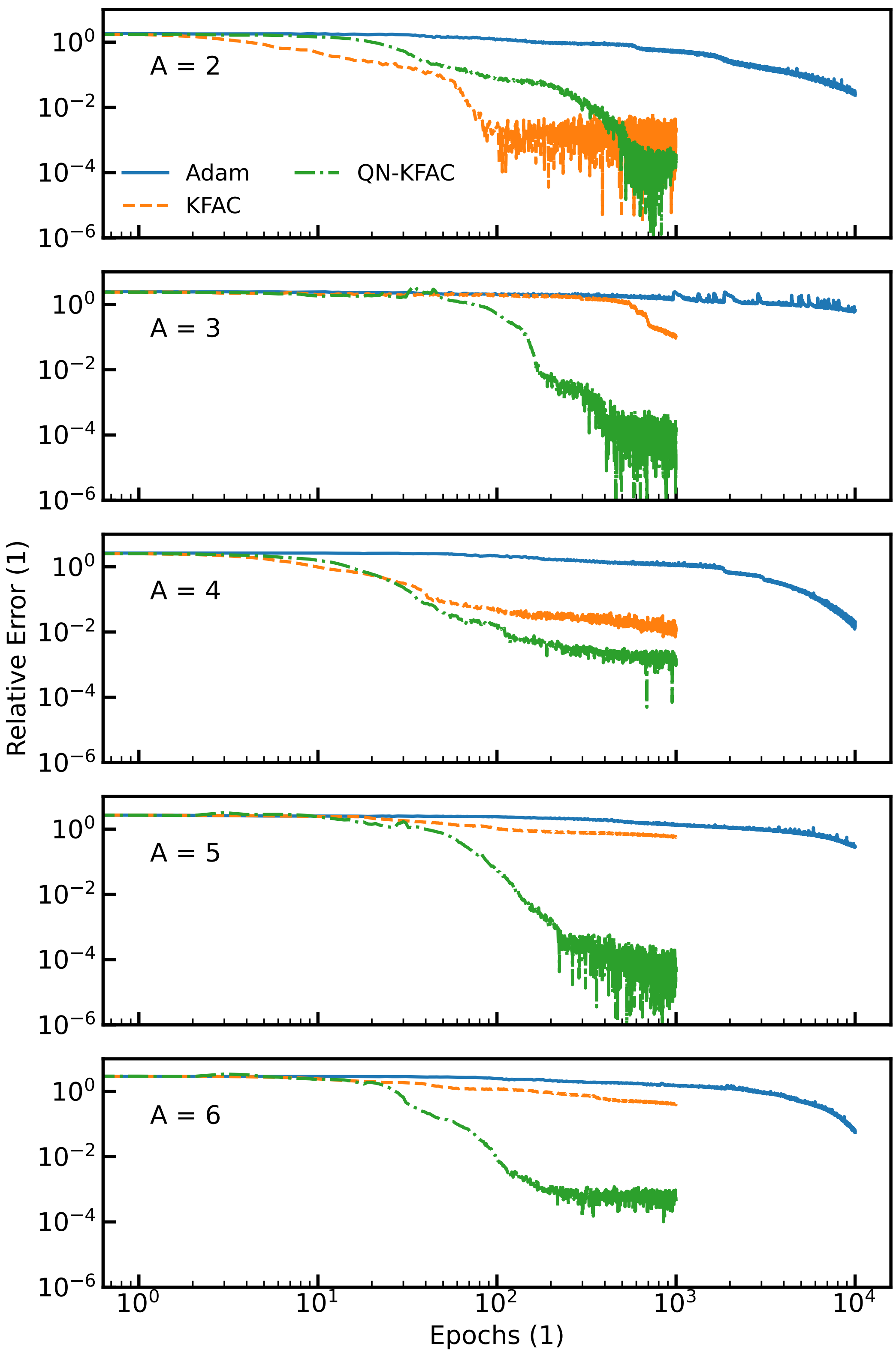


Figure 5.6: A comparison of the Adam (blue), KFAC (orange), and QN-KFAC (green) optimisers for energy convergence for the Harmonic Oscillator (HO) Hamiltonian, for $A = 2$ (top panel) to $A = 6$ (bottom panel). The results are shown as a log-log plot for the relative error of the energy as a function of epochs.

Chapter 6

Interacting Fermions

6.1 Energy Minimisation

Within this chapter we expanded upon the initial results of Chapter 5 towards introducing a finite-range interaction. The Hamiltonian, in ho units, is defined as,

$$\hat{H} = -\frac{1}{2}\sum_i \nabla_i^2 + \frac{1}{2}\sum_i x_i^2 + \frac{V_0}{\sqrt{2\pi}\sigma_0}\sum_{i<j}\exp\left(-\frac{\left(x_i - x_j\right)^2}{2\sigma_0^2}\right), \tag{6.1}$$

where the terms are the kinetic, potential, and interaction respectively. This interaction term denotes a finite-range two-body Gaussian interaction which decays exponentially on the distance $x_i - x_j$ of 2 fermions at positions x_i and x_j with the interaction having a given strength, V_0, and finite-range, σ_0. Both V_0 and σ_0 are varied throughout to test the network's ability to calculate the ground-state wavefunction. When varying the interaction strength, σ_0 is fixed to 0.5 as discrepancies with other methods were maximal here. When varying the finite-range, V_0 is set to ± 20 in order to quantify the performance of the network in both limits of the interaction strength. All units within this chapter are ho units unless stated otherwise.

In the one-dimensional case, bosons and fermions can behave like each other and hold similar properties. This is referred to as the Fermi-Bose duality, and is particularly noticeable in the 'spinless' case which is the setting of this thesis [145]. Within this 'duality', strongly interacting fermions can act like weakly interacting bosons or weakly interacting fermions can act like strongly interacting bosons [146]. As this thesis deals with 'spinless' fermions, comparison will be made with respect to weakly interacting bosons. In most cases, the comparison is made under contact interactions. However, such an interaction is forbidden in the 'spinless' as the PEP forbids particles from having the same position as the spatial component of the wavefunction must be antisymmetric with respect to particle position. This has the physical effect of forbidding s-wave interactions, and hence the interaction is primarily determined by p-wave interactions[1].

For the interaction of this thesis, there exist two distinct physical phenomena for the interaction strength, V_0. These are bosonisation in the limit of $V_0 \mapsto -20$, and Wigner crystallisation in the limit of $V_0 \mapsto +20$ [147, 145, 148]. These phenomena are a core component of this chapter and give physical interpretation to the results discussed throughout this chapter.

6.2 Ground-state Convergence

The ground-state convergence of the DNN ansatz is shown in Fig. 6.1 for a variety of V_0 values. The initial wavefunction is pre-trained to the non-interacting ground-state wavefunction in accordance to Eq. 5.6. The interaction strength is changed from $V_0 = 0$ to $\{-20, -10, 0, +10, +20\}$, and the expectation value of the energy is minimised via the VMC algorithm defined in Sec. 5.1.2. The vANN results for $L = 2$ and $H = 32$, 64, and 128 are shown in Fig. 6.1. The number of parameters of the ansatz ($\sim 8.5 \times 10^3$ for H=64, L=2) remain constant as the interaction strength is changed which highlights the flexibility of the DNN ansatz. The exact ground-state are also computed by a Configuration Interaction (CI) method within a HO basis which is exact within the chosen model space, and shown in a red dashed line in the figure [52]. The CI method cannot be used for large A, but it also struggles for small A in some regimes when the model space is truncated. For this thesis, memory limitations occur in the strongly attractive case (V_0 = -20) where the basis is truncated. This explains why the vANN results for V_0 = -20 are lower than the exact results. The CI results are accurate for $V_0 \geq -10$ for all values of A.

The rate of convergence from the non-interacting ground-state to the interacting ground-state is affected by the interaction strength, V_0. As the interacting strength increases, the convergence towards the interacting ground-state requires more epochs. The analysis for $V_0 = 0$ will be omitted as it was discussed previously in Chp. 5. For the 5 interaction strengths, the network performs well in all cases for all values of A. The initial convergence is strong in all cases with fast convergence towards the ground-state, however, as $V_0 \mapsto -20$ noticeable slow down in the convergence can be seen even though the DNN ansatz significantly improves upon the CI results. In the repulsive regime, the convergence is consistently stronger. Overall, the energy is minimised in all cases of V_0 and A highlighting the robust methodology of DNN. We see strong convergence towards the ground-state across all values of V_0, even in the V_0 = -20 case where CI fails to resolve the exact ground-state. In the repulsive regime there is strong steady convergence, in the attractive regime the convergence is still present albeit slower.

In the $A = 2$ case, the shape of the ground-state wavefunction remains somewhat similar

[1] This physical interpretation for s-wave and p-wave interactions derive from the 3D case, and hence is not *directly* applicable in the 1D case.

in the range of $0 \leq V_0 \leq 20$, and hence the DNN starts near the ground-state and can converge quickly to the interacting ground-state. However, as the number of fermions increase the energy difference between the starting energy and the ground-state energy does increase. The difference between the 'exact' and the vANN results in the strongly attractive case increase as the number of particles increase. There are no exact values for $A = 6$ to compare against so we follow the variational principle of Eq. 2.28, i.e. lower is better.

For $A = 2$, the width of the hidden layers doesn't seem to impact performance at all, except in the non-interacting case, with the initial energies starting close to one another regardless of V_0. As A increases to 6 some discrepancies emerge for the starting energy indicating that the pre-training for larger values of A finish at lower values of $\mathcal{L}^{Pre}(\theta)$, and hence start with lower energies. In the limit of $A = 6$, wider DNNs ($H = 128$) seem to provide better rates of convergence for the expectation value of the energy by minimising more efficiently than their narrower counterparts, however, for $A = 2$ the opposite behaviour is present most likely due to a bias-variance trade-off. For $V_0 \geq 0$, the convergence reaches the ground-state by 10^4 epochs with the remaining epochs refining the expectation value of the energy by reducing the variance. For $V_0 < 0$, the convergence is slower with more oscillatory behaviour.

This could possibly be a result of the PEP, as the interaction strength becomes more attractive the average distance between fermions decrease, however, the PEP enforces that fermions never overlap exactly. Therefore, when the DNN ansatz minimises the energy and forces the fermions closer together the loss surface *might* become more oscillatory which affects efficient convergence. As the fermions are forced together they minimise their potential energy, but their kinetic energy increases. If the functional form of the wavefunction were to change drastically, this would result in a noticeable change to the kinetic energy which in turn will lead to oscillatory behaviour in the convergence. For the repulsive convergence, it remains steady and smooth whereas the attractive case seems to hold more oscillatory behaviour.

Overall, the performance of the DNN depends on both its width and depth, but more so its depth (from Chp. 5). The increased depth seems vital in achieving the ground-state energies, however, the width seems to play a key role in ensuring quicker, more stable convergence. Furthermore, as number of particles increase the DNN ansatz performs better, as measured by the ground-state energy, with wider networks when the depth is fixed. These results indicate a form of bias-variance trade-off for this physical system where system with fewer particles perform better with narrower DNN ansätze (H=32) whereas systems with more particles prefer wider DNN ansätze (H=128). Therefore, we seek a compromise between narrow and wide networks by setting H=64 (with L=2) for all values of A and V_0.

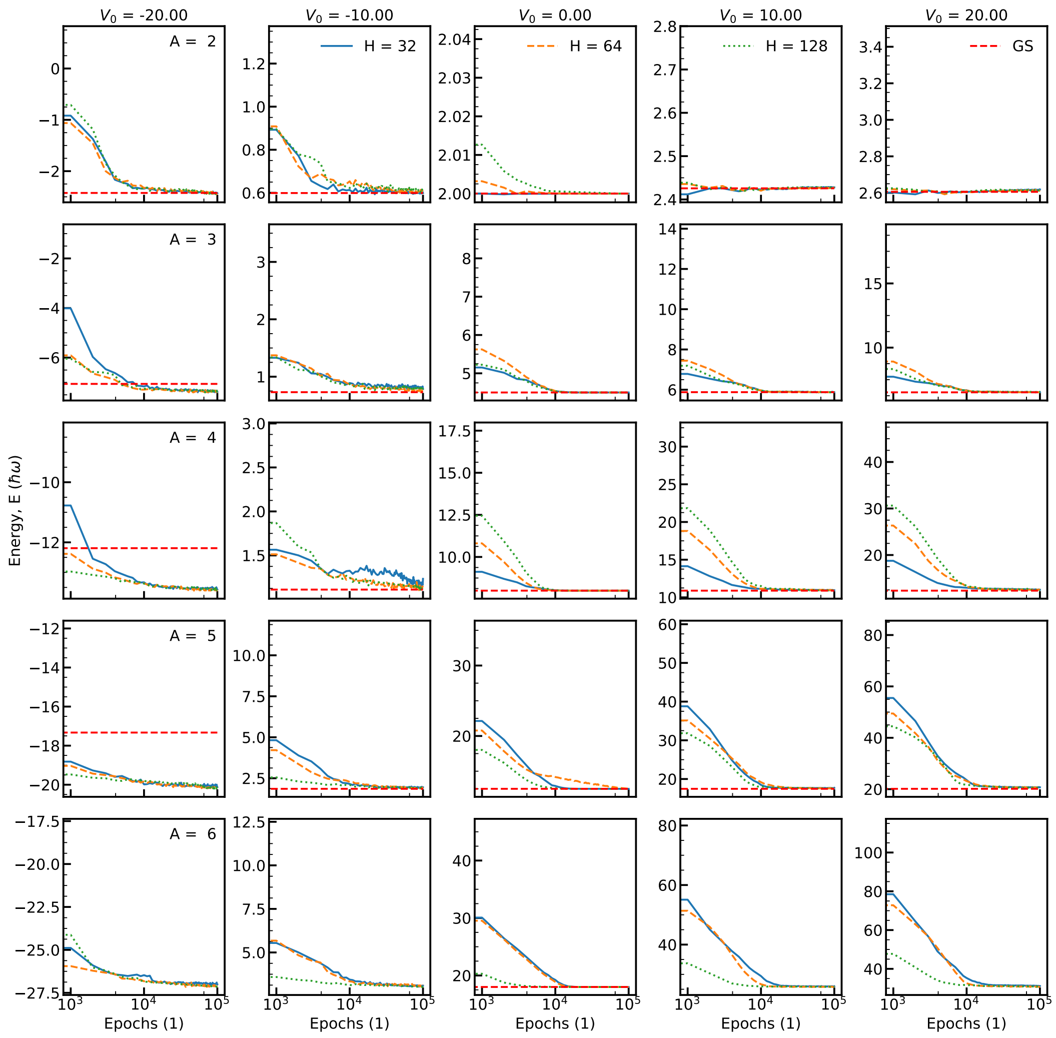


Figure 6.1: The convergence of the DNN ansatz for the expectation value of the energy for varying hidden width, H, and number of layers, L, for a single generalised Slater determinant, $D = 1$, as a function of epochs.

6.3 Varying interaction strength

The interaction strength, V_0, is varied between -20 to +20 at a fixed finite-range of $\sigma_0 = 0.5$ to quantify how effective the DNN is at representing the wavefunction within different interacting regimes. In the limit of these two regimes, two different physical phenomena emerge.

These vANN results are benchmarked against three other numerical methods; Hartree-

Fock (HF); direct diagonalisation (Space); and the diagonalisation method of Ref. [52]. The HF method is a mean-field approach for the Schrödinger equation via an effective one-body interaction as discussed in Sec. 2.6. The direct diagonalisation in real space is separated into centre of mass (COM) and relative terms where the relative wavefunction is solved within a discretised mesh. However, this method is only feasible for $A = 2$. Therefore, we result to the CI method (hereafter referred to as Diag) which is based on the methodology of Ref. [52] in which CI is performed within a HO basis with a fixed number of modes. The number of modes were fixed to 20 which were sufficient for most values of V_0, although, in the strongly attractive regime this became insufficient. The number of nodes were fixed due to memory limitations. All three benchmark methods are compared against the results of this thesis, vANN. The methods HF and Diag, like vANN, are variationally bound and provide an upper bound on the expectation value of the ground-state. The 'space' benchmark is an exact methodology used to provide a highly accurate result in the $A = 2$ case.

The interaction strength, V_0, is varied and the ground-state energy is variationally found with the respective methodology. These results are shown in Fig. 6.2 for all numerical methods with a grey dashed line to represent the non-interacting ground-state. The behaviour of the ground-state energy is monotonic in V_0, i.e. as $V_0 = -20 \mapsto +20$ the ground-state energy increases continuously. In the repulsive limit the ground-state energy plateaus towards a maximum value whereas in the attractive limit it begins to decay linearly with respect to the interaction strength. This behaviour is shown across all numerical methods and all values of A.

In the 2 fermions (top panel) case we compare all numerical methods whereas for $A \geq 3$ we restrict the analysis to all but the Space numerical method. The poorest performing method for $A = 2$ is HF which is to be expected. As HF is a mean-field approach it will become more accurate as the number of particles increase wherein the interactions between particles become 'averaged' out. For A=2, the relative error of HF against the exact results is from 1% to 100% with the error dominating within the strongly attractive regime. The Diag method is, within a given basis, an exact numerical method at solving the Schrödinger equation which results in a lower variational bound on the energy. The improvement of the Diag method over HF is not only to be expected but is consistent across values of V_0. Interestingly enough, one can see a small discrepancy in the ground-state energy in the strongly attractive regime against the aforementioned 'Diag' method when compared with the 'Space' method which emerges from the truncated basis. The Diag method will be used as the 'exact' baseline for all $A > 2$ results. In general this method computes an accurate ground-state energy that agrees well with the vANN results for $-10 \leq V_0 \leq +20$. For $V_0 \leq -10$, the effects of the truncated basis for the Diag method begin to show and the vANN method reaches lower expectation values for the energy indicating an improvement of using the DNN ansatz. This discrepancy is negligible for $A = 2$, however, becomes more prominent as A increases for $V_0 \leq$ -10.

The HF method initially performed poorly for A=2 gradually improves its ground-state

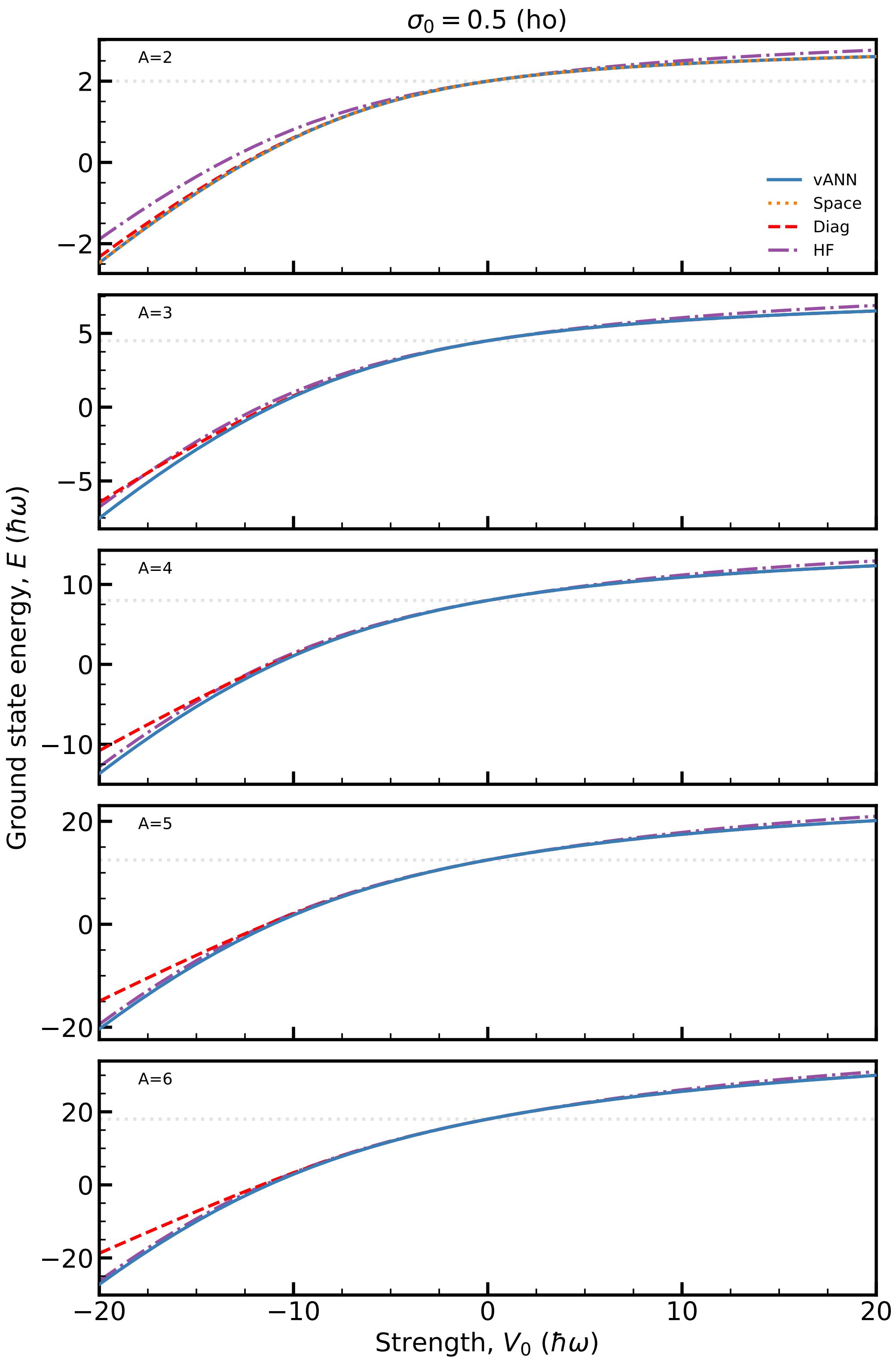


Figure 6.2: The ground-state energies for both extremes of the interaction strength, V_0, for $\sigma_0 = 0.5$. The results of the network are compared against numerous numerical methodologies (see legend) to highlight its competitiveness.

energies upon the Diag methodology solely within the attractive regime. This behaviour becomes clear at A=4 where in the strongly attractive limit, a clear improvement of HF over Diag is seen, however the vANN results are still the superior results. As HF is a purely mean-field approach, the bias of this method denotes the missing 'correlation energy' between the fermions which all other methodologies explicitly incorporate. As the number of particles increases, the correlation energy as a fraction of the total ground-state decreases which allows for a noticeable improvement as the number of particles increase. For interaction strengths of $|V_0| \leq 10$, the HF results perform very well at approximating the exact ground-state, except in the case of A=2 where it only approximate the ground-state energy well for $|V_0| \leq 2$.

In both extremes, $|V_0| \geq 10$, we can see that the vANN results are slightly lower than the HF results which indicates that as one approaches either extremes there exists further correlation energy beyond the mean field. In the strongly repulsive limit, the Diag and vANN results give near identical values with HF results residing slightly above. This indicates discrepancies in this regime are purely beyond mean-field effects. In the strongly attractive regime, the Diag results break down due to a truncated basis and the HF results reside just above the vANN results which indicates that further correlation energy exists too.

6.4 Varying interaction range

The finite-range of the Gaussian interaction, σ_0, is varied between 0.1 and 3 with the limit of 0.1 being approximately a contact interaction [52]. The contact interaction is omitted here because in our use case it gives no contribution towards the energy. As σ_0 approaches 0, the Gaussian interaction tends to a Dirac-delta interaction which only has a contribution to the energy if two fermions have the same position. This thesis deals with 'spinless' fermions, and so the only degree of freedom for the fermions is the position, i.e. we neglect spin. However, an antisymmetric function of purely position cannot have two particles at the same position which therefore results in the fermions not feeling a zero-range interaction. In order to circumvent this, we default to a finite-range interaction which in the limit of $\sigma_0 \mapsto 0$ equals a contact interaction.

The many-body positions are sampled via the stochastic process of a Markov chain, and within this process the probability is used to determine if a proposed move is accepted. If a walker is moved to either a node or the many-body position of another walker then the probability of such a position is zero and hence rejected by the MH sampler. The results of the ground-state energies for a range of σ_0 are shown in Fig. 6.3 for $V_0 = \pm 20$. The results shown contain Perturbation Theory of a given order X (PTX), the mean-field approach of Hartree-Fock (HF), the exact diagonalisation (Space, n=0), and the work of this thesis - neural networks (vANN). The PTX results are only shown for A=2 whereas all other methodologies are shown for all values of A. The results are shown for a wide range of particles from A=2 up to A=6, however, we only compare HF to vANN for A>2.

For 2 fermions we can compare all numerical methods. Firstly, the established baseline of diagonalisation (shown in orange) allows us to compare all of methods (especially vANN) to an 'exact' value. The perturbation theory results shows poor performance in reproducing the exact baseline, although, they do improve by increasing the order of the perturbation theory as expected. The results shows that PT (shown in green) only reproduces longer range interactions on the order of $\sigma_0 > 1$ with the minimum σ_0 at which discrepancies arise increasing as the order of PT decreases, i.e. for PT1 discrepancies emerge at $\sigma_0 \approx 3$ whereas for PT3 discrepancies emerges at 1.75. In the attractive regime (top left panel), for $\sigma_0 < 0.5$ PT3 gives a similar rapid increase in the ground-state energy, although, there is a noticeable increase from the exact baseline. In the repulsive regime, the difference is more oscillatory and only matches the exact baseline for $\sigma_0 < 0.2$ and for $\sigma_0 > 1.7$ there is excellent agreement. The methodology of PT is not variationally bound like HF, Space, or vANN which explains how PT can predict ground-state values below the exact Space results. This preliminary analysis for PT allows one to see which regions of σ_0 are perturbative or not. This motivated fixing $\sigma_0 = 0.5$ for when the interaction strength is varied as the results within this regime are non-perturbative, and hence act as an ideal benchmark for this system in order to fundamentally test the DNN ansatz in Sec. 6.3.

The next method is Hartree-Fock (HF) whose agreement between HF and the baseline is excellent for $\sigma_0 > 1.7$ ho, and for $\sigma_0 < 1.7$ ho the ground-state energy holds a linear structure as a function of σ_0 and is a uniform upper bound on the exact baseline. This implies that for such distance of σ_0 there exists further correlation than what solely a mean field can define. And, in the limit of $\sigma_0 = 0$ HF converges to the same value as the non-interacting case which is shown in each subfigure of Fig. 6.3 as a fainted grey dashed line. This happens because at $\sigma_0 \mapsto 0$ and $\sigma_0 \mapsto \infty$, the two-body interaction tends to a contact interaction and no interaction at all respectively. This has the effect of taking the interacting Hamiltonian and turning it in the non-interacting Hamiltonian in these limits of σ_0. For reference, the relative error for HF varied on the order of 10^0 to 10^{-6} with the largest relative error occurring at the smallest values of σ_0 and decreasing as the finite-range is increased.

In the case of V_0 = -20, the relative error for HF peaks at $\sigma_0 = 0.2$ with the error on the order of 10^0 whereas the vANN results have a maximum relative error of 10—4 at $\sigma_0 = 0.1$. In the case of $V_0 = +20$, the maximum relative error occurred at $\sigma_0 = 0.4$. In both case, the minimum relative error of HF occurs at the maximum range of $\sigma_0 = 3$. This highlights that as the finite-range decreases, i.e. $\sigma_0 \mapsto 0$, additional correlation energy is represented in the ground-state wavefunction. However, in the exact limit of $\sigma_0 = 0$ there is no correlation energy from the two-body interaction as it becomes a contact interaction which does not contribute towards the expectation value of the energy for 'spinless' fermions [148].

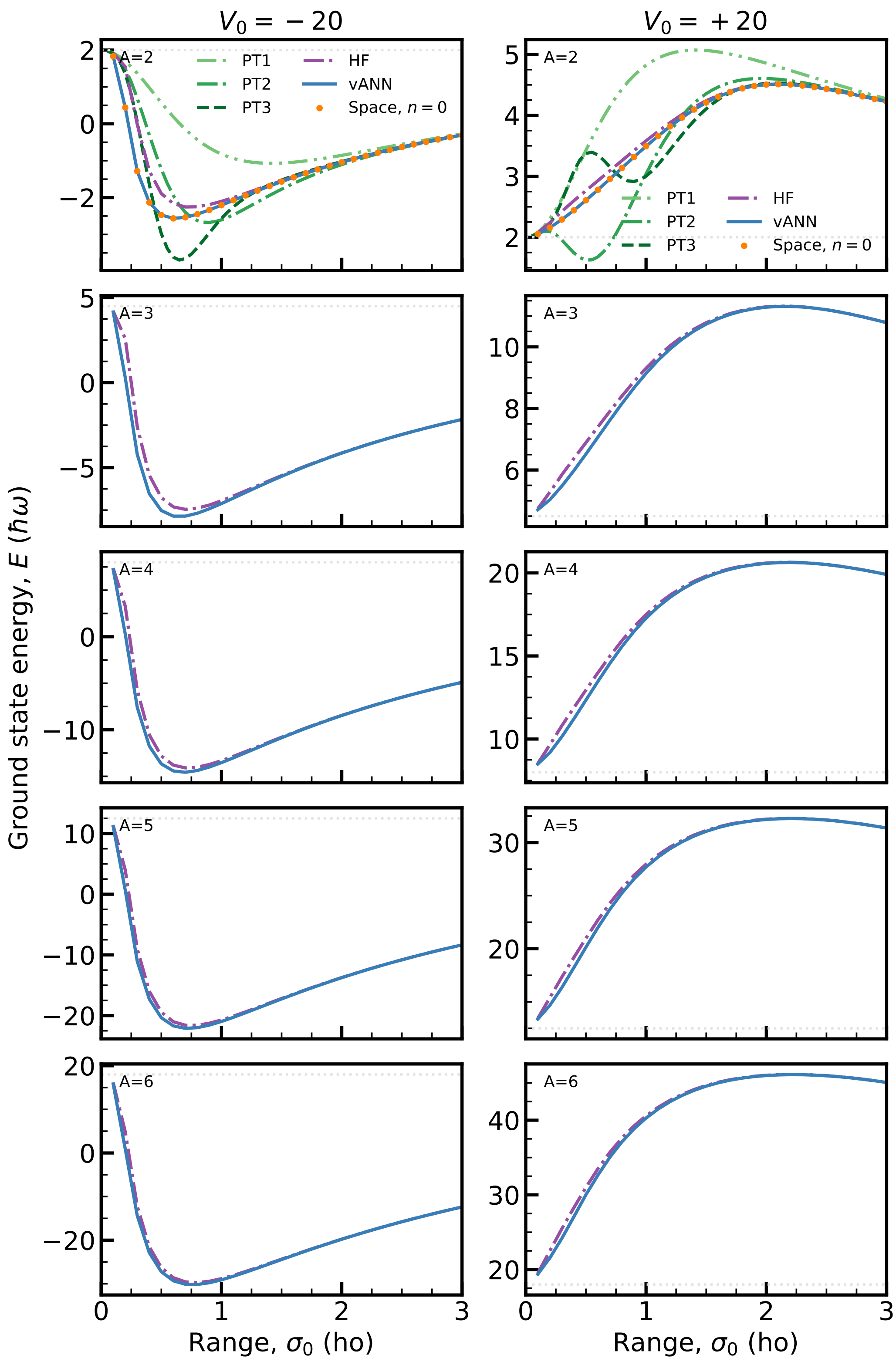


Figure 6.3: The ground-state energies for both extremes of the interaction strength, V_0, for a range of σ_0 values. The range of σ_0 lies in $[0.1, 3.0]$ in intervals of 0.1. The results of the network are compared against numerous numerical methodologies to highlight its competitiveness.

The final method to discuss for two fermions is the vANN method of representing the wavefunction as a DNN. The top panels of Fig. 6.3 show strong agreement between the vANN results (blue) and the space benchmark (orange) for all values of σ_0 for $V_0 = \pm 20$. The ability for the vANN method to reproduce the baseline across all values of σ_0 shows not only a robustness to the methodology but also a compressibility of a neural network ansatz in representing the complexity of the wavefunction. The relative error of the vANN results to the exact results vary on the order of 10^{-4} to 10^{-6}. For V_0 = -20 and $\sigma_0 < 0.3$ ho the relative error is on the order of 10^{-4} with that decreasing as σ_0 increases. For $\sigma_0 \geq 0.3$ ho the relative error decreases on the order of 10^{-5} to 10^{-6} which follows similar performance to HF in that also the relative error improves for larger σ_0. For V_0 = +20 the relative error peaks at $\sigma_0 = 0.2$ on the order of 10^{-5} and holds a relative error on the order of 10^{-6} for $\sigma_0 \geq 0.2$. The consistency of the relative error for both extreme cases of interaction strength for a wide range of interaction range indicates the methodology is robust and accurate at representing many-body wavefunctions.

As can be seen for $A \geq 2$ with V_0 = -20, HF and vANN hold similar performance for $\sigma_0 >$ 1 with noticeable discrepancies for $\sigma_0 < 1$. It can be seen in Fig. 6.3 that the vANN results (blue) are below the HF results (purple) indicating that the neural network ansatz can represent higher order correlations beyond the mean field approach which dominate for smaller interaction ranges of $\sigma_0 < 1$. The same applies for the V_0 = +20, although, the discrepancies between HF and vANN appear at larger ranges of $\sigma_0 > 1.5$ with again the vANN results improving upon the mean field approach.

6.5 Pairing Gap

In the case of nuclei, protons and neutrons are the constituent particles of the nucleus. In nature there are only 4 stable nuclei that have both an odd number of protons and an odd number of neutrons with the remaining being odd-even, even-odd, and even-even nuclei [50]. The 'pairing' effect emerges from the tendency of particles of the same spin to increase the binding energy of the system which has the effect of reducing the ground-state energy further. In the case of this thesis, we deal with 'spinless' fermions and so we only have *one* type of nucleon. The 'pairing-gap' denotes the difference in energy between a system of A particles and the average energy of $A+1$ and $A-1$ particles. If these two values were to be equal to one another then the fermions would not have any pairing. The formula for this pairing gap is defined as,

$$\Delta_A = \langle E_A \rangle - \frac{1}{2}\left(\langle E_{A+1} \rangle + \langle E_{A-1} \rangle\right). \tag{6.2}$$

where Δ_A is the pairing-gap, and $\langle E_A \rangle$ is the ground-state energy with A denoting number of fermions. The results are shown in Fig. 6.4 for A = 2 to 5 particles for V_0 from -20 to 20. The physical interpretation from Eq. 6.2 is that if Δ_A is negative then the ground-state energy for A particles is smaller than the average of $A-1$ and $A+1$ particles.

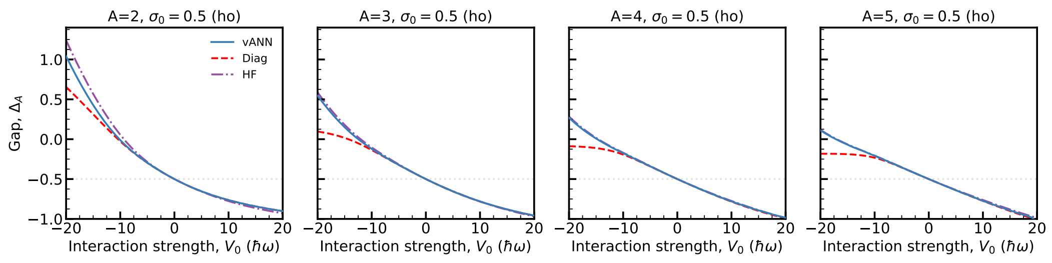


Figure 6.4: The pairing gap energy for A=2 to A=5 particles for a wide range of interaction strength, V_0 at the finite-range of $\sigma_0 = 0.5$.

This indicates that pairing has further reduced the ground-state energy. If Δ_A were to be positive, then this indicates that pairing has increased the ground-state energy. The non-interacting case (dashed grey line) acts as a baseline to highlight how varying the interaction strength affects the amount of pairing-gap is present within the Hamiltonian. This value is determined by solving for the ground-state energy for $V_0 = 0$ and computing the pairing-gap, Δ_A, in accordance to Eq. 6.2. In general the pairing effect helps lower the ground-state energy of an A fermion system by increasing the binding energy. This can be seen in the non-interacting ($V_0 = 0$) case where the pairing gap, Δ_A is always negative regardless of the number of particles. However, by varying the interaction strength from V_0 = -20 to +20 we can see how the pairing effect can increase or decrease the ground-state energy of an A fermion system. In fact, this property of the system can also be used as a benchmark towards other numerical techniques. For this property we benchmark our vANN numerical results against our 'exact' results of the Diag method of Ref. [52], and the mean-field approach of HF.

In the repulsive regime all three numerical methods have excellent agreement which compare well with the ground-state energies of Fig. 6.2. However, in the attractive regime differences emerge between the numerical methods and these differences become more profound as the interaction strength becomes more attractive. For the $A = 2$ case, all three numerical methods give different results for the pairing-gap. As the pairing-gap is the difference between expectation values it does not abide by a variational principle, hence, a lower value of Δ_A doesn't correspond to a superior numerical method. However, the pairing gap energies emerge from the ground-state energies which *do* abide by the variational principle. Therefore, it stands that the results of the vANN methodology holds more accurate results as the associated ground-state values for the energy are lower with this method when compared against all other methods. The HF method overestimates the pairing-gap indicating that in the attractive regime the mean-field approach overestimates the increase in the ground-state energy due to pairing. The Diag method again predicts that the ground-state energy increases due to pairing but it underestimates the increase in the pairing-gap. As the number of fermions increase the agreement between vANN and HF improve and in the limit of $A = 5$ results in a linear relationship between Δ_A and V_0. The pairing-gap predicted by the Diag method saturates as the interaction strength becomes more attractive and eventually plateaus towards a maximum contribution. The mismatch of the Diag results with both HF and vANN most likely emerges due to the lack of modes used in the calculations which impacts performance across all properties of the physical system and not just the pairing-gap [52].

The general trend of the pairing-gap is that as A increases, Δ_A with respect to V_0 becomes more linear. If the trend were to continue for higher values of A, one would expect the pairing-gap to become negative in the limits of the strongly attractive regime which has the physical interpretation of the pairing effect reducing the ground-state energy.

6.6 Individual contributions to the energy

The ground-state energy, $\langle E \rangle$, of the Hamiltonian is comprised of three components; kinetic energy, $\langle T \rangle$; potential energy, $\langle V \rangle$; and interaction energy, $\langle V_{int} \rangle$. As the interaction strength is varied different physical phenomena emerge that can be explained by comparing each component of the ground-state energy to each other.

The Wigner crystal emerges in the strongly repulsive limit as $V_0 \mapsto 20$ [147]. Within this limit the fermions try to maximise their distance from each other as the interaction term forces the fermions apart. This can be seen in Fig. 6.5 shown in red where the contribution of the interaction term of the Hamiltonian increases as V_0 increases. The OBD distribution as $V_0 \mapsto 20$ leads to a fixed lattice structure which reduces the available positions, x, for the fermions to reside [147]. This has the effect of reducing the kinetic contribution as the fermions are effectively fixed in-place (shown in dashed yellow). The potential energy contribution increases as $V_0 \mapsto 20$ as the density of particles begin to push against the HO trap. This is a key property of Wigner crystals where the potential (as well as the interaction) energy contribution dominate over the kinetic energy which 'fixes' the fermions in-place leading to a crystallines structure [149].

In the strongly attractive limit of $V_0 \mapsto -20$ the fermions undergo a process of bosonisation. In this limit all fermions become centred at the origin of the HO well and form a single peak in the OBD distribution. This leads to the kinetic energy contribution to increases as all fermions try to avoid each other due to the PEP. As the particles are located at the centre of the HO well, the potential energy contribution tends to zero. The contribution from the interaction becomes increasingly attractive as the interaction strength tends to $V_0 \mapsto -20$ with the asymptotics of its contribution towards the energy being linear with interaction strength. The interaction contribution dominates over the kinetic energy term and confines the particles towards the centre of the HO well which results in the near-zero potential energy contribution.

As the number of particles increase the overall behaviour of each component of the ground-state energy remains consistent, i.e. the kinetic contribution dominates over potential in the strongly attractive regime and vice versa in the strongly repulsive regime. The only key difference is that as the number of particles increase the magnitude of the kinetic, potential, and interaction energy contributions increase in magnitude as well. This leads to the ground-state energy to increase in magnitude as the number of particles increase.

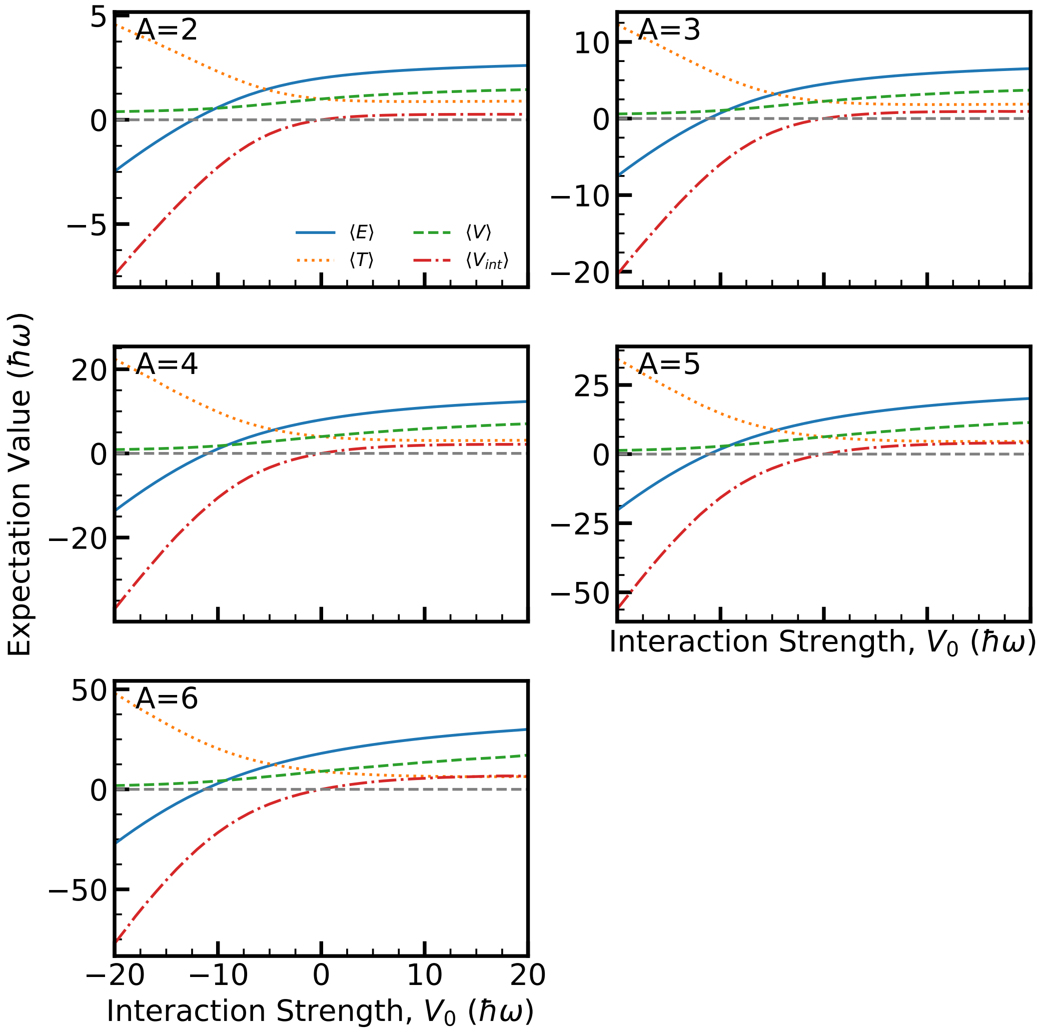


Figure 6.5: The ground-state energy (solid blue) with the kinetic (dotted orange), potential (dashed green), and interaction (dashed-dotted red) contributions for A=2 to A=6 particles for a wide range of interaction strength, V_0 at the finite-range of $\sigma_0 = 0.5$. This shows how the individual components of the ground-state energy varies against interaction strength.

6.7 One-body density

One key property in visualising the ground-state wavefunction is the one-body density (OBD) distribution which is defined by integrating the Born probability of the many-body wavefunction over all but one spatial degrees of freedom, or by integrating over all spatial degrees of freedom with a Dirac-delta function applied to the first spatial coordinate (Eq. 5.14). The one-body density can reveal the internal structure of a many-body wavefunction. The oscillations of the OBDM (Eq. 5.15) emerge from the fermionic nature of the Hamiltonian, however, the OBD is purely positive semi-definite (PSD).

The leftmost column of Fig. 6.6 shows the density in the strongly attractive limit, $V_0 =$ -20. The OBD forms a single peak centred at the origin. This behaviour holds close similarity to a non-interacting bosonic system in which the particles sit side-by-side forming a single peak in the OBD [146, 145, 150]. This physical phenomena is referred to as 'bosonisation' as the strongly attractive interacting fermions have physical properties that mirror an equivalent non-interacting bosonic system [151].

In the rightmost column of Fig. 6.6, we show the density in the strongly repulsive limit wherein a different physical phenomenon is present. As the interaction strength pushes the fermions further apart, the fermions have to localise in order to minimise their energy. This leads to the physical phenomena of 'crystallisation' in which the OBD is multi-mode distribution of equidistant peaks [149, 147]. The results of Fig. 6.6 are performed from 2 to 6 fermions, and follow the same form as Fig. 6.3 with the vANN results compared against the Diag method of Ref. [52] and the mean field approach of HF. In the case of 2 fermions we compare against exact diagonalisation (Space) too. In the non-interacting case we can see that all methods give consistent results. For all values of V_0 shown for 2 fermions (top row), the exact results align excellently with the vANN results. For the repulsive interaction values the exact and vANN results are virtually identical with the Diag results. However, the mean field approach of HF begins to fail as the interaction strength becomes more repulsive. The asymptotic behaviour of the one-body density in this limit is correct but the central behaviour of the one-body density falls short. The mismatch between HF and Space diagonalisation occurs at the peaks (troughs) of the OBD where the predicted density is higher (lower) than the exact OBD. This has the physical interpretation of over-predicting particle densities at the peaks and under-predicting at the troughs. The discrepancies of the OBD between the vANN and Space results do differ based on V_0. The OBD for $A = 2$ and $V_0 < 0$ with HF gives a stronger centralised peak at the origin with the asymptotic of the open boundary conditions beginning at smaller distances from the origin which results in a sharper peak. In general, the mismatch in the central peak becomes much more mitigated for the HF approach as A increases due to HF performing better with larger A, however, the Diag methodology becomes much more exacerbated. In the case of A=2, the Diag results match quite closely with the exact Space diagonalisation results yet as the number of particles increases the peak of OBD becomes smaller in magnitude and noticeably broader which significantly impacts the

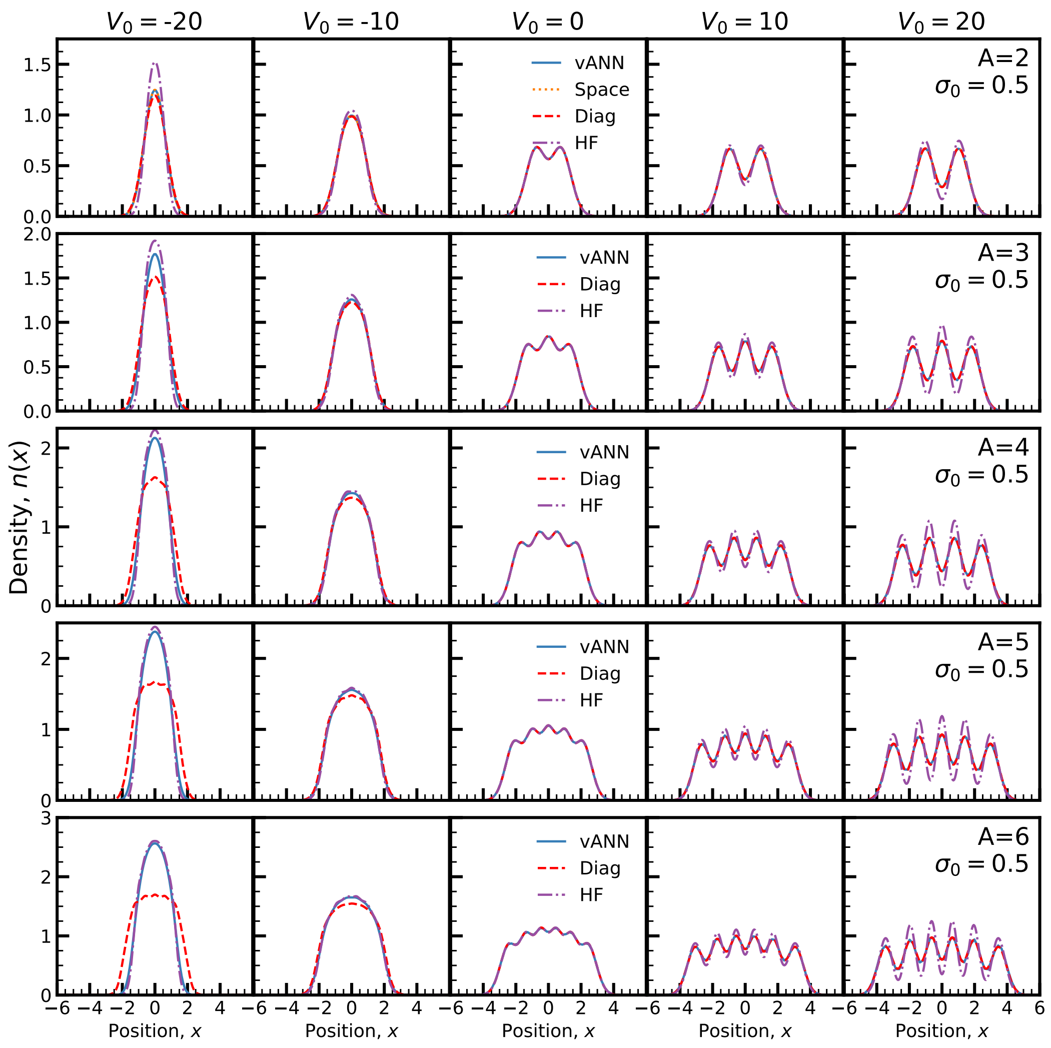


Figure 6.6: The one-body density distribution plots for A=2 to A=6 particles for a wide range of interaction strength, V_0 at the finite-range of $\sigma_0 = 0.5$. This shows how the one-body density distribution varies against interaction strength from a bosonic gas at $V_0 =$ -20 to a Wigner crystal at $V_0 = +20$ for a variety of numerical methods.

performance of the Diag methodology.

As we increase the number of particles to A > 2 these effects become more profound. In the strongly repulsive limit the peaks and troughs of the one-body density become sharper which is physically interpreted as the particles being more equally spaced from one another [147]. The vANN and Diag results gave better predictions for the ground-state energy so their OBD distributions should be superior in quality. In the strongly attractive limit the difference between the exact Diag results, and the vANN and HF results is more clear. Firstly, the difference between the Diag and vANN results mainly differs in the density of the central peak as the Diag results severely undercount the true one-body density. This discrepancy of the central peak becomes more evident as the number of particles increases. In the case of V_0 = -10, the asymptotics of the OBD, as $|x| \mapsto \infty$, are in general agreement for all three methodogloies. However, as the interaction strength becomes more attractive to V_0 = -20 the asymptotics clearly begin to differ. It is more subtle between the vANN and HF results, but the clear differences is for the Diag results which begin at a larger distance and decay more sharply. In fact, this behaviour is difficult to represent with the 20 modes of Ref. [52] because it may be difficult to capture of the structure of an object that is much narrower than the average size of the HO basis. This leads to the properties of the ground-state OBD to differ from the true ground-state OBD. This is clear seen by viewing the OBD of the Diag method as the interaction strength varies from $V_0 = 20 \mapsto$ -20 we can see that the results clearly breakdown and fail to resolve the correct OBD distribution. This deficit in accurately representing the OBD within the strongly attractive regime emerges due to the aforementioned lack of modes within the methodology [52].

6.8 Root-mean-square Size

The size of the physical system can be calculated via the root-mean-square (RMS) which is defined as,

$$\text{RMS} = \sqrt{\langle x^2 \rangle} = \sqrt{\frac{\int_{-\infty}^{\infty} x^2 \rho(x)\, dx}{\int_{-\infty}^{\infty} \rho(x)\, dx}}. \tag{6.3}$$

where $\rho(x)$ is the one-body density distribution. The results are compared against the 'exact' Diag method of Ref. [52] and the mean-field approach of HF [9]. In the $A = 2$ case, we also compare with exact diagonalisation (Space). The vANN results are benchmarked against all numerical methods. The functional form of the RMS against V_0 is somewhat sigmoidal in nature with the size plateauing at both extremes of interaction strength with a monotonic increase as V_0 increases. This behaviour is to be expected as in the strongly attractive limit the PEP will enforce a minimum size, and in the strongly repulsive limit the HO potential will enforce a maximum size of the system. This monotonic shape of the RMS is consist for values of A.

In the A=2 case, the vANN results hold superb agreement with the space diagonalisation methodology across all values of V_0 studied with near negligible differences in results. Additionally, the Diag results align excellently with the vANN results until the Diag methodology fails near $V_0 \approx$ -15 due to the truncated model space [52]. In this limit, the RMS size of the physical plateaus to a constant value regardless of the interaction strength [52].

Furthermore, the mean-field approach of HF holds excellent agreement for $|V_0| \leq 5$, outside this range there is an asymmetrical discrepancy between HF and exact results. In the strongly repulsive limit, the discrepancy between HF and the exact results is minimal indicating that a mean-field approach appears to encapsulate most of the RMS size with the remaining discrepancy emerging from additional correlation from beyond the mean-field approach. However, in the strongly attractive limit a clear divergence between the numerical methods emerges. The mean-field approach severely underestimates the size of the physical system as the interaction strength becomes more attractive. The discrepancy between the vANN results and HF increases as $V_0 \mapsto -20$, however, as the number of particles increases the two methodologies alignment closer. This behaviour is expected as a mean-field approach will becomes more realistic as more particles generate a mean-field interaction. Overall, HF routinely underestimates the size of the physical system in the strongly attractive limit.

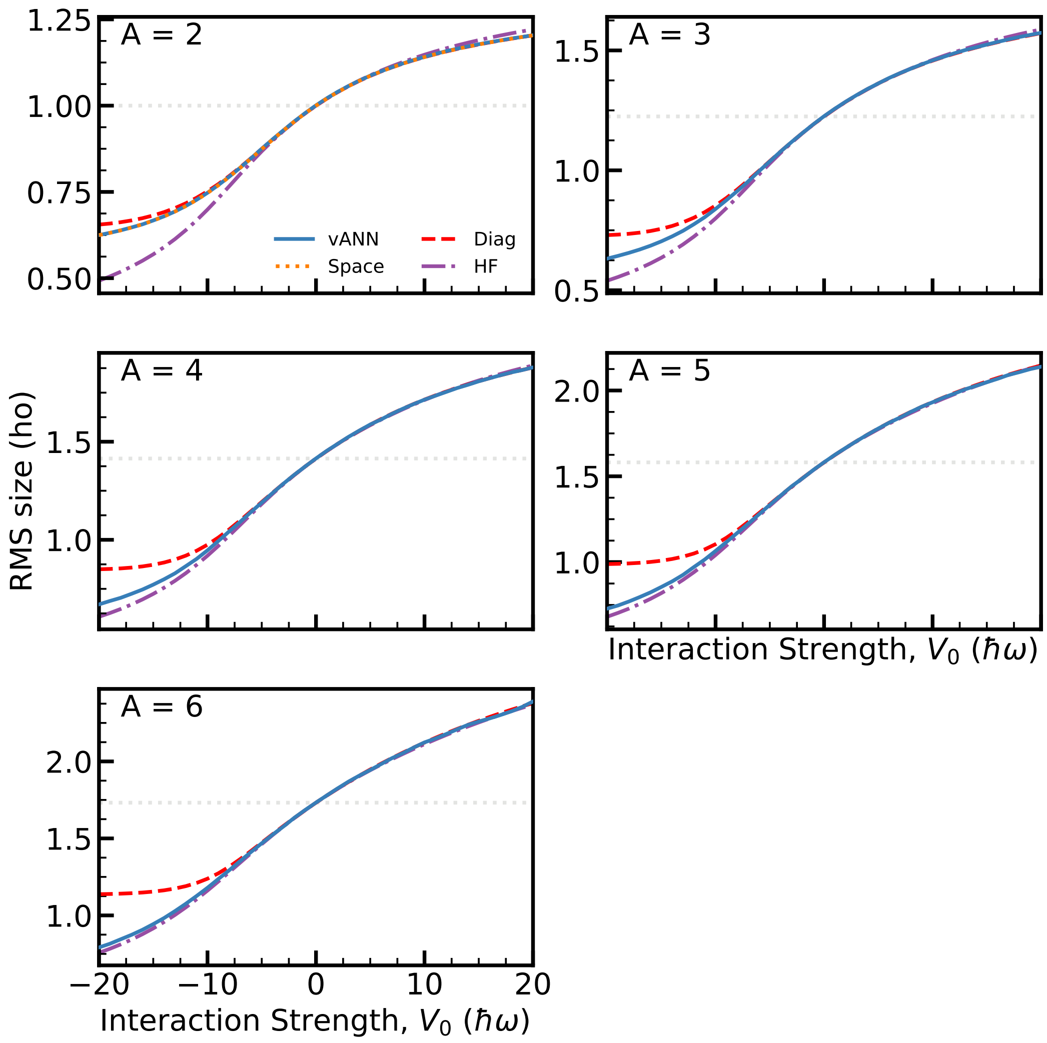


Figure 6.7: The Root-Mean Square (RMS) size for A=2 to A=6 particles for a wide range of interaction strength, V_0 at the finite-range of $\sigma_0 = 0.5$. This shows how the RMS size varies against interaction strength for a variety of numerical methods.

6.9 One-body density matrix

A natural extension of the one-body density is the one-body density matrix which requires integrating the wavefunction over all but two spatial degrees of freedom (see Sec. 5.3, Eq. 5.15). The one-body density matrix is a Hermitian matrix which represents the mixture of single-particle orbitals, ϕ, within the wavefunction, Ψ. The one-body density matrix can give quite an insight into the internal structure of the wavefunction. Within

Fig. 6.8 we can see the one-body density matrices for the Harmonic Oscillator plus finite-range interaction of a wide range of strength, V_0, for 2 to 6 fermions.

In the non-interacting case we can see that there exists a ripple-like effect between positive correlation (red) and negative correlation (blue) which oscillates in the direction orthogonal to the diagonal. This is the same ripple-like effect from Chp. 5, the density of the ripples can be understood in terms of the widths of the single-particle states. As the interaction strength becomes more attractive, the ripples becomes more dense. On the other hand in the repulsive limit, the ripples become more spread out. As mentioned in the non-interacting case, the number of bands that lie off the diagonal equals $A-1$ where A is the number of particles with each successive band holding one less peak/trough. For example, in the case of $A=3$ the central band has 3 peaks with 2 (3-1) bands off the diagonal. The first band has 2 (3-1) clear troughs with the next successive band holding 1 (3-2) faint peak. This behaviour can be seen for all combinations of V_0 and A in Fig. 6.8 with it being the most clear in the $V_0 \mapsto +20$ limit. In the limit of $V_0 \mapsto$ -20, the fermions undergo bosonisation where the density of spinless fermions acts like non-interacting bosons which is a key property of the Fermi-Bose duality [146]. However, in the off-diagonal matrix elements there exists a clear fermionic sign structure. The structure of the OBDM is symmetric along the diagonal due to the OBDM being Hermitian.

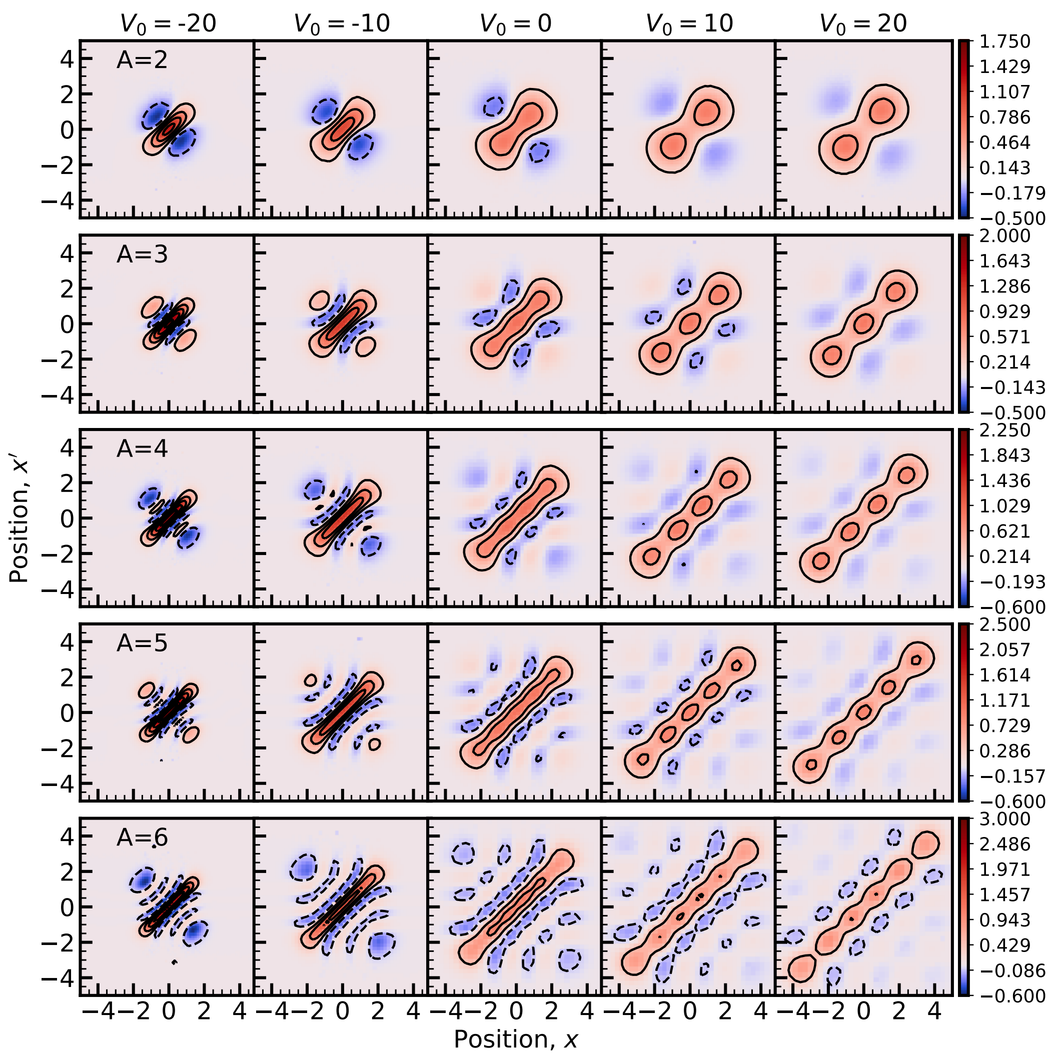


Figure 6.8: The one-body density matrix plots for A=2 to A=6 particles for a wide range of interaction strength, V_0 at the finite-range of $\sigma_0 = 0.5$. This shows how the one-body density matrix varies against interaction strength for a variety of numerical methods.

6.10 Occupation Numbers

The occupation numbers are the eigenvalues of the OBDM and represent how ‘filled’ a corresponding state is. The OBDM is normalised such that its trace is equal to the number of particles, and its eigenvalues lie in the range of $0 \leq n_i \leq 1$ whose sum equals the number of particles, i.e. $\sum_i n_k = A$ [144]. In the non-interacting case, the one-body density matrix is idempotent, i.e. $\rho^2 = \rho$, this results in the eigenvalues being either

0 or 1. If the eigenvalue of a given orbital is 1 it represents a particle fully occupying that state, and if its value is 0 it represents no particle occupying that state. This results in the wavefunction having the form of a Slater determinant which emerges from the Hartree-Fock mean-field approach. However, as the interaction strength becomes non-zero the orbitals of the ground-state wavefunction move from being a pure-state to being a mixed-state which are represented by the eigenvalues no longer being strictly 1 or 0. In the interacting case, they lie in the aforementioned range, $0 \leq n_i \leq 1$ where n_i is the eigenvalue of the i^{th} single-particle state [9].

The eigenvalues are calculated via the use of `numpy.linalg.eigh` on the one-body density matrices of Fig. 6.8. The vANN results are compared against the Diag method of Ref. [52], and in the A=2 case we compare with the exact diagonalisation (Space). In the A=2 case, the occupation numbers of the $i < A$ occupied orbitals have excellent agreement across all interaction strengths. For the occupation numbers of the $i \geq A$ unoccupied orbital, there's initially excellent agreement in the occupation numbers in both the attractive and repulsive regimes. As the number of particles increase, we do see a general degradation of the alignment between the vANN occupation number and the Diag results. In the attractive limit, the truncated model space once again affects the behaviour of the results as the occupation number begin to plateau and deviate from the vANN occupation numbers which follow a linear trend with respect to the interaction strength.

Furthermore, in the attractive limit there seems to exist a degeneracy for the two highest occupied orbitals and two lowest unoccupied orbitals. In the repulsive limit, there exists no such degeneracy with all occupation values in both the occupied and unoccupied orbitals being clearly separated. The vANN results do clearly reproduce the exact CI results, however, there is a noticeable noise to this which emerges from the statistical noise floor of the methodology. However, as the interaction strength is decreased towards the non-interacting case a clear discrepancy arises which is apparent for all other $n_i > A$ orbitals for all values of A. The A=2 case has perfect alignment for the first unoccupied orbital until $V_0 \approx \pm 5$ wherein a noise floor severely affects the results. In the limit of either strongly attractive or repulsive there is general agreement in the trends of occupation numbers for the unoccupied orbitals although discrepancies do worsen in the attractive limit as A increases.

The current numerical method for calculating the occupation numbers suffers from a statistical noise floor where the resolution of the results breaks down on the order of 10^{-2}. This statistical noise floor does increase with the number of particles which most likely emerges from the statistical nature of integration technique. As the dimensionality of the integral increases so too does the error. This isn't necessarily a noticeable problem in the $A = 2$ and $i \leq A$ case as the occupation numbers lie in the range of $0.9 \leq n_i \leq 1.0$. However, this is clearly a problem with the unoccupied orbitals as the associated occupation numbers are near zero as the orbitals become mixed under the interaction. The source of this statistical noise emerges from the fact that our density matrices are computed via the ‘ghost’ method of Ref. [18]. This error for the occupation numbers also propagates towards the eigenvectors as well. For this reason, we omit the results for the natural orbitals.

The results of the occupation numbers are shown within Fig. 6.9 with the left (right) column showing the occupied (unoccupied) orbitals in blue (red). As there are multiple occupation numbers to shown for A particles for all values of V_0 we indicate each orbital via the opacity of the corresponding state. For hole states (i < A) the darker the colours the higher the orbital is whereas for particle states (i > A) the opposite is true with darker colours representing lower orbitals. The maximum orbital shown is arbitrarily chosen to 14 as it allows all occupied orbitals to be shown regardless of A. In the non-interacting case the hole states will be one with particles states set to zero, however, as the interaction strength becomes non-zero these occupations numbers will deviate to any value $0 \leq n_i \leq 1$.

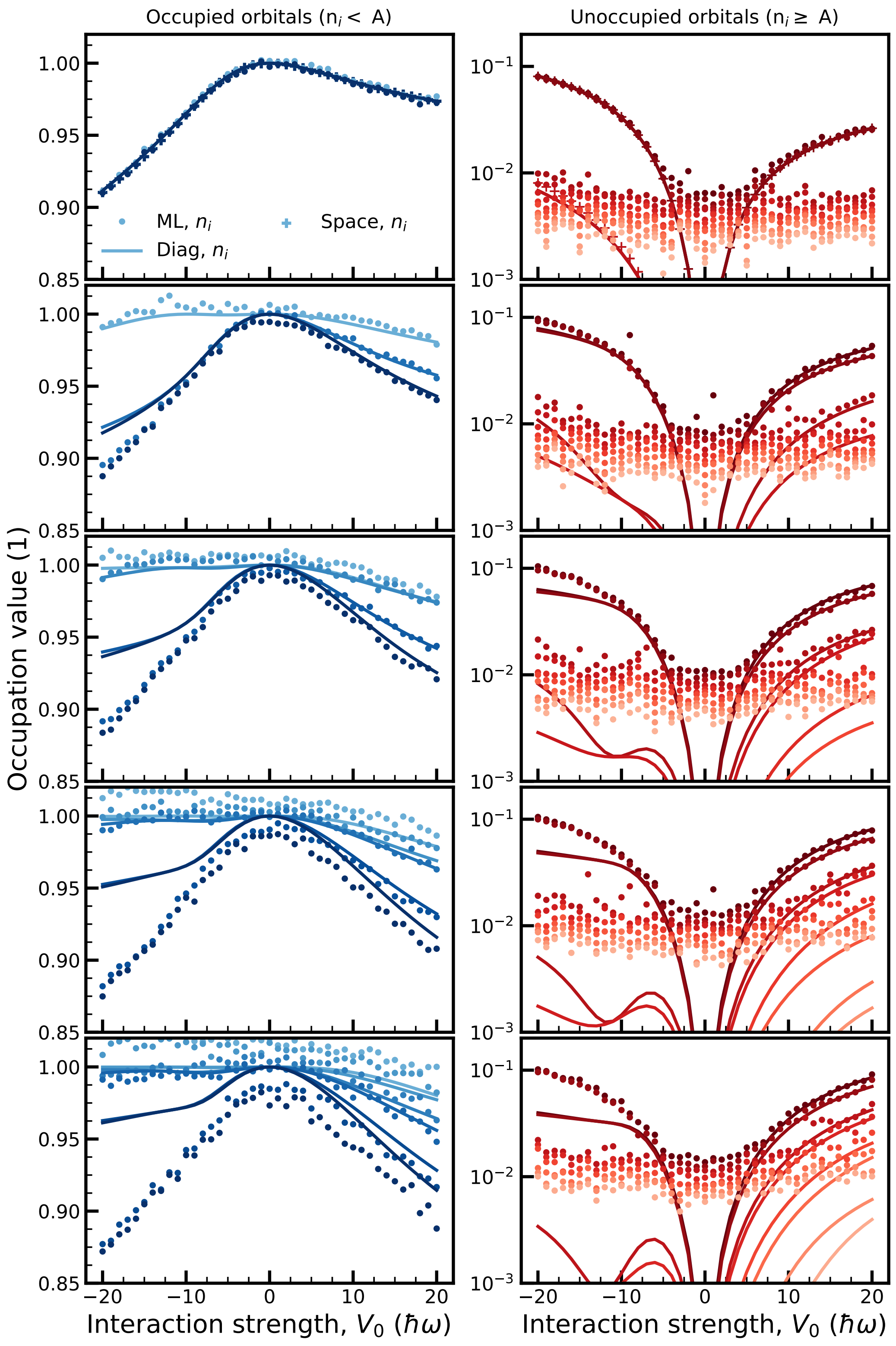


Figure 6.9: The eigenvalues of the one-body density matrix which are denoted the occupation numbers. These plots are shown for A=2 (top) to A=6 (bottom) particles for a wide range of interaction strength, V_0 at the finite-range of $\sigma_0 = 0.5$. The numerical methods are defined within the legend of this figure.

6.11 Two-body density

A natural extension to the OBDM is the two-body density matrix (TBDM),

$$\Gamma^{(2)}\left(x_1', x_2', x_1, x_2\right) = \binom{A}{2} \int_{-\infty}^{\infty} \Psi^*\left(x_1, x_2, \ldots, x_A\right) \Psi\left(x_1', x_2', \ldots, x_A\right) dx_3 \ldots dx_A. \quad (6.4)$$

This four-dimensional tensor is Hermitian and antisymmetric in its indices whose trace is the two-body density shown in Fig. 6.10.

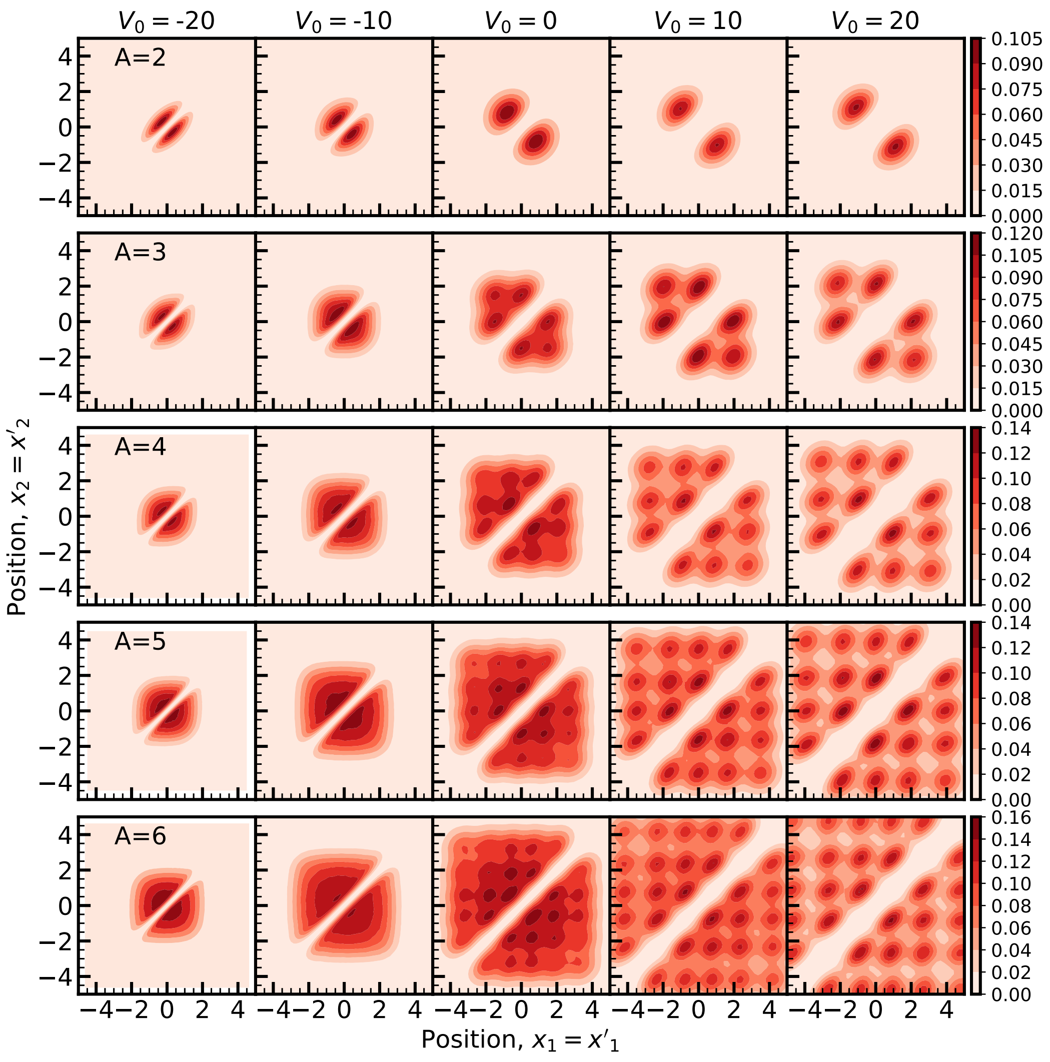


Figure 6.10: The two-body density are shown for A=2 to A=6 particles for a wide range of interaction strength, V_0 at the finite-range of $\sigma_0 = 0.5$.

In the OBDM case, the diagonal of the OBDM was the OBD which represented the classical probability of finding a particle at point, x, within some volume, dx, multiplied by the number of particles. In the TBDM case, the diagonal is the two-body density (TBD) which is normalised to the number of pairs of fermions, i.e. $\binom{A}{2}$. This is defined as $\Gamma^{(2)}(x_1, x_2, x_1, x_2)$ where $x_1' = x_1$ and $x_2' = x_2$ [144]. The antisymmetry of this matrix also exhibits the PEP such that if two indices are identical then the corresponding matrix element equals zero [144]. This can be seen in Fig. 6.10 where there exists a 'valley' separating the off-diagonal elements along the diagonal whose width is dependent on the interaction strength, V_0, however it is always *non-zero* due to the PEP. This property emerges from the fermionic nature of the physical system, and is a consistent property of the system regardless of the number of particles and interaction strength as expected.

The TBD, like the OBD, is calculated stochastically over multiple batches of walkers. We follow the same stochastic methodology of the OBD (Sec. 5.2) to compute the TBD efficiently. We sample the ground-state wavefunction with 4096 walkers over 10^4 batches with a thinning factor of 10 to generate a superset of samples, $X \sim |\Psi|^2$. The samples of x_1 and x_2 from $X = [x_1, \dots, x_A]$ are then taken with the remaining $x_3, \dots, x_A$ being discarded. A 2D histogram is then constructed via the x_1 and x_2 samples which are weighted by $\frac{1}{N_s \Delta x^2}\binom{A}{2}$ where Δx is the bin width of the histogram and $N_s \sim 4.1 \times 10^7$ is the total number of samples. This effectively takes the samples distributed in $\mathbb{R}^2$ and discretises them on to a mesh. This histogram is then normalised to $\binom{A}{2}$ in accordance to Eq. 6.4. This allows for an effective stochastic methodology for calculating the two-body density.

As the diagonal components are strictly PSD [144] this has an added numerical benefit when computing expectations. For the OBDM, it was composed of positive and negative contributions which cancel within the expectation. For the TBD, all contributions are positive and hence 'count' towards the expectation. This allows for the TBD to be finer than the OBDM for the same number of samples. A pedagogical example of this, in a different context, is given within Sec. 8.1.1 of Ref. [152].

The two-body densities for a wide range of A and V_0 are shown in Fig. 6.10. The physical interpretation of the two-body density (TBD), $\gamma^{(2)}(x_1' = x_1, x_2' = x_2)\mathrm{dx}_1\mathrm{dx}_2$, is the number of pairs $\binom{A}{2}$ multiplied by the probability of finding one particle in dx_1 and another particle in dx_2 with all other particles having an arbitrary position [144]. This can be interpreted as an average distances between arbitrary pairs of fermions within the ground-state wavefunction. The distribution of fermions is heavily dependent on the interaction strength of the Gaussian interaction and will lead to different physical behaviour in either limit of V_0.

The interaction strength also plays a role in the average distance between pairs of particles. For example, in the strongly attractive limit the fermions coalesce around one another which results in a liquid-like peak, albeit with a valley of zero probability along the diagonal. This effectively compresses all the fermions together so they sit side by side, although, they never sit at the same position due to the PEP [153, 148].

On the other hand in the strongly repulsive regime, the fermions become fixed in space which results in a crystalline structure with the average inter-particle distance increasing as the interaction strength increases [149, 147]. This has the physical interpretation in the TBD of creating a distribution of equidistant peaks as the particles try to minimise the total energy by maximising the distance between one another. In addition, the number of peaks within the TBD is directly related to the number of particles by the relation of $A(A-1)$. A similar behaviour was noted with the OBDM where the number of ripple along the off-diagonal matrix elements was $A-1$. In the TBD case, the number of peaks along the off-diagonal matrix elements become significantly clearer as the interaction strength becomes more repulsive as the interaction contributions confine the fermions into a rigid crystalline structure.

Chapter 7

Conclusion and Outlook

We now conclude the work of this thesis with final remarks and highlight three potential avenues of future work.

7.1 Concluding remarks

The work of this thesis combines two domains of science; Variational Monte Carlo (VMC) for Ab-initio many-body physics, and Machine Learning (ML) to see if the advancements of ML are applicable to VMC. Neural networks are applied to three Hamiltonians with two separate ansäzte and compared against exact diagonalisation as well as other numerical methods. In short, the results for both ansätze show excellent agreement for all three Hamiltonians with exact diagonalisation across a numerous properties of the ground-state wavefunction.

In chapter 4 the deuteron wavefunction for both the S- and D-state is calculated by a FFNN which is shown to efficiently represent both state wavefunction simultaneously. The ground-state energy is calculated by Gauss-Quadrature techniques which coupled with a FFNN ansatz allows for highly accurate expectation values (99.9%). The fidelity of these state wavefunctions is compared against the exact state wavefunction which nears unity indicating that the FFNN ansatz can accurately represent the wavefunction. Interestingly enough, the D-state probability which is correlated to the Tensor force is also accurately modelled with the exact value being 4.51%. The FFNN ansatz reaches the same value with the error being on the order of 0.05%.

In chapter. 5 the main DNN ansatz of this work is used to solve the TISE for the HO Hamiltonian for a many-body system of A=2 to A=6 spinless fermions. The DNN is initially pre-trained towards to a set of target single-particle orbitals and then minimised towards the ground-state wavefunction. The methodology used to solve this A-body

Schrödinger equation is presented for both Chp. 5 and Chp. 6, and involves two phases of pre-training to a target wavefunction then subsequently minimising the energy of the DNN towards the ground-state in reinforcement-like manner. The convergence of the pre-training and energy minimisation is also compared against different DNN architectures. I conclude that the best architecture in terms of energy minimisation for the work of this thesis is a DNN of 2 layers of 64 nodes each with a single GSD. The use of a single GSD emphasises the efficient use of backflow in the single-particle orbitals of the DNN. In principle, the results of this thesis could be further improved by increasing the number of GSDs. The ground-state energy wasn't the only property studied of the many-body wavefunction. I also discussed the OBD and OBDM which were both calculated via stochastic methods. These results were compared against the exact results and showed that DNN ansätze can accurately compete against exact results. The one-body density results have excellent fidelity across all values of A studied. The OBDM of the vANN results show great fidelity against the HO OBDM with some artefacts emerging from the use of our stochastic methods. The main bulk of the ground-state wavefunction analysis is reserved for Chp. 6.

In chapter 6, we compare the vANN methodology for the HO + finite-range interaction Hamiltonian against exact diagonalisation as well as other numerical methods for a wide range of properties of the ground-state wavefunction. The ground-state energy is benchmarked against exact methods for a wide range of interaction strengths, V_0, for a fixed finite-range $\sigma_0 = 0.5$. The interaction range is also varied from 0.1 to 3.0 with $V_0 = \pm 20$ to benchmark how the range of the finite-range interaction affects performance in either limits of the Gaussian interaction. The results of the vANN methodology in both ($V_0 = \pm 20$) cases showed that it achieved excellent results on par with exact methods. The vANN results even surpass the 'exact' results in the strongly attractive limit due to basis set truncation of the 'exact' method from which the vANN method did not suffer as it works directly within the continuum, and hence achieves much lower ground-state energies. The pairing-gap is also measured in which the vANN results accurately reflect the 'exact' pairing-gap. This highlights that the vANN can accurately account for pairing effects for fermions for all values of the interaction strength studied.

As previously mentioned, the ground-state energy is accurately computed by the vANN methodology. By comparing the individual components of the ground-state energy; kinetic, potential, and interaction, allow for different physical phenomena to be inferred. In the strongly attractive limit the interaction dominates over the kinetic, with the potential being negligible, which forces the particles to be pressed together forming a single peak one-body density referred to as Bosonisation. In the strongly repulsive limit the potential dominates over the kinetic which equals to the interaction. The minimising of the kinetic results in the fermions being near stationary and equidistant from one another which is referred to as Wigner crystallisation.

The one-body density describes how the fermions are distributed within one-dimensional space. In the interacting case, there were two main physical phenomena; bosonisation in the strongly attractive case, and Wigner crystallisation in the strongly repulsive case. In the former, a single peak is formed at the origin whereas in the latter a crystal is formed with distinct peaks equally separated in distance to each other. By taking the second moment of this distribution, defines the root-mean-square (RMS) size which is the effective size of the physical system. As the interaction strength increases from -20 to +20 the RMS size increases as the particles separate from one another under the interaction strength. These values were compared against ‘exact’ and mean-field methods and showed strong consistency until both methods breakdown in the strongly attractive limit.

The statistical algorithm of Ref. [18] allowed me to compute the OBDM which represents how two different points within the one-dimensional space correlate with each other. The results highlighted how this distribution varies with interaction strength with the peaks of the distribution being directly correlated with the interaction strength. The eigenvalues show general agreement across interaction strength albeit with slight difference in the strongly attractive regime. The main issue is the statistical noise floor which seems to be a main fault of the current methodology which affects eigenvalues below 10^{-2}. The final aspect of analysis was the two-body density which also exhibited the bosonisation and crystallisation phenomena.

In the case for the deuteron, the ansatz can resolve the ground-state wavefunction with a fidelity of over 99.99% and compute near exact ground-state energies. For the non-interacting system of A fermions, the DNN ansatz can effectively resolve the ground-state for a few number of particles indicating significant compression of the many-body wavefunction. This allowed for numerous properties of the ground-state wavefunction be efficiently computed via the vANN methodology. In the interacting case, for A fermions, the DNN ansatz could accurately resolve the ground-state wavefunction on par with exact methodologies for a wide number of particles, interaction strength, and interaction range. These results were benchmarked across a wide range of physical properties of the system and agreement strongly with ‘exact’ methodologies. The results of this thesis give clear evidence that DNN ansätze are efficient functional forms for many-body wavefunctions and can accurately represent ground-state wavefunctions.

7.2 Future Work

The future work of this thesis has numerous avenues of potential improvement. As a first step, one could consider adding convolution layers that explicitly incorporate 2-body position information to the DNN ansatz [5]. This would entail comparing the ground-state energy of the DNN ansatz when it utilises not only one-body positions but also two-body positions from all pairs of fermions. Adding more components to the DNN ansatz would allow for feature ablation studies to fundamentally study which parts of the DNN ansatz are accurately representing the ground-state wavefunction. It could perhaps guide the development of new components for DNN ansätze to accurately model fermionic systems more efficiently. The inclusion of additional features is not restricted to just the input layer but could also be applied to intermediate layers as well, an example could be moving from generalised *Slater* determinants to generalised *Vandermonde* determinants which scale quadratically rather than cubically [109].

A different extension to the network would be to explicitly incorporate the spin of the fermions, so that the DNN ansatz becomes anti-symmetric to permutation of position and permutation of spin. This results in the global wavefunction being symmetric in space but antisymmetric in spin and vice versa. The trick of Ref. [5] was to explicitly split the fermions in space among spin-up and spin-down particles because the Hamiltonian was spin independent. However, with nuclear Hamiltonians this trick is not applicable and therefore the ansatz must explicitly deal with this symmetry. This would require the creation of a new intermediate equivariant layer which handles both symmetries directly or one could define two streams within the network where each individual stream handles the space and spin symmetries separately.

Besides extensions to the DNN ansatz, the sampling process could be changed as well. The use of neural networks for sampling is referred to as *generative modelling*. One example is Normalising Flows (NFs) in which one can exactly sample from an arbitrary PDF in an efficient and tractable manner. The trick of this method is to partition the input into two subsets, X_A and X_B, and then transform one subset based on input from the other subset in an alternating manner which results in the Jacobian of this transformation being triangular. This triangular property is useful as when transforming a probability distribution the probabilities pre-transformation need to be updated by the determinant of the transformation. If the Jacobian is triangular then its determinant can be calculated in linear time with respect to the dimensionality of the PDF whereas the naïve determinant calculation scales cubicly.

This partitioning trick allows for NFs to model PDFs in a tractable way, however, when representing the Born probability of a many-body wavefunction the PDF must abide by certain symmetries [54]. In the case of fermionic many-body wavefunctions, its Born PDF must be symmetric with respect to particle exchange, however, doing so breaks the triangular property of the Jacobian and therefore makes NFs for symmetric PDFs intractable. A solution to this intractbility issue is to move pass NFs to Continuous Normalising Flows (CNFs) which can circumvent this issue [154, 155].

Finally, the numerical optimisation of this thesis focusses primarily around first-order methods, e.g. Adam. Some preliminary work around preconditioning methods, e.g. QN-KFAC, were discussed and compared against KFAC and Adam in Chp. 5. This preconditioning optimiser can be further extended by modifying two components of this optimiser. Firstly, the trust-region in which the rescaling is performed by directly regularising the preconditioning matrix as opposed to directly employing trust-region methods. The current version results in a 'hypersphere' within the parameter space which treats all directions equally. In practice this isn't ideal because not all directions are equally valid. In the ideal setting, we'd want to directly utilise trust-region methods and employ a hyperellipsoid for the trust-region so that each direction within the parameter space is trust separately which should lead to more efficient optimisation.

Secondly, the preconditioning matrix used is a mean-centred Fisher Information Matrix (QFIM) which preconditions the steepest gradient to the 'natural' gradient. This effectively rotates (and scales) the gradient within the parameter space to a more optimal direction, however, it's not *the* optimal direction. The optimal preconditioning would be the Hessian. However, a full calculation of the Hessian would scale with the cube of the number of parameters which is impractical for DNN ansätze. Also, given that the Hessian isn't PSD its convergence is also unbounded. For this reason, it motivates creating a preconditioning matrix which is a PSD approximation to the Hessian of the expectation value of the energy. This would define an even more optimal direction for the preconditioned gradient within the parameter space, and would in principle allow for even more improvement for VMC optimisation with DNN ansätze. In order for such a preconditioning matrix to be as computationally efficient as the QFIM, we would need to apply some form of KA (or equivalent) to decompose the preconditioning gradient in to the Kronecker product of two smaller matrices. If such a methodology were feasible, then the creation of an even more efficient KFAC variant is, in theory, possible. This would allow for the creation of a specifically tailored optimiser for VMC simulations that could efficiently simulate many-body systems with 10s of particles. Currently these systems are hard to tackle with first-order methods because of the large number of required epochs to achieve convergence.

In summary, the avenues of future research for DNN ansätze is bountiful and encompass numerous fields of research from improving ML models to accurately represent many-body wavefunctions, to advances in specifically tailored higher-order optimisation algorithms which can fully exploit the flexibility of ML research techniques.

Appendix A

Stable Summed Signed-Log Determinant

This chapter solely entails the full derivation of the custom summed determinant function as well as its motivation.

A.1 Function Definition

The DNN ansatz of this work effectively reduces to a sum of D generalised Slater determinants, GSDs, where B_d denotes the d^{th} GSD (including envelopes). We omit any weighting on the GSDs due to it being a redundant parameter as per Ref. [6]. This results in the wavefunction being defined as,

$$\Psi(x_1, \ldots, x_D) = \sum_{d=1}^{D} \det(B_d) . \tag{A.1}$$

The sign and log-absolute values of Ψ are calculated via directly taking $\mathrm{sgn}(\Psi)$ and $\log|\Psi|$ respectively. In practice, we perform the determinant calculation within the log-domain for added numerical stability and perform a signed-logsumexp function which becomes equivalent to Eq. A.1 [110]. The signed-log determinant function is computed via PyTorch's `torch.linalg.slogdet` function which is computed via an LU-decomposition. If we denote $\phi_d = \det(B_d)$, its sign-log determinant is defined as,

$$\{\mathrm{sgn}(\phi_d), \log|\phi_d|\} = \mathrm{slogdet}(B_d) . \tag{A.2}$$

This function takes an $n \times n$ matrix and returns a tuple of the sign of the determinant as well as the log of the abs of the determinant. Now we could immediately recombined these values to reconstruct the determinant within the linear domain, however, this can lead to numerical issues if the determinants are poorly scaled. In other words, if we have a set of D determinants and all but one are of similar magnitudes then the remaining determinant value will skew the sum and can lead to numerical instability when taking its exponent. Therefore, we introduce a ‘max-subtraction’ operation which shift all determinants to be zero-centred around the maximum log-determinant value. This redefines the determinant as,

$$\underset{shift}{\det}(B_d) = \text{sgn}(\phi_d) * \exp\left(\log|\phi_d| - \max_d(\log|\phi_d|)\right) \tag{A.3}$$

where $\underset{shift}{\det}(B_d)$ denotes the ‘max-subtracted’ determinant value for the d^{th} matrix. These values are then summed, and the absolute log value is calculated. The maximum log-absolute value is added back in to this sum to define the total log-absolute value of Eq. A.1,

$$\log|\Psi| = \max_d(\log|\phi_d|) + \sum_d \underset{shift}{\det}(B_d), \tag{A.4}$$

and the sign is calculated via,

$$\text{sgn}(\Psi) = \text{sgn}\left(\sum_d \underset{shift}{\det}(B_d)\right). \tag{A.5}$$

The redefinition of the wavefunction in terms of its log-absolute and sign separately within the log-domain allows for more numerical stability when computing the determinant of each GSD. In addition, the max-subtracted sum yields further numerical stability when the determinants within the sum are poorly scaled. One drawback of using the signed-log determinant is that it is ill-defined for singular matrices as the derivative is,

$$\frac{\partial \text{logdet}(B_d)}{\partial B_d} = B_d^{-1} \tag{A.6}$$

which for a singular matrix involves a divide by zero error. This numerical instability can be significantly mitigated by the use of SVD-decomposition on the first and second derivatives which effectively act as pseudo-inverses. These derivatives will be defined in the following sections using reverse-mode AD (Sec. 3.6.3) firstly by the analytical derivatives then a numerical stable version will be defined by use of SVD-decompositions. As Eqs. A.4, A.5 are numerically equivalent to Eq. A.1 subsequent derivations for the derivatives start from Eq. A.1.

A.2 First Derivative Derivation

All derivatives derived here work for arbitrary scalar losses. This custom derivative, as well as its second derivative, is implemented via the use of custom autograd function API - `torch.autograd.Function`. At the time of implementation, only reverse-mode AD functions were definable and hence we only show the reverse-mode derivation for this function. The function we defined is comprised of 2 input tensors, in the language of PyTorch, and returns 2 output tensors which represent the sign and log-absolute values of the sum of determinants from Eq. A.1. The input, B_d, of Eq. A.1 is defined as,

$$B_d = \mathcal{A}_d \odot \exp\left(S_d\right) \tag{A.7}$$

where $\mathcal{A}_d$ is the d^{th} GSDs and S_d is the d^{th} log-envelope matrix which enforces the correct asymptotic behaviour for the orbitals of the corresponding GSD. Both $\mathcal{A}_d$ and S_d are implemented as a single tensor object of size '[B, D, A, A]' where B corresponds to the number of samples (which is broadcastable), D the number of GSDs, and A is the number of particles/orbitals of the physical system being represented by the DNN ansatz. This results in the custom function requiring an analytical expression for the derivative with respect to both A and S. For an arbitrary scalar loss, $\mathcal{L}$, its derivative with respect to A is defined as,

$$\frac{\partial \mathcal{L}}{\partial \mathcal{A}_{i,kl}} = \frac{\partial \mathcal{L}}{\partial \operatorname{sgn}(\Psi)} \frac{\partial \operatorname{sgn}(\Psi)}{\partial \mathcal{A}_{i,kl}} + \frac{\partial \mathcal{L}}{\partial \log|\Psi|} \frac{\partial \log|\Psi|}{\partial \mathcal{A}_{i,kl}}. \tag{A.8}$$

This can be simplified further due to the fact that $\frac{\partial \operatorname{sgn}(\Psi)}{\partial \mathcal{A}_{i,kl}}$ equals zero by definition. It is ill-defined at $\Psi(X) = 0$, however, it is statistically impossible for samples to populate the region $\Psi(X) = 0$ as the MH-sample will reject such samples. We further reduce Eq. A.8 by following the adjoint notation of Sec. 3.6.3 to,

$$\frac{\partial \mathcal{L}}{\partial \mathcal{A}_{i,kl}} = \overline{\log|\Psi|}^{\mathcal{L}} \frac{\partial \log|\Psi|}{\partial \mathcal{A}_{i,kl}}. \tag{A.9}$$

We can simplify Eq. A.9 by introducing partial derivatives with respect to $B_{j,mn}$, so that,

$$\frac{\partial \mathcal{L}}{\partial \mathcal{A}_{i,kl}} = \overline{\log|\Psi|}^{\mathcal{L}} \sum_{j,mn} \frac{\partial \log|\Psi|}{\partial B_{j,mn}} \frac{\partial B_{j,mn}}{\partial \mathcal{A}_{i,kl}}. \tag{A.10}$$

We further use the partial derivatives and the mathematical identity (Eq. A.12),

$$\frac{\partial \log|\Psi|}{\partial B_{j,mn}} = \frac{\partial \log|\Psi|}{\partial \Psi} \frac{\partial \Psi}{\partial \det\left(B_j\right)} \frac{\partial \det\left(B_j\right)}{\partial B_{j,mn}} = \frac{\det\left(B_j\right) B_{j,mn}^{-T}}{\Psi}, \tag{A.11}$$

$$\frac{\partial B_{j,mn}}{\partial \mathcal{A}_{i,kl}} = e(S_j, mn)\, \delta_{ji} \delta_{mk} \delta_{nl}, \tag{A.12}$$

so that Eq. A.11 becomes,

$$\frac{\partial \mathcal{L}}{\partial \mathcal{A}_{i,kl}} = \frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi} \sum_{j=1}^{D} \sum_{mn} \operatorname{Cof}\left(\mathcal{A}_j \odot S_j\right)_{mn} e\left(S_{j,mn}\right) \delta_{ji} \delta_{mk} \delta_{nl} = \frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi} \operatorname{Cof}(\mathcal{A}_i \odot S_i)_{kl}\, e\left(S_{i,kl}\right). \tag{A.13}$$

The first derivative of $\mathcal{L}$ with respect to $S_{i,kl}$ can be determined in a similar manner,

$$\frac{\partial \mathcal{L}}{\partial S_{i,kl}} = \frac{\partial \mathcal{L}}{\partial \operatorname{sgn}(\Psi)} \frac{\partial \operatorname{sgn}(\Psi)}{\partial S_{i,kl}} + \frac{\partial \mathcal{L}}{\partial \log|\Psi|} \frac{\partial \log|\Psi|}{\partial S_{i,kl}} = \overline{\log|\Psi|}^{\mathcal{L}} \frac{\partial \log|\Psi|}{\partial S_{i,kl}}. \tag{A.14}$$

$$= \overline{\log|\Psi|}^{\mathcal{L}} \sum_{j=1}^{D} \sum_{mn} \frac{\partial \log|\Psi|}{\partial B_{j,mn}} \frac{\partial B_{j,mn}}{\partial S_{i,kl}} \tag{A.15}$$

By using the definition of Eq. A.11, and,

$$\frac{\partial B_{j,mn}}{\partial S_{i,kl}} = \left(\mathcal{A}_{j,mn} \odot e(s_{j,mn})\right) \delta_{ji} \delta_{mk} \delta_{nl}, \tag{A.16}$$

we find that,

$$\frac{\partial \mathcal{L}}{\partial S_{i,kl}} = \frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi} \left(\mathcal{A}_{i,kl} \odot e(S_{i,kl})\right) \operatorname{Cof}\left(\mathcal{A}_{i,kl} \odot e(S_{i,kl})\right). \tag{A.17}$$

Therefore, we can construct a custom derivative for the first derivative of Eq. A.1 by plugging in Eqs. A.13, A.17.

Within both formulae there exist 2 points of instability, the division by Ψ and the Cofactor matrix which involves the inverse of the input matrix can become ill-defined. Therefore, we turn towards applying a SVD-decomposition to effectively construct a numerically stable pseudo-inverse in the case of inverting a singular matrix. The Singular Value Decomposition (SVD) re-defines an arbitrary matrix, M, as the product of 3 matrices, U, Σ, V^T, i.e.,

$$M \overset{\text{SVD}}{=} U \Sigma V^T \tag{A.18}$$

where U and V^T are orthonormal matrices, with determinants equal to ± 1, and Σ is a diagonal matrix of strictly positive matrix elements with its determinant always greater than zero. With this decomposition, the ratio of the cofactor and the inverse ϕ matrices in Eq. A.17 reduces to,

$$\frac{\operatorname{Cof}\left(\mathcal{A}_{i,kl} \odot e(S_{i,kl})\right)}{\Psi} = \frac{\det(U) \det(\Sigma) \det\left(V^T\right) \left(U \Sigma V^T\right)^{-T}}{\Psi}. \tag{A.19}$$

The cofactor matrix is defined as $\operatorname{Cof}(X) = \det(X) X^{-1}$. We use the SVD-decomposition to efficiently define a numerically stable backward, and double-backward derivation for the derivatives of `torch.linalg.slogdet` beyond the standard PyTorch derivation. Due to U and V^T being orthonormal their determinants are purely ± 1 with their inverse transposes equally themselves, i.e. $U^{-T} = U$ and $V^{T^{-T}} = V^T$. The determinant of Σ is,

$$\det(\Sigma) = \prod_k \sigma_k \tag{A.20}$$

which can be merged with Σ^{-1} to define the Γ matrix,

$$\gamma_l = \sigma_l^{-1} \prod_k \sigma_k = \prod_{k \neq l} \sigma_k. \tag{A.21}$$

In the case of a singular (or near singular) matrix, the SVD decomposition results in a Σ matrix where a σ_k value can equal zero (or be close to zero). By multiplying then dividing by such a small value can result in round-off and in the case of $\sigma_k = 0$ a divide by zero error, therefore, omitting this redundant multiply-divide operation results in a numerical stable operation for Eq. A.21. The last remaining instability of Eq. A.19 is the divide by Ψ factor. This can be mitigated by decomposing Ψ into a sign and log-absolute factor which are merged into a 'normed' Γ matrix,

$$\frac{\mathrm{Cof}\left(\mathcal{A}_{i,kl} \odot S_{i,kl}\right)}{\Psi} = \frac{\det(U)\det\left(V^T\right)\left(U\Gamma V^T\right)}{\Psi} \tag{A.22}$$

$$= \det(U)\det\left(V^T\right)\mathrm{sgn}(\Psi)\frac{1}{\exp(\log|\Psi|)}\left(U\Gamma V^T\right) \tag{A.23}$$

As $\mathrm{sgn}(x) = \frac{1}{\mathrm{sgn}(x)}$ we multiply by the sign rather than divide for an increase in speed. The $\exp(\log|\Psi|)$ term is directly included into Γ to define a normalised equivalent,

$$\Gamma_i^{\mathrm{norm}} = \mathrm{diag}\left(\exp\left(\sum_{l \neq k} \log(\sigma_i)_l - \log|\Psi|\right)_k\right). \tag{A.24}$$

Eq. A.24 is equivalent to Eq. A.21 divided by Ψ, and is represented as a single numerically stable quantity. Therefore, Eq. A.19 simplifies to,

$$\frac{\mathrm{Cof}\left(\mathcal{A}_{i,kl} \odot S_{i,kl}\right)}{\Psi} = \det(U_i)\det\left(V_i^T\right)\mathrm{sgn}(\Psi) e\left(l\left(U_i\Gamma_i^{\mathrm{norm}}V_i^T\right)\right) \odot s\left(U_i\Gamma_i^{\mathrm{norm}}V_i^T\right), \tag{A.25}$$

where $l(M)$ and $s(M)$ are element-wise log and sign functions respectively, and $\odot$ is an element-wise multiplication between matrices. By utilising Eq. A.25, the numerically stable SVD version of Eqs. A.13 and A.17 are,

$$\frac{\partial \mathcal{L}}{\partial \mathcal{A}_{i,kl}} = \overline{\log|\Psi|}^{\mathcal{L}} \det(U_i)\det\left(V_i^T\right)\mathrm{sgn}(\Psi)s\left(U_i\Gamma_i^{\mathrm{norm}}V_i^T\right) \odot e\left(l\left(U_i\Gamma_i^{\mathrm{norm}}V_i^T\right) + S_i\right) \tag{A.26}$$

$$\frac{\partial \mathcal{L}}{\partial S_{i,kl}} = \overline{\log|\Psi|}^{\mathcal{L}} \det(U_i)\det\left(V_i^T\right)\mathrm{sgn}(\Psi)s\left(U_i\Gamma_i^{\mathrm{norm}}V_i^T\right) \odot \mathcal{A}_i \odot e\left(l\left(U_i\Gamma_i^{\mathrm{norm}}V_i^T\right) + S_i\right) \tag{A.27}$$

This concludes the first derivative for the Stable Summed Signed-Log Determinant function. We now turn towards deriving a stable second derivative.

A.3 Second Derivative Derivation

The implementation of a custom second derivative function requires embedding a `torch.autograd.Function` within a `torch.autograd.Function`. This original object outputs Eq. A.1, and its backward method calls a second `torch.autograd.Function` with its forward method and backward method representing the 1st derivative and 2nd derivative respectively. Therefore, when computing the 2nd derivative for this function we just have to manually define not just the second derivative with respect to A_i and S_i but also the backward sensitivities of the first derivative, i.e. we also need manual expressions for derivatives with respect to $\overline{\text{sgn}(\Psi)}$ and $\overline{\log|\Psi|}$. Therefore, we need to determine, $\frac{\partial \mathcal{F}}{\partial \mathcal{A}_i}$, $\frac{\partial \mathcal{F}}{\partial S_i}$, $\frac{\partial \mathcal{F}}{\partial \overline{\text{sgn}(\Psi)}^{\mathcal{L}}}$, and $\frac{\partial \mathcal{F}}{\partial \overline{\log|\Psi|}^{\mathcal{L}}}$.

Due to the use of reverse-mode AD, PyTorch's AD engine will return the following quantities;

$$\frac{\partial \mathcal{F}}{\partial \left(\frac{\partial \mathcal{L}}{\partial \mathcal{A}_i}\right)} = \overline{\left(\frac{\partial \mathcal{L}}{\partial \mathcal{A}_i}\right)}^{\mathcal{F}} := \overline{G}_i \tag{A.28}$$

$$\frac{\partial \mathcal{F}}{\partial \left(\frac{\partial \mathcal{L}}{\partial S_i}\right)} = \overline{\left(\frac{\partial \mathcal{L}}{\partial S_i}\right)}^{\mathcal{F}} := \overline{H}_i \tag{A.29}$$

$$\frac{\partial \mathcal{F}}{\partial \left(\frac{\partial \mathcal{L}}{\partial \overline{\text{sgn}(\Psi)}^{\mathcal{L}}}\right)} = \overline{\left(\frac{\partial \mathcal{L}}{\partial \overline{\text{sgn}(\Psi)}^{\mathcal{L}}}\right)}^{\mathcal{F}} \tag{A.30}$$

$$\frac{\partial \mathcal{F}}{\partial \left(\frac{\partial \mathcal{L}}{\partial \overline{\log|\Psi|}^{\mathcal{L}}}\right)} = \overline{\left(\frac{\partial \mathcal{L}}{\partial \overline{\log|\Psi|}^{\mathcal{L}}}\right)}^{\mathcal{F}} \tag{A.31}$$

Firstly, we will provide expressions for Eqs. A.30 and A.31 respectively before deriving Eqs. A.28 and A.29. Eq. A.30 is solved by the definition of the adjoints independence to its input. In short, $\frac{\partial \mathcal{L}}{\partial \overline{\text{sgn}\,\Psi}^{\mathcal{L}}}$ is independent of A_i and S_i, hence its derivative is equal to zero by definition, i.e.,

$$\frac{\partial \mathcal{F}}{\partial \overline{\text{sgn}(\Psi)}^{\mathcal{L}}} = 0. \tag{A.32}$$

By using the chain rule,

$$\frac{\partial \mathcal{F}}{\partial \overline{\log|\Psi|}^{\mathcal{L}}} = \sum_{j=1}^{D}\sum_{mn} \overline{G}_{j,mn} \frac{\partial \left(\frac{\partial \mathcal{L}}{\partial \mathcal{A}_{j,mn}}\right)}{\partial \overline{\log|\Psi|}^{\mathcal{L}}} + \sum_{j=1}^{D}\sum_{mn} \overline{H}_{j,mn} \frac{\partial \left(\frac{\partial \mathcal{L}}{\partial S_{j,mn}}\right)}{\partial \overline{\log|\Psi|}^{\mathcal{L}}} \tag{A.33}$$

By using Eqs. A.13 and A.17, we can reduce the derivative to,

$$\begin{aligned}\frac{\partial \mathcal{F}}{\partial \overline{\log|\Psi|}^{\mathcal{L}}} &= \sum_{j=1}^{D}\sum_{mn}\overline{G}_{j,mn} e(S_{j,mn})\frac{\mathrm{Cof}\left(\mathcal{A}_j \odot e(S_j)\right)_{mn}}{\Psi} \\ &+ \sum_{j=1}^{D}\sum_{mn}\overline{H}_{j,mn}\mathcal{A}_{j,mn} \odot e(S_{j,mn})\frac{\mathrm{Cof}\left(\mathcal{A}_j \odot e(S_j)\right)_{mn}}{\Psi} &\text{(A.34)}\\ &= \frac{1}{\Psi}\sum_{j=1}\mathrm{Tr}\left[\overline{G}_j^T\left(e(S_j)\mathrm{Cof}\left(\mathcal{A}_j \odot e(S_j)\right)\right)\right] \\ &+ \frac{1}{\Psi}\sum_{j=1}\mathrm{Tr}\left[\overline{H}_j^T\left(\mathcal{A}_j \odot e(S_j)\mathrm{Cof}\left(\mathcal{A}_j \odot e(S_j)\right)\right)\right] &\text{(A.35)}\\ &= \frac{1}{\Psi}\sum_{j=1}^{D}\mathrm{Tr}\left[\left((\overline{G}_j + \overline{H}_j \odot \mathcal{A}_j) \odot e(S_j)\right)^T \cdot \mathrm{Cof}\left(\mathcal{A}_j \odot e(S_j)\right)\right]. &\text{(A.36)}\end{aligned}$$

The derivations of Eqs. A.28 and A.29 will be now be shown. By utilising the chain rule again,

$$\frac{\partial \mathcal{F}}{\partial \mathcal{A}_{i,kl}} = \sum_{j=1}^{D}\sum_{mn}\overline{G}_{j,mn}\frac{\partial\left(\frac{\partial \mathcal{L}}{\partial \mathcal{A}_{j,mn}}\right)}{\partial \mathcal{A}_{i,kl}} + \sum_{j=1}^{D}\sum_{mn}\overline{H}_{j,mn}\frac{\partial\left(\frac{\partial \mathcal{L}}{\partial S_{j,mn}}\right)}{\partial \mathcal{A}_{i,kl}}. \tag{A.37}$$

Firstly, we solve for the first term of Eq. A.37,

$$\begin{aligned}\frac{\partial\left(\frac{\partial \mathcal{L}}{\partial \mathcal{A}_{j,mn}}\right)}{\partial \mathcal{A}_{i,kl}} &= \frac{\partial}{\partial \mathcal{A}_{i,kl}}\left(\overline{\log|\Psi|}^{\mathcal{L}} e(S_{j,mn})\mathrm{Cof}\left(\mathcal{A}_{j,mn} \odot e(s_{j,mn})\right)\frac{1}{\Psi}\right)\\ &= \overline{\log|\Psi|}^{\mathcal{L}} e(S_{j,mn})\frac{\partial}{\partial \mathcal{A}_{i,kl}}\left(\mathrm{Cof}\left(\mathcal{A}_{j,mn} \odot e(s_{j,mn})\right)\frac{1}{\Psi}\right)\\ &= \overline{\log|\Psi|}^{\mathcal{L}} e(S_{j,mn})\left(\frac{\partial \mathrm{Cof}\left(\mathcal{A}_{j,mn} \odot e(S_{j,mn})\right)}{\partial \mathcal{A}_{i,kl}}\frac{1}{\Psi} + \mathrm{Cof}\left(\mathcal{A}_{j,mn} \odot e(S_{j,mn})\right)\frac{\partial}{\partial \mathcal{A}_{i,kl}}\left(\frac{1}{\Psi}\right)\right)\\ &= \frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi} e(S_{j,mn})\left(\frac{\partial \mathrm{Cof}\left(\mathcal{A}_{j,mn} \odot e(S_{j,mn})\right)}{\partial \mathcal{A}_{i,kl}} - \frac{\mathrm{Cof}\left(\mathcal{A}_{j,mn} \odot e(S_{j,mn})\right)}{\Psi}\frac{\partial \Psi}{\partial \mathcal{A}_{i,kl}}\right)\end{aligned} \tag{A.38}$$

The individual derivatives of Eq. A.38 can be simplified to,

$$\begin{aligned}
\frac{\partial \mathrm{Cof}\left(\mathcal{A}_{j,mn} \odot e(S_{j,mn})\right)}{\partial \mathcal{A}_{i,kl}} = {} & \frac{\partial \det\left(\mathcal{A}_j \odot e(S_j)\right)}{\partial \mathcal{A}_{i,kl}} \left(\mathcal{A}_{j,mn} \odot e(S_{j,mn})\right)^{-T} \\
& + \det\left(\mathcal{A}_j \odot e(S_j)\right) \frac{\partial \left(\mathcal{A}_{j,mn} \odot e(S_{j,mn})\right)^{-T}}{\partial \mathcal{A}_{i,kl}} \\
& = \delta_{ij} \mathrm{Cof}\left(\mathcal{A}_{i,kl} \odot e(S_{i,kl})\right) e(S_i, kl) \left(\mathcal{A}_{j,mn} \odot e(S_{j,mn})\right)^{-T} \\
& - \delta_{ij} \det\left(A_j \odot e(S_j)\right) \left(\mathcal{A}_{j,nk} \odot e(S_{j,nk})\right)^{-1} e(S_{i,kl}) \left(\mathcal{A}_{j,lm} \odot e(S_{j,lm})\right)^{-1}
\end{aligned} \tag{A.39}$$

and,

$$\frac{\partial \Psi}{\partial \mathcal{A}_{i,kl}} = \frac{\partial}{\partial \mathcal{A}_{i,kl}} \left(\sum_d^D \det\left(A_d \odot e(S_d)\right) \right) = \mathrm{Cof}\left(\mathcal{A}_{i,kl} \odot e(S_{i,kl})\right) e(S_{i,kl}) \tag{A.40}$$

This simplifies Eq. A.38 into,

$$\begin{aligned}
\frac{\partial \left(\frac{\partial \mathcal{L}}{\partial \mathcal{A}_{j,mn}} \right)}{\partial \mathcal{A}_{i,kl}} = {} & \frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi} e(S_{j,mn}) \Bigg\{ \\
& \delta_{ij} \mathrm{Cof}\left(\mathcal{A}_{i,kl} \odot e(S_{i,kl})\right) e(S_i, kl) \left(\mathcal{A}_{j,mn} \odot e(S_{j,mn})\right)^{-T} \\
& - \delta_{ij} \det\left(A_j \odot e(S_j)\right) \left(\mathcal{A}_{j,nk} \odot e(S_{j,nk})\right)^{-1} e(S_{i,kl}) \left(\mathcal{A}_{j,lm} \odot e(S_{j,lm})\right)^{-1} \\
& - \frac{\mathrm{Cof}\left(\mathcal{A}_{j,mn} \odot e(S_{j,mn})\right)}{\Psi} \mathrm{Cof}\left(\mathcal{A}_{i,kl} \odot e(S_{i,kl})\right) e(S_{i,kl}) \Bigg\}.
\end{aligned} \tag{A.41}$$

Therefore, the first part of Eq. A.37 reduces to,

$$\begin{aligned}
& \sum_{j=1}^{D} \sum_{mn} \overline{G}_{j,mn} \frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi} e(S_{j,mn}) \Bigg\{ \\
& \qquad \delta_{ij} \mathrm{Cof}\left(\mathcal{A}_{i,kl} \odot e(S_{i,kl})\right) e(S_i, kl) \left(\mathcal{A}_{j,mn} \odot e(S_{j,mn})\right)^{-T} \\
& \qquad - \delta_{ij} \det\left(A_j \odot e(S_j)\right) \left(\mathcal{A}_{j,nk} \odot e(S_{j,nk})\right)^{-1} e(S_{i,kl}) \left(\mathcal{A}_{j,lm} \odot e(S_{j,lm})\right)^{-1} \\
& \qquad - \frac{\mathrm{Cof}\left(\mathcal{A}_{j,mn} \odot e(S_{j,mn})\right)}{\Psi} \mathrm{Cof}\left(\mathcal{A}_{i,kl} \odot e(S_{i,kl})\right) e(S_{i,kl}) \Bigg\}.
\end{aligned} \tag{A.42}$$

The individual components of Eq. A.42 will be simplified separately and recombined.

The first component of Eq. A.42 reduces to,

$$\begin{aligned}
& \sum_{j=1}^{D} \sum_{mn} \overline{G}_{j,mn} \frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi} e(S_{j,mn}) \left\{ \delta_{ij} \mathrm{Cof}\left(\mathcal{A}_{i,kl} \odot e(S_{i,kl})\right) e(S_i, kl) \left(\mathcal{A}_{j,mn} \odot e(S_{j,mn})\right)^{-T} \right\} \\
& = \frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi} \mathrm{Cof}\left(\mathcal{A}_{i,kl} \odot e(S_{i,kl})\right) e(S_{i,kl}) \mathrm{Tr}\left[\left(\overline{G}_i \odot e(S_i) \right) \cdot \left(\mathcal{A}_i \odot e(S_i) \right)^{-1} \right].
\end{aligned} \tag{A.43}$$

The second component of Eq. A.42 reduces to,

$$\sum_{j=1}^{D}\sum_{mn}\overline{G}_{j,mn}\frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi}e(S_{j,mn})\Big\{-\delta_{ij}\det\big(A_j\odot e(S_j)\big)\big(\mathcal{A}_{j,nk}\odot e(S_{j,nk})\big)^{-1}e(S_{i,kl})\big(\mathcal{A}_{j,lm}\odot e(S_{j,lm})\big)^{-1}\Big\}$$
$$=-\frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi}\det(\mathcal{A}_i\odot e(S_i))e(S_{i,kl})\Big[(\mathcal{A}_i\odot e(S_i))^{-T}\big(\overline{G}_i\odot e(S_i)\big)^{T}(\mathcal{A}_i\odot e(S_i))^{-T}\Big]_{kl}. \tag{A.44}$$

The third component of Eq. A.42 reduces to,

$$\sum_{j=1}^{D}\sum_{mn}\overline{G}_{j,mn}\frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi}e(S_{j,mn})\Big\{-\frac{\mathrm{Cof}\big(\mathcal{A}_{j,mn}\odot e(S_{j,mn})\big)}{\Psi}\mathrm{Cof}\big(\mathcal{A}_{i,kl}\odot e(S_{i,kl})\big)e(S_{i,kl})\Big\} \tag{A.45}$$

$$=-\frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi}\frac{\mathrm{Cof}\big(\mathcal{A}_{i,kl}\odot e(S_{i,kl})\big)}{\Psi}e(S_{i,kl})\sum_{j=1}^{D}\mathrm{Tr}\Big[\big(\overline{G}_j\odot e(S_j)\big)^{T}\mathrm{Cof}\big(\mathcal{A}_j\odot e(S_j)\big)\Big]. \tag{A.46}$$

This simplifies Eq. A.42 to,

$$\begin{aligned}\sum_{j=1}^{D}\sum_{mn}\overline{G}_{j,mn}\frac{\partial\left(\frac{\partial\mathcal{L}}{\partial\mathcal{A}_{j,mn}}\right)}{\partial\mathcal{A}_{i,kl}}=\frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi}e(S_{i,kl})\Big\{&\\ &\mathrm{Cof}\big(\mathcal{A}_{i,kl}\odot e(S_{i,kl})\big)\mathrm{Tr}\Big[\big(\overline{G}_i\odot e(S_i)\big)\cdot(\mathcal{A}_i\odot e(S_i))^{-1}\Big]\\ &-\det(\mathcal{A}_i\odot e(S_i))\Big[(\mathcal{A}_i\odot e(S_i))^{-T}\big(\overline{G}_i\odot e(S_i)\big)^{T}(\mathcal{A}_i\odot e(S_i))^{-T}\Big]_{kl}\\ &-\frac{\mathrm{Cof}\big(\mathcal{A}_{i,kl}\odot e(S_{i,kl})\big)}{\Psi}\sum_{j=1}^{D}\mathrm{Tr}\Big[\big(\overline{G}_j\odot e(S_j)\big)^{T}\mathrm{Cof}\big(\mathcal{A}_j\odot e(S_j)\big)\Big]\Big\}\end{aligned} \tag{A.47}$$

We now turn to second part of Eq. A.37 which expands to,

$$\begin{aligned}\sum_{j=1}^{D}\sum_{mn}\overline{H}_{j,mn}\frac{\partial\left(\frac{\partial\mathcal{L}}{\partial S_{j,mn}}\right)}{\partial\mathcal{A}_{i,kl}}&=\sum_{j=1}^{D}\sum_{mn}\overline{H}_{j,mn}\frac{\partial}{\partial\mathcal{A}_{i,kl}}\left(\mathcal{A}_{j,mn}\odot\frac{\partial\mathcal{L}}{\partial\mathcal{A}_{j,mn}}\right)\\ &=\sum_{j=1}^{D}\sum_{mn}\overline{H}_{j,mn}\left(\delta_{ij}\delta_{km}\delta_{ln}\frac{\partial\mathcal{L}}{\partial\mathcal{A}_{j,mn}}+\mathcal{A}_{j,mn}\frac{\partial^2\mathcal{L}}{\partial\mathcal{A}_{i,kl}\partial\mathcal{A}_{j,mn}}\right)\end{aligned} \tag{A.48}$$

From Eq. A.48 we can see that this equation differs from Eq. A.47 by the inclusion of a $\mathcal{A}_{j,mn}$ factor to $\frac{\partial^2\mathcal{L}}{\partial\mathcal{A}_{i,kl}\partial\mathcal{A}_{j,mn}}$, and the addition of a first derivative term $\frac{\partial\mathcal{L}}{\partial\mathcal{A}_{j,mn}}$ albeit with

Kronecker deltas. The first derivative component simplifies to,

$$\begin{aligned}
&\sum_{j=1}^{D}\sum_{mn}\overline{H}_{j,mn}\left(\delta_{ij}\delta_{km}\delta_{ln}\frac{\partial\mathcal{L}}{\partial\mathcal{A}_{j,mn}}\right)\\
&=\sum_{j=1}^{D}\sum_{mn}\overline{H}_{j,mn}\left(\delta_{ij}\delta_{km}\delta_{ln}\frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi}\mathrm{Cof}(\mathcal{A}_i\odot S_i)_{kl}\,e\left(S_{i,kl}\right)\right)\\
&=\frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi}\overline{H}_{i,kl}e(S_{i,kl})\mathrm{Cof}\left(\mathcal{A}_{i,kl}\odot S_{i,kl}\right) \qquad \text{(A.49)}
\end{aligned}$$

We can now define the second derivative component as the repeat of Eq. A.47 with $\overline{G}_i\mapsto\mathcal{A}_i\odot\overline{H}_i$,

$$\begin{aligned}
\sum_{j=1}^{D}\sum_{mn}\overline{H}_{j,mn}\mathcal{A}_{j,mn}\frac{\partial\left(\frac{\partial\mathcal{L}}{\partial\mathcal{A}_{j,mn}}\right)}{\partial\mathcal{A}_{i,kl}}=\frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi}e(S_{i,kl})\Big\{&\\
&\mathrm{Cof}\left(\mathcal{A}_{i,kl}\odot e(S_{i,kl})\right)\mathrm{Tr}\left[\left(\overline{H}_i\odot\mathcal{A}_i\odot e(S_i)\right)\cdot(\mathcal{A}_i\odot e(S_i))^{-1}\right]\\
&-\det(\mathcal{A}_i\odot e(S_i))\left[(\mathcal{A}_i\odot e(S_i))^{-T}\left(\overline{H}_i\odot\mathcal{A}_i\odot e(S_i)\right)^{T}(\mathcal{A}_i\odot e(S_i))^{-T}\right]_{kl}\\
&-\frac{\mathrm{Cof}\left(\mathcal{A}_{i,kl}\odot e(S_{i,kl})\right)}{\Psi}\sum_{j=1}^{D}\mathrm{Tr}\left[\left(\overline{H}_j\odot\mathcal{A}_j\odot e(S_j)\right)^{T}\mathrm{Cof}\left(\mathcal{A}_j\odot e(S_j)\right)\right]\Big\}
\end{aligned} \qquad \text{(A.50)}$$

Interestingly enough this results in both components of Eq. A.37 being element-wise scaled versions of each other. This allows for the nearly all of the derivative to be re-defined by $\overline{G}_i\mapsto\left(\overline{G}_i+\overline{H}_i\odot\mathcal{A}_i\right)$ and including the first derivative component which emerges as a result of Eq. A.48

$$\begin{aligned}
\frac{\partial\mathcal{F}}{\partial\mathcal{A}_{i,kl}}&=\sum_{j=1}^{D}\sum_{mn}\overline{G}_{j,mn}\frac{\partial\left(\frac{\partial\mathcal{L}}{\partial\mathcal{A}_{j,mn}}\right)}{\partial\mathcal{A}_{i,kl}}+\sum_{j=1}^{D}\sum_{mn}\overline{H}_{j,mn}\frac{\partial\left(\frac{\partial\mathcal{L}}{\partial S_{j,mn}}\right)}{\partial\mathcal{A}_{i,kl}}\\
&=\frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi}\overline{H}_{i,kl}e(S_{i,kl})\mathrm{Cof}\left(\mathcal{A}_{i,kl}\odot e(S_{i,kl})\right)\\
&+\frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi}e(S_{i,kl})\Big\{\mathrm{Cof}\left(\mathcal{A}_{i,kl}\odot e(S_{i,kl})\right)\mathrm{Tr}\left[\left(\left(\overline{G}_i+\overline{H}_j\odot\mathcal{A}_j\right)\odot e(S_i)\right)\cdot(\mathcal{A}_i\odot e(S_i))^{-1}\right]\\
&-\det(\mathcal{A}_i\odot e(S_i))\left[(\mathcal{A}_i\odot e(S_i))^{-T}\left(\left(\overline{G}_i+\overline{H}_j\odot\mathcal{A}_j\right)\odot e(S_i)\right)^{T}(\mathcal{A}_i\odot e(S_i))^{-T}\right]_{kl}\\
&-\frac{\mathrm{Cof}\left(\mathcal{A}_{i,kl}\odot e(S_{i,kl})\right)}{\Psi}\sum_{j=1}^{D}\mathrm{Tr}\left[\left(\left(\overline{G}_i+\overline{H}_j\odot\mathcal{A}_j\right)\odot e(S_j)\right)^{T}\mathrm{Cof}\left(\mathcal{A}_j\odot e(S_j)\right)\right]\Big\}.
\end{aligned} \qquad \text{(A.51)}$$

This can simplify $\frac{\partial \mathcal{F}}{\partial \mathcal{A}_{i,kl}}$ to,

$$\begin{aligned}\frac{\partial \mathcal{F}}{\partial \mathcal{A}_{i,kl}} = \frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi} e(S_{i,kl}) \Big\{ \big(\mathcal{A}_{i,kl} \odot e(S_{i,kl})\big)^{-T} \Big\{ \mathrm{Tr}\Big[\big(\big(\overline{G}_i + \overline{H}_i \odot \mathcal{A}_i\big) \odot e(S_i)\big)^T \cdot \mathrm{Cof}(\mathcal{A}_i \odot e(S_i))\Big]\mathcal{I} - \\ \big(\big(\overline{G}_i + \overline{H}_i \odot \mathcal{A}_i\big) \odot e(S_i)\big)^T \mathrm{Cof}(\mathcal{A}_i \odot e(S_i)) \Big\} \\ + \mathrm{Cof}\big(\mathcal{A}_{i,kl} \odot e(S_{i,kl})\big) \Big\{ \overline{H}_{i,kl} - \frac{1}{\Psi} \sum_{j=1}^{D} \mathrm{Tr}\Big[\big(\big(\overline{G}_i + \overline{H}_i \odot \mathcal{A}_i\big) \odot e(S_i)\big)^T \mathrm{Cof}\big(A_j \odot e(S_j)\big)\Big] \Big\}\end{aligned} \tag{A.52}$$

We arrive at this simplified equation by merging Eq. A.51, and using the definition of $\mathrm{Tr}\big[AB^T\big] = \mathrm{Tr}\big[A^T B\big]$, decomposing $\mathrm{Cof}(A)$ to $\det(A)A^{-T}$, and moving the $\det(A)$ factor inside the third component of Eq. A.51. This allows the third component of Eq. A.51 to merge with the second component Eq. A.51 via the common $\big(\big(\overline{G}_i + \overline{H}_i \odot \mathcal{A}_i\big) \odot e(S_i)\big)^T \mathrm{Cof}(\mathcal{A}_i \odot e(S_i))$ factor. Finally, we turn to solving $\frac{\partial \mathcal{F}}{\partial S_i}$,

$$\begin{aligned}\frac{\partial \mathcal{F}}{\partial S_{i,kl}} &= \sum_{j=1}^{D} \sum_{mn} \big(\overline{G}_{j,mn} + \overline{H}_{j,mn} \odot \mathcal{A}_{j,mn}\big) \frac{\partial^2 \mathcal{L}}{\partial S_{i,kl} \partial \mathcal{A}_{j,mn}} \\ &= \sum_{j=1}^{D} \sum_{mn} \big(\overline{G}_{j,mn} + \overline{H}_{j,mn} \odot \mathcal{A}_{j,mn}\big) \left(\delta_{ij}\delta_{km}\delta_{ln} \frac{\partial \mathcal{L}}{\partial \mathcal{A}_{i,kl}} + \mathcal{A}_{i,kl} \frac{\partial^2 \mathcal{L}}{\partial \mathcal{A}_{i,kl} \partial \mathcal{A}_{j,mn}} \right) \\ &= \big(\overline{G}_{i,kl} + \mathcal{A}_{i,kl} \odot \overline{H}_{i,kl}\big) \frac{\partial \mathcal{L}}{\partial \mathcal{A}_{i,kl}} + \mathcal{A}_{i,kl} \cdot \underbrace{\sum_{j=1}^{D} \sum_{mn} \big(\overline{G}_{j,mn} + \mathcal{A}_{j,mn} \odot \overline{H}_{j,mn}\big) \frac{\partial^2 \mathcal{L}}{\partial \mathcal{A}_{i,kl} \partial \mathcal{A}_{j,mn}}}_{\text{Equivalent to first part of } Eq.A.37 \text{ with } \overline{G}_j \mapsto (\overline{G}_j + \overline{H}_j \odot A_j)}\end{aligned} \tag{A.53}$$

Therefore, this allows for the derivation of $\frac{\partial \mathcal{F}}{\partial S_{i,kl}}$ to utilise the derivation of $\frac{\partial \mathcal{F}}{\partial \mathcal{A}_{i,kl}}$ albeit by scaling the terms from the right-hand side of Eq. A.53 by $\mathcal{A}_{i,kl}$. This leads to,

$$\begin{aligned}\frac{\partial \mathcal{F}}{\partial S_{i,kl}} = \frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi} \big(\mathcal{A}_{i,kl} \odot e(S_{i,kl})\big) \odot \Big\{ \big(\mathcal{A}_{i,kl} \odot e(S_{i,kl})\big)^{-T} \Big\{ \\ \mathrm{Tr}\Big[\big(\big(\overline{G}_i + \overline{H}_i \odot \mathcal{A}_i\big) \odot e(S_i)\big)^T \cdot \mathrm{Cof}(\mathcal{A}_i \odot e(S_i))\Big]\mathcal{I} - \\ \big(\big(\overline{G}_i + \overline{H}_i \odot \mathcal{A}_i\big) \odot e(S_i)\big)^T \mathrm{Cof}(\mathcal{A}_i \odot e(S_i)) \Big\} \\ - \frac{\mathrm{Cof}\big(\mathcal{A}_{i,kl} \odot e(S_{i,kl})\big)}{\Psi} \sum_{j=1}^{D} \mathrm{Tr}\Big[\big(\big(\overline{G}_j + \overline{H}_j \odot A_j\big) \odot e(S_i)\big)^T \mathrm{Cof}\big(A_j \odot e(S_j)\big)\Big] \Big\} \\ + \frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi} e(S_i) \odot \big(\overline{G}_{j,kl} + \mathcal{A}_{i,kl} \odot \overline{H}_{i,kl}\big) \odot \mathrm{Cof}\big(\mathcal{A}_{i,kl} \odot e(S_{i,kl})\big)\end{aligned} \tag{A.54}$$

The final step after deriving all derivative terms is to re-write them within a numerical stable manner following the convention for the first derivatives.

In order to derive a numerical stable equivalent second derivative, we first group matrices together with the following definitions,

$$c = \frac{\partial \mathcal{F}}{\partial \overline{\log|\Psi|}^{\mathcal{L}}} = \frac{1}{\Psi}\sum_{j=1}^{D}\mathrm{Tr}\Big[\big((\overline{G}_j + \overline{H}_j \odot A_j) \odot e(S_j)\big)^T \cdot \mathrm{Cof}\big(A_j \odot e(S_j)\big)\Big] \tag{A.55}$$

$$B_i = \mathcal{A}_i \odot e(S_i) \tag{A.56}$$

$$\overline{K}_i = \overline{G}_i \odot e(S_i) + \overline{H}_i \odot B_i \tag{A.57}$$

$$D_i = \mathrm{Cof}(B_i) \tag{A.58}$$

$$J = \begin{bmatrix} 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \end{bmatrix} \text{s.t.}\, M_i \odot J = J \odot M = M \tag{A.59}$$

This allows the second derivatives to be re-defined as such,

$$\frac{\partial \mathcal{F}}{\partial \mathcal{A}_i} = \frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi} e(S_i) \odot \Big\{B_i^{-T} \cdot \Big(\mathrm{Tr}\Big[\overline{K}_i^T \cdot D_i\Big]\mathcal{I} - \overline{K}_i^T \cdot D_i\Big) + D_i\big(\overline{H}_i - cJ\big)\Big\} \tag{A.60}$$

$$\frac{\partial \mathcal{F}}{\partial S_i} = \frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi} B_i \Big\{B_i^{-T}\Big(\mathrm{Tr}\Big[\overline{K}_i^T D_i\Big]\mathcal{I} - \overline{K}_i^T D_i\Big) - D_i c\Big\} + \frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi}\overline{K}_i \odot D_i \tag{A.61}$$

We first begin by simplifying Eq. A.60. To aid the numerical stability of the second derivatives, we follow the same methodology as the first derivatives and perform SVD decompositions, therefore,

$$B_i \overset{\text{SVD}}{=} U_i \Sigma_i V_i^T \tag{A.62}$$

$$D_i = \det(U_i)\det\big(V_i^T\big)U_i\Gamma_i V_i^T. \tag{A.63}$$

$$M_i = V_i^T \overline{K}_i^T U_i \tag{A.64}$$

Therefore, the numerical stable equivalent for $\frac{\partial \mathcal{F}}{\partial A_{i,kl}}$ becomes,

$$
\begin{aligned}
\frac{\partial \mathcal{F}}{\partial \mathcal{A}_i} &= \frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi} e(S_i) \odot \left\{ B_i^{-T} \cdot \left(\mathrm{Tr}\left[\overline{K}_i^T \cdot D_i \right] \mathcal{I} - \overline{K}_i^T \cdot D_i \right) + D_i \left(\overline{H}_i - cJ \right) \right\} \\
&= \frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi} e(S_i) \det(U_i) \det\left(V_i^T\right) \odot \left\{ U_i \Sigma_i^{-1} V_i^T \left(\mathrm{Tr}\left[\overline{K}_i^T U_i \Gamma_i V_i^T \right] \mathcal{I} - \overline{K}_i^T U_i \Gamma_i V_i^T \right) + U_i \Gamma_i V_i^T \left(\overline{H}_i - cJ \right) \right\} \\
&= \frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi} e(S_i) \det(U_i) \det\left(V_i^T\right) \odot \left\{ U_i \left\{ \underbrace{\Sigma_i^{-1} \left(\mathrm{Tr}\left[M_i \Gamma_i \right] \mathcal{I} - M_i \Gamma_i \right)}_{\Xi_i} - c\Gamma_i \right\} V_i^T + U_i \Gamma_i V_i^T \odot \overline{H}_i \right\} \\
&= \frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi} e(S_i) \det(U_i) \det\left(V_i^T\right) \odot \left\{ U_i \left\{ \Xi_i - c\Gamma_i \right\} V_i^T + U_i \Gamma_i V_i^T \odot \overline{H}_i \right\} \\
&= \overline{\log|\Psi|}^{\mathcal{L}} e(S_i) \, \mathrm{sgn}(\Psi) \det(U_i) \det\left(V_i^T\right) \odot \left\{ U_i \left\{ \Xi_i^{\mathrm{norm}} - c\Gamma_i^{\mathrm{norm}} \right\} V_i^T + U_i \Gamma_i^{\mathrm{norm}} V_i^T \odot \overline{H}_i \right\}
\end{aligned}
\tag{A.65}
$$

Likewise, the numerical stable second derivative for $\frac{\partial \mathcal{F}}{\partial S_{i,kl}}$ becomes,

$$
\begin{aligned}
\frac{\partial \mathcal{F}}{\partial S_i} &= \frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi} B_i \left\{ B_i^{-T} \left(\mathrm{Tr}\left[\overline{K}_i^T D_i \right] \mathcal{I} - \overline{K}_i^T D_i \right) - D_i c \right\} + \frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi} \overline{K}_i \odot D_i \\
&= \frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi} \det(U_i) \det\left(V_i^T\right) \mathcal{A}_i \odot e(S_i) \left\{ U_i \Sigma_i^{-1} V_i^T \left\{ \mathrm{Tr}\left[\overline{K}^T U_i \Gamma_i V_i^T \right] \mathcal{I} - \overline{K}^T U_i \Gamma_i V_i^T \right\} - U_i \Gamma_i V_i^T c \right\} \\
&\qquad + \frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi} \overline{K}_i \odot D_i
\end{aligned}
\tag{A.66}
$$

$$
\begin{aligned}
&= \frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi} \det(U_i) \det\left(V_i^T\right) \left(\mathcal{A}_i \odot e(S_i) \right) \left\{ U_i \underbrace{\left\{ \Sigma_i^{-1} \left(\mathrm{Tr}\left[V_i^T \overline{K}^T U_i \Gamma_i \right] \mathcal{I} - V_i^T \overline{K}^T U_i \Gamma_i \right) \right\}}_{\Xi_i} V_i^T \right\} \\
&\qquad + \frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi} \det(U_i) \det\left(V_i^T\right) \overline{K}_i \odot U_i \Gamma_i V_i^T \\
&= \frac{\overline{\log|\Psi|}^{\mathcal{L}}}{\Psi} \det(U_i) \det\left(V_i^T\right) \left\{ \left(\mathcal{A}_i \odot e(S_i) \right) \odot U_i \left[\Xi_i - c\Gamma_i \right] V_i^T + \overline{K}_i \odot U_i \Gamma_i V_i^T \right\} \\
&= \overline{\log|\Psi|}^{\mathcal{L}} \mathrm{sgn}(\Psi) \det(U_i) \det\left(V_i^T\right) \left\{ \left(\mathcal{A}_i \odot e(S_i) \right) \odot U_i \left[\Xi_i^{\mathrm{norm}} - c\Gamma_i^{\mathrm{norm}} \right] V_i^T + \overline{K}_i \odot U_i \Gamma_i^{\mathrm{norm}} V_i^T \right\}
\end{aligned}
\tag{A.67}
$$

When performing the SVD decomposition, elements of Σ may contain a matrix element whose value is zero. Hence, naïvely multiplying by Σ^{-1} can lead to divide by zero errors, however, we can follow the methodology of defining of Γ to define a numerical stable value of Ξ. We follow Ref. [5] and decompose the definition of Ξ between on-diagonal and off-diagonal matrix elements,

$$
\Xi = \Sigma^{-1} \left(\mathrm{Tr}\left[M\Gamma \right] \mathcal{I} - M\Gamma \right). \tag{A.68}
$$

As Σ is purely diagonal, as is the trace operation, the off-diagonal elements are purely defined by $\Sigma^{-1}M\Gamma$.

$$\begin{aligned}
\Xi_{ij} &= -\sigma_i^{-1} M_{ij} \gamma_j \\
&= -\sigma_i^{-1} M_{ij} \prod_{k \neq j} \sigma_k \\
&= -M_{ij} \prod_{k \neq j,i} \sigma_k
\end{aligned} \tag{A.69}$$

The on-diagonal matrix elements are defined as,

$$\begin{aligned}
\Xi_{ii} &= \sigma_i^{-1} \sum_j M_{jj} \gamma_j - \sigma_i^{-1} M_{ii} \gamma_i \\
&= \sigma_i^{-1} \left(\sum_j M_{jj} \gamma_j - M_{ii} \gamma_i \right) \\
&= \sigma_i^{-1} \left(\sum_{j \neq i} M_{jj} \gamma_j \right) \\
&= \sigma_i^{-1} \left(\sum_{j \neq i} M_{jj} \prod_{k \neq j} \sigma_k \right) \\
&= \sum_{j \neq i} M_{jj} \prod_{k \neq j,i} \sigma_k
\end{aligned} \tag{A.70}$$

These SVD decompositions avoid divide-by-zero errors and provide numerically stable definitions of all functions and derivatives, this allows our code to be run at float precision as opposed to double precision which significantly speeds up our numerical simulations.

Appendix B

Kronecker Factored Approximate Curvature applied towards Variational Monte Carlo

This appendix entails the use of beyond first-order gradient descent methods into higher-order optimisation methods for DNN ansätze. The main use case here will be the optimiser of Ref. [20] - Kronecker Factored Approximate Curvature (KFAC). The full algorithm will be described in full with a novel extensions towards VMC applications.

B.1 Motivation

The main motivation behind the use of KFAC is to fundamentally accelerate convergence of the loss by utilising efficient preconditioning techniques. One such approach is to move from 'steepest' gradient descent to 'natural' gradient descent. The numerical optimiser of the thesis was Adam which performed a modified form of steepest gradient descent which tuned the learning rate for each individual parameter via exponential moving average (EMA) of the first- and second-moment of the gradient [19]. This is a strictly diagonal preconditioning method. Ideally, one would like to use the Hessian to precondition the gradient, however, the Hessian is dense and non-diagonal, and therefore does not scalable favourably. By following Ref. [156] we can precondition the gradient via the Fisher Information Matrix (FIM) which defines the 'natural' gradient. The natural gradient emerges from Information Geometry, and defines the largest change in the loss, $\mathcal{L}$, with respect to the parameters of our DNN ansatz, θ [157]. This defines the natural gradients as,

$$\Delta = \mathcal{F}^{-1}\frac{\partial \mathcal{L}}{\partial \theta} \tag{B.1}$$

with gradient descent under such gradient as,

$$\theta_{t+1} = \theta_t - \alpha \Delta_t \tag{B.2}$$

with t denoting the epoch of the optimisation. In principle, this can allow for significantly more efficient updates by converging the loss more quickly to a local minima - as measured by the number of epochs.

In this work, we follow Refs. [5, 106] by mean-centring the FIM to define the 'Quantum' Fisher Information Matrix (QFIM). This 'mean-centring' emerges due to working with unnormalised Born probability distributions which defines the QFIM as,

$$\mathcal{F}_{ij} = \mathbb{E}_{|\Psi|^2}\left[\frac{\partial \log|\Psi|}{\partial \theta_i}\frac{\partial \log|\Psi|}{\partial \theta_j}\right] - \mathbb{E}_{|\Psi|^2}\left[\frac{\partial \log|\Psi|}{\partial \theta_i}\right]\mathbb{E}_{|\Psi|^2}\left[\frac{\partial \log|\Psi|}{\partial \theta_j}\right] \tag{B.3}$$

$$= \mathbb{E}_{|\Psi|^2}\left[\left(\frac{\partial \log|\Psi|}{\partial \theta_i} - \mathbb{E}_{|\Psi|^2}\left[\frac{\partial \log|\Psi|}{\partial \theta_i}\right]\right)\left(\frac{\partial \log|\Psi|}{\partial \theta_j} - \mathbb{E}_{|\Psi|^2}\left[\frac{\partial \log|\Psi|}{\partial \theta_j}\right]\right)\right] \tag{B.4}$$

with $i = 1, \ldots, M$ and $j = 1, \ldots, M$ representing the arbitrary parameters of the DNN ansätze, and M denoting the total number of parameters within our DNN [5]. One main concern with preconditioning by the QFIM is that the number of matrix elements scales quadratically in the number of parameters of the DNN ansätze and its inverse scales cubically. This scaling is only practical for DNN ansätze of less than 10^4 parameters. In this thesis, the DNN ansätze has $\sim 10^4$ parameters, however, the network for Ref. [5] had nearly 10^6 parameters. Therefore, utilising the entire QFIM is impracticable which leads us to seek an approximation to such preconditioning matrix. This leads us to the KFAC optimiser of Ref. [20] which provides an avenue for such approximation specifically designed for DNN ansätze. This algorithm approximates the QFIM two-fold; firstly via a block-diagonal approximation (BDA) by assuming each layer of the DNN ansätze are independent from one another, and secondly via a Kronecker approximation (KA) to the block-wise approximation to further reduce the numerical complexity. The KA will be discussed before moving to the BDA.

B.2 Kronecker Approximation

Automatic Differentiation (AD) works well in ML libraries due to the ability to efficient compute gradient via reverse-mode AD. As mentioned in Sec. 3.6.3 the gradient of the loss with respect to the parameters is computed by multiplying an adjoint with a local derivative to determine the gradient. In the case of differentiating the log-absolute wavefunction with respect to θ, the gradient would be defined as,

$$\frac{\partial \log|\Psi|}{\partial \theta_i} = \underbrace{\frac{\partial \log|\Psi|}{\partial s_\ell}}_{\mathrm{e}_\ell}\underbrace{\frac{\partial s_\ell}{\partial \theta_i}}_{\mathrm{a}_\ell^T} \tag{B.5}$$

where s is the output of layer, ℓ. The two terms of the gradient are the reverse-mode adjoint, e_ℓ, and local derivative, a_ℓ, respectively. This concept of representing the gradient as the product of two independent terms is referred to as *gradient decomposition*. This has the statistical interpretation of independence between the local derivative and the reverse-mode adjoints. As we utilise the QFIM rather than the FIM, we need to mean-centre gradient decomposition terms which are denoted via carets,

$$\hat{a}_\ell = a_\ell - \mathbb{E}_{|\Psi|^2}[a_\ell], \tag{B.6}$$

$$\hat{e}_\ell = e_\ell - \mathbb{E}_{|\Psi|^2}[e_\ell], \tag{B.7}$$

which generalises the aforementioned statistical independence argument to independence of covariances. This allows Eq. B.4 to be written in terms of the gradient decomposition terms,

$$\mathcal{F}_{\ell\ell'} = \begin{bmatrix} \mathbb{E}_{|\Psi|^2}[(\hat{a}_1 \otimes \hat{e}_1)(\hat{a}_1 \otimes \hat{e}_1)] & \dots & \mathbb{E}_{|\Psi|^2}[(\hat{a}_1 \otimes \hat{e}_1)(\hat{a}_{\ell'} \otimes \hat{e}_{\ell'})] \\ \vdots & \ddots & \vdots \\ \mathbb{E}_{|\Psi|^2}[(\hat{a}_\ell \otimes \hat{e}_\ell)(\hat{a}_1 \otimes \hat{e}_1)] & \dots & \mathbb{E}_{|\Psi|^2}[(\hat{a}_\ell \otimes \hat{e}_\ell)(\hat{a}_{\ell'} \otimes \hat{e}_{\ell'})] \end{bmatrix}. \tag{B.8}$$

This can be viewed as an $L \times L$ matrix represented via gradient decomposition terms. Eq. B.8 can be further approximated via the KA which is defined by as the expectation of outer products approximates to the outer product of expectations,

$$\mathbb{E}_{|\Psi|^2}[(\hat{a}_\ell \otimes \hat{e}_\ell)(\hat{a}_{\ell'} \otimes \hat{e}_{\ell'})] = \underbrace{\mathbb{E}_{|\Psi|^2}[\hat{a}_\ell \hat{a}_{\ell'} \otimes \hat{e}_\ell \hat{e}_{\ell'}] \approx \mathbb{E}_{|\Psi|^2}[\hat{a}_\ell \hat{a}_{\ell'}] \otimes \mathbb{E}_{|\Psi|^2}[\hat{e}_\ell \hat{e}_{\ell'}]}_{\text{Kronecker Approximation}} = \bar{A}_{\ell,\ell'} \otimes \bar{E}_{\ell,\ell'}, \tag{B.9}$$

with $\bar{A}_{\ell,\ell'}$ and $\bar{E}_{\ell,\ell'}$ will be referred to as the Kronecker factors. This approximations allows the inverse to be done much more efficiently as the inverse of outer products is equal to the outer product of inverses. This approximation will differ noticeably from the exact QFIM, however, it can capture the 'coarse' structure which allows it to work well enough in practice to justify its approximation [20]. Additionally, the error of this KA is bounded and is dependent on higher-order cumulants of the Kronecker factors[1]. If the joint distribution between $\bar{A}_{\ell,\ell'}$ and $\bar{E}_{\ell,\ell'}$ closely follows a Gaussian then higher-order cumulants terms will be near zero, and hence lead to a smaller error on the KA [20].

B.3 Block-Diagonal Approximation

The intractability of preconditioning via a full $M \times M$ matrix leads to one solution to alleviate this issue - block diagonal approximation (BDA). This solution has the statistical approximation of assuming that different layers within the network act independently of each other. In the case of DNN ansätze we can take this BDA approach by separating the

[1] Cumulants are generalisations of means and covariance to higher orders. The mean (covariance) is the first (second) order cumulant, and measures if the interaction is explicitly of a given order, k. More information regarding this can be found in Sec. 2.1 of Ref. [20]

preconditioning matrix between layers and ignore preconditioning between layers which has the interpretation that layers act ‘independently’ of each other. In the case of DNN ansätze, the number of parameters with a given layer can vary from 100s to 100,000s of parameters and hence even this approximation can be impractical. Hence, we utilise the KA of Sec. B.2 to reduce the complexity of the BDA approximation. The diagonal approximate QFIM is defined as,

$$\breve{F}_{\ell,\ell'} = \mathrm{diag}\left(F_{1,1}\dots F_{\ell,\ell}\right) \approx \mathrm{diag}\left(\bar{A}_{1,1}\otimes\bar{E}_{1,1},\dots,\bar{A}_{\ell,\ell}\otimes\bar{E}_{\ell,\ell}\right), \tag{B.10}$$

where the inverse is trivially defined as,

$$\breve{F}_{\ell,\ell'} = \mathrm{diag}\left(F^{-1}_{1,1}\dots F^{-1}_{\ell,\ell}\right) \approx \mathrm{diag}\left(\bar{A}_{1,1}\otimes\bar{E}_{1,1},\dots,\bar{A}^{-1}_{\ell,\ell'}\otimes\bar{E}^{-1}_{\ell,\ell'}\right). \tag{B.11}$$

This results in the preconditioning formula of Eq. B.1 being defined as,

$$\Delta_\ell = \breve{F}^{-1}_{\ell,\ell}\frac{\partial\mathcal{L}}{\partial\theta_\ell} = \left(\bar{A}^{-1}_{\ell,\ell}\otimes\bar{E}^{-1}_{\ell,\ell}\right)\frac{\partial\mathcal{L}}{\partial\theta_\ell}, \tag{B.12}$$

with ℓ denoting a layer of the DNN. As the preconditioning is done via a KA-equivalent QFIM we need to utilise the mathematical identity,

$$(A\otimes B)\,\mathrm{vec}(X) = \mathrm{vec}\left(BXA^T\right). \tag{B.13}$$

where A and B represents the Kronecker factors, X is the steepest gradient, with vec being the vectorisation operator that takes a matrix and iteratively stacks its columns against each other to define a new vector [20] . This results in the preconditioning of Eq. B.12 being defined as,

$$\Delta_\ell = \bar{E}^{-1}_{\ell,\ell}\frac{\partial\mathcal{L}}{\partial\theta_\ell}\bar{A}^{-1}_{\ell,\ell}, \tag{B.14}$$

which defines the preconditioning as the matrix-multiplication of two smaller matrices rather than one larger matrix [2].

One way for further improvement upon the Kronecker factors is to take an EMA of the these factors over previous epochs via,

$$\bar{A}_{(\ell,\ell),t} \coloneqq (1-\epsilon)\bar{A}_{(\ell,\ell),t} + \epsilon\bar{A}_{(\ell,\ell),t-1}, \tag{B.15}$$

with the current estimate weighted by (1-ϵ), and the previous epoch weighted as ϵ [20]. This EMA scheme allows for the Kronecker factors of previous epochs to be stored efficiently. The non-equal weighting allows the effective Kronecker factors to gradually change over simulation time so as to be more representative of the local loss landscape [20].

[2] The transpose on $\bar{A}^{-1}_{\ell,\ell}$ can be ignored due to this matrix being a positive-semi definite matrix whose transpose equals itself.

The loss function of this work is the expectation of the local energy, however, the reverse-mode adjoints for this preconditioning requires the loss function to be the log-absolute of the wavefunction. In order to remedy this issue, we compute two loss functions and simply cache all required intermediate values. These 'intermediate' values are the forward activations for all hidden nodes of the network, and backward sensitivities (reverse-mode adjoints) which were defined in Eq. B.5. In fact, as the forward activations of all layers are independent of the loss function we can cache from the local energy calculation, and calculate the appropriate reverse-mode adjoints for this preconditioning. This results in the correct expectations as all many-body positions are sampled from the same Born probability distribution of the wavefunction which allows for such caching to give the correct expectation without re-computing any intermediate values [54]. As the preconditioning has now been fully described, we turn our efforts to numerical stability of the previously mentioned preconditioning method via the use of damping and regularisation.

B.4 The Motivation of Damping

In order for second-order methods to work in practice, damping and other regularisation methods are needed to stop preconditioned gradients from overshooting local minima [20]. The information geometry perspective of Ref. [157] only holds for infinitesimally small updates within the parameter space, and hence defeats the main motivation of taking *large* updates with preconditioned methods. Therefore, the natural gradient should be referred to more as an *optimal* direction as opposed to an *optimal* update.

This motivates the use of re-scaling our natural gradient within a Quadratic Model (QM) wherein we 'trust' our updates to accurately reflect the loss landscape of our DNN ansatz. The rescaling can be defined as,

$$\delta = \alpha\Delta \tag{B.16}$$

with the optimal update, δ^*, being define by re-scaling Δ within a QM, M, such that we minimise said QM, i.e. $\text{argmin}_\delta\,(M)$, with QM being defined as,

$$M(\delta) = \frac{1}{2}\delta^T F\delta + \nabla\mathcal{L}(\theta)^T + \mathcal{L}(\theta). \tag{B.17}$$

This term can be seen as a second-order Taylor expansion of the loss, $\mathcal{L}$, with the second-order component replaced with the QFIM. In Ref. [20], the QM is key to getting the natural gradient to work well in practice as it efficiently approximates the Hessian of loss function for *that* given problem. In Refs. [5, 6, 64] no QM is used which leaves further improvements for the KFAC optimiser for VMC applications; such improvements will be discussed in Sec. B.11.

B.5 Factored Tikhonov Regularisation

Tikhonov Regularisation (TR) is a method for regularising curvature matrices by including a non-zero damping term, λ, along the diagonal of such matrices, i.e. $\mathcal{F} \mapsto \mathcal{F} + \lambda\mathcal{I}$. This directly ties into the aforementioned 'trust' region where there magnitude of the regularisation is inversely proportional to how much we trust our QM. This aids the minimisation in two ways; mitigates inaccuracies which arise from the KA, and attenuate shortcomings in representing the second-order components of the QM as the QFIM instead of the exact Hessian of the loss. The QFIM and Hessian are only equivalent over small distances which leads to large damping strengths[3].

The damping strength, λ, is preferred to be initialised at a larger magnitude rather than a smaller magnitude in order to start with a more conservative trust-region to avoid overshooting local minima. One drawback of using a large damping-strength is that it shifts the eigenspectrum of the preconditioning matrix such that eigenvalues with small magnitudes are 'washed out'. This can have the effect of degrading the asymptotic convergence of the loss if the trust-region becomes too small and the parameters get stuck within a local region of the parameter space.

Naïve regularisation of the Kronecker product breaks the efficient method of inverting our Kronecker product,

$$((A \otimes E) + \lambda(\mathcal{I} \otimes \mathcal{I}))^{-1} \neq (A + \lambda\mathcal{I})^{-1} \otimes (E + \lambda\mathcal{I})^{-1}. \tag{B.18}$$

We follow Ref. [20] and apply a factorised Tikhonov regularisation (FTR) factor to each Kronecker factor individually,

$$\left(\left(A_{ii} + \pi_i\sqrt{\lambda}\right) \otimes \left(E_{ii} + \frac{\sqrt{\lambda}}{\pi_i}\right)\right) \tag{B.19}$$

where π_i is a scalar constant. This factorised form is also an approximation, and hence when Eq. B.19 is fully expanded it differs from Eq. B.18 via a residual,

$$\left(A_{\ell,\ell} + \pi_\ell\sqrt{\lambda} \otimes E_{\ell,\ell} + \frac{\sqrt{\lambda}}{\pi_\ell}\right) = A_{\ell,\ell} \otimes E_{\ell,\ell} + \lambda(\mathcal{I} \otimes \mathcal{I}) + \underbrace{\pi_\ell\sqrt{\lambda}\mathcal{I} \otimes E_{\ell,\ell} + \frac{\sqrt{\lambda}}{\pi_\ell}\mathcal{I} \otimes A_{\ell,\ell}}_{\text{residual}}. \tag{B.20}$$

The residual of this expression can be minimising via minimising for a given matrix norm[4]. For a PSD matrix, the trace-norm is equal to the trace of the matrix. This defines

[3] This most likely explains why Refs. [5, 6, 64] don't utilise the re-scaling to a QM technique of Ref. [20] as this reference is suited to a Supervisor task with a loss function whose Hessian is efficiently approximated via the FIM itself. In the case of VMC calculations, this property no longer holds.

[4] Ref. [20] states this via use of the *triangle inequality* which defines the given norm of a sum of matrices is at most equal to sum of the matrix norm of the individual matrices, i.e. $||\bar{A} + \bar{E}|| \leq ||\bar{A}|| + ||\bar{E}||$

an optimal value of π_ℓ whose value we seek to minimise such that the error imposed by this FTR approach is minimal. This defines π_ℓ as,

$$\pi_\ell = \sqrt{\frac{||\bar{A}_{\ell,\ell} \otimes \mathcal{I}||_\nu}{||\mathcal{I} \otimes \bar{G}_{\ell,\ell}||_\nu}} = \sqrt{\frac{\mathrm{Tr}\left[\bar{A}_{\ell,\ell}\right]}{\mathrm{Tr}\left[\bar{E}_{\ell,\ell}\right]} \frac{\dim\left(\bar{E}_{\ell,\ell}\right)}{\dim\left(\bar{A}_{\ell,\ell}\right)}}. \tag{B.21}$$

where, $||\cdot||_\nu$ denotes the matrix-norm of type, ν, $\mathrm{Tr}[\cdot]$ denotes the trace operator, and $\dim(\cdot)$ denotes the dimensionality of a given matrix which in this case equates to the number of input (output) nodes of layer for $\bar{A}_{\ell,\ell}$ ($\bar{E}_{\ell,\ell}$)[5]. This has the physical interpretation of increasing π_ℓ as the number of neurons in a given layer, ℓ, increase. The other term in the residual is the damping strength, λ. The initial value of λ was set high to offset inaccuracies of the QM, however, in the ideal setting λ should change as the parameters update within the parameter space in order to respect the given curvature of the loss landscape. Therefore, this motivates updating λ on-the-fly via some form of heuristic adaption.

One such way is the Levenberg-Marquardt Adjustment (LMA) rule which is an adaptive heuristic procedure which can adaptively change λ during the energy minimisation process. Originally within Ref. [135], Tikhonov damping was used albeit without adaption. This can become problematic with a high damping strength significantly slowing down convergence, and a low damping strength providing little regularisation. This motivates a heuristic procedure which can adaptively change λ to better suit the local curvature, and thereby the appropriate regularisation. The LMA rule is defined as,

$$\rho = \frac{\mathcal{L}(\theta_t + \delta_t) - \mathcal{L}(\theta_t)}{\mathcal{M}(\delta_t) - \mathcal{M}(0)}, \tag{B.22}$$

where $\mathcal{L}(\theta_t)$ denotes the loss for the parameters at epoch, t, and $\mathcal{L}(\theta_t + \delta_t)$ denotes the loss under the preconditioned update, δ_t. The quantities, $\mathcal{M}(\delta_t)$ and $\mathcal{M}(0)$ denote the value of the QM for the preconditioned updates and no updates[6] respectively. The term ρ is referred to as the 'reduction-ratio' because it's a ratio of the change as measured by the loss against what's measured by the QM. This term can efficiently quantify how effective the QM is at predicting the change in the loss. This information can be used to tune the value of λ such that the QM accurately predicts the change in the loss which updates Eq. B.17 with $\mathcal{F} \mapsto \mathcal{F} + \lambda\mathcal{I}$ in order to account for the regularisation. As the quality of the QM is measured by the reduction-ratio, it therefore stands to bifurcate ρ into two effective domains; $\rho < 1$, and $\rho \geq 1$.

In the former case, the physical interpretation is that the predicted change in the loss under the QM is *larger* than the actual change loss under the loss function,

$$\mathcal{L}(\theta_t + \delta_t) - \mathcal{L}(\theta_t) < \mathcal{M}(\delta_t) - \mathcal{M}(0) \tag{B.23}$$

[5] It also denotes the ratio of the average eigenvalues of each component of the gradient decomposition which relates the innate curvature of the layer and future layers w.r.t the loss

[6] The value of the QM for $\mathcal{M}(0)$ corresponds to passing a zero vector, and computing the QM's value. This reduces Eq. B.17 to just the loss at the current epoch, $\mathcal{L}(\theta_t)$

and hence, we should increase the damping strength in order to mitigate the QM's shortcomings. This has the effect of restricting future updates, as measured by the Euclidean norm, to align within a trust-region wherein updates are trusted. In the latter case, the physical interpretation is that the predict change in the loss under the QM is *smaller* than the actual change loss,

$$\mathcal{L}(\theta_t + \delta_t) - \mathcal{L}(\theta_t) \geq \mathcal{M}(\delta_t) - \mathcal{M}(0) \tag{B.24}$$

which highlights that the damping strength is too conservative in the updates, and should be reduced to allow for larger updates, as measured by the Euclidean norm [158]. The heuristic scheme of the LMA rule can be succinctly defined as,

$$\lambda_{t+1} = \begin{cases} \omega\lambda_t, & \text{if } \rho > 3/4. \\ \omega^{-1}\lambda_t, & \text{if } \rho < 1/4. \\ \lambda_t, & \text{otherwise.} \end{cases} \tag{B.25}$$

where ω is the 'decay-constant' which is a real-valued constant, $0 < \omega < 1$. In Ref. [158] ω was set to equal $2/3$, however, it was noted in Ref. [159] that having a small decay-constant within a stochastic setting can lead to large oscillations of the damping strength. It motivates the use of a 'softer' decay-constant closer to unity such that λ incrementally changes over a larger number of epochs rather than rapidly changing over fewer epochs. This reduces the chances of the damping strength oscillating during energy minimisation [159].

In order to further quantify how λ affects convergence it makes sense to look at both extremes of the magnitude of λ, e.g. $\lambda \mapsto 0$ and $\lambda \mapsto \infty$. The damping of the QFIM is equivalent to adding a scalar, λ, to the diagonal component of the QFIM. By applying an eigendecomposition to the QFIM leads to $\mathcal{F} \overset{\text{eig}}{=} Q\Lambda Q^{-1}$, where Λ is a diagonal matrix with strictly positive eigenvalues[7], and Q are the associated eigenvectors of the eigendecomposition. Adding the regularisation, λ, is directly equivalent to shifting all the eigenvalues of the preconditioning matrix by λ,

$$\mathcal{F} \overset{\text{eig}}{=} \left(Q\Lambda Q^{-1}\right) + \lambda\mathcal{I} = Q(\Lambda + \lambda\mathcal{I})Q^{-1}. \tag{B.26}$$

In the case of $\lambda \mapsto 0$, the eigenvalues remain unchanged and hence the curvature is unaffected. In the case of $\lambda \mapsto \infty$, all eigenvalues tend to λ and hence all curvature is effectively destroyed as the regularised QFIM will tend to an identity matrix scaled by λ. This results in the update being equal to SGD with the learning rate set to the reciporal of the damping-strength, i.e. $\lambda \mapsto \infty, (\mathcal{F} + \lambda\mathcal{I})^{-1} g \mapsto \frac{g}{\lambda}$ where g denotes the tangential gradient from SGD.

One can summarise both events via multiplying the non-regularised update of the parameters by $\frac{\lambda_i}{\lambda_i + \lambda}$ where λ_i denotes the eigenvalue of the i^{th} parameter. In the ideal case

[7] The eigenvalues here are strictly positive only in the case of a positive-semi definite matrix which the QFIM is

of $\lambda \mapsto 0$ the update is unchanged whereas in the case of strong regularisation, $\lambda \mapsto \infty$ this multiple becomes 0 which effectively strands the parameters at their current location within the parameter space thereby abruptly stopping convergence [158].

This LMA rule, although simple to implement, is quite expensive as it effectively doubles the walltime for epoch due to the re-calculation of the loss with the natural gradient update. This initial caveat can be mitigated via amortising this operation, i.e. only compute the LMA every T_1 epochs and cache the previous value to utilise for further updates.

B.6 Separate Damping of the Approximate Fisher with Greedy Adjustment

So far, the regularisation discussed has been applied to the *exact* QFIM, however, no regularisation how been applied to our KA QFIM. In Ref. [20] it states that ideally one should decouple the regularisation of the exact QFIM and the approximate QFIM as these are two different quantity whose regularisation should differ. The role of λ is to ensure that updates under the natural gradient lie within a trust-region wherein updates approximate the curvature well. The role of γ differs to λ as its magnitude is directly related to the re-scaling of the natural gradients. Ideally, the re-scaling should be equal to unity such that the natural gradients are optimal without having to re-scale them. Of course, this never occurs in practice, and hence requires additional damping in order to compensate for inaccuracy of the KA QFIM. This summaries to,

$$\mathcal{F} \mapsto \mathcal{F} + \lambda\mathcal{I} \tag{B.27}$$

$$\breve{\mathcal{F}} \mapsto \breve{\mathcal{F}} + \gamma\mathcal{I} \tag{B.28}$$

with $\mathcal{F}$ and $\breve{\mathcal{F}}$ being the exact QFIM and approximate (KA) QFIM respectively. We follow by the inspiration of the LMA in adapting λ to adapt γ, however, the concept of a reduction-ratio isn't applicable here. Therefore, we default to a 'greedy' adaption of picking candidate values of γ; $\{\gamma, \omega_2\gamma\gamma/\omega_2\}$ with $\omega_2 \in [0,1)$ being a decay-constant [20]. The initial value of γ is initialised to $\sqrt{\lambda+\eta}$ with η being a ℓ_2-regularisation term that can be inferred as a minimum damping that's set to 10^{-4}. The optimal value of γ is determined by computing an associated score via the QM with the optimal regularisation, γ^*, having the lowest QM value.

This computation can become expensive to compute as it scales linearly with the number of γ-candidates. Therefore, we can follow the same strategy of λ by amortisation with computing this 'optimal' γ regularisation of the KA QFIM every T_2 epochs. As the performance of γ is quantified by the QM, the value of the QM can essentially be computed for free as it's previously computed within the LMA rule.

B.7 Re-scaling natural gradients

As previously mentioned the natural gradient is better thought of as an *optimal* direction within the parameter space rather than an *optimal* step under gradient descent. Therefore, this motivates calculating how far one should step under the natural gradient. This concept is referred to as *re-scaling*. As the QM is used prolifically throughout this optimiser, it makes sense to utilise this information when performing re-scaling. The natural gradient is denoted Δ, and its re-scaling is performed via multiplying it by a scalar constant, α. This defines a re-scaled natural gradient, under a given QM, which is denoted δ. The QM of this re-scaled gradient is defined as,

$$\mathcal{M}(\delta) = \mathcal{M}(\alpha\Delta) = \frac{\alpha^2}{2}\Delta^T (F + \lambda\mathcal{I})\Delta + \alpha\nabla\mathcal{L}(\theta)^T \Delta + \mathcal{L}(\theta). \tag{B.29}$$

The optimal value of α such that $\mathcal{M}(\alpha\Delta)$ is minimised is defined as the QM equalling the Loss, i.e. $\mathcal{M}(\alpha\Delta) - \mathcal{L}(\theta) = 0$.

$$0 = \frac{\alpha^2}{2}\Delta^T (F + \lambda\mathcal{I})\Delta + \alpha\nabla\mathcal{L}(\theta)^T \Delta \tag{B.30}$$

$$\alpha^* = \frac{-\nabla\mathcal{L}(\theta)^T \Delta}{\Delta^T (F + \lambda\mathcal{I})\Delta}. \tag{B.31}$$

An understated takeaway from Eq. B.31 is that the re-scaling constant here is effectively the optimal learning rate under the QM at the current location within the parameter space. An additional benefit of such re-scaling is that the learning rate isn't limited to strictly positive values with first order optimisation methods. This gives KFAC the innate ability to 'step-back' if it were to overshoot a local minima. Furthermore such heuristic approaches to defining the learning rate forgo any form of hyperparameter optimisation and directly allow for the optimiser to chose a learning rate such that the loss is minimised under the current QM. This shifts the importance to ensuring that the QM accurately predicts the correct curvature within the loss landscape.

The most expensive component of this operation is the $\Delta^T \mathcal{F}\Delta$ term which is a nested loop over both indices of the QFIM, i.e. $\Delta^T F\Delta = \sum_{ij} \Delta_i^T F_{ij}\Delta_j$. As the QFIM is a PSD matrix we can decompose it in its original outer product and reduce the complexity of the operation from being quadratic in the number of parameters to being linear in the number of parameters. This can be done via exploiting associativity of matrix multiplication,

$$\begin{aligned}\Delta_i^T \mathcal{F}_{ij} \Delta_j &= \sum_{ij} \Delta_i^T \mathbb{E}_{|\Psi|^2}\left[\left(\frac{\partial \log|\Psi|}{\partial \theta_i} - \mathbb{E}_{|\Psi|^2}\left[\frac{\partial \log|\Psi|}{\partial \theta_i}\right]\right)\left(\frac{\partial \log|\Psi|}{\partial \theta_j} - \mathbb{E}_{|\Psi|^2}\left[\frac{\partial \log|\Psi|}{\partial \theta_j}\right]\right)\right]\Delta_j \\ &= \sum_{ij} \mathbb{E}_{|\Psi|^2}\left[\Delta_i^T\left(\frac{\partial \log|\Psi|}{\partial \theta_i} - \mathbb{E}_{|\Psi|^2}\left[\frac{\partial \log|\Psi|}{\partial \theta_i}\right]\right)\left(\frac{\partial \log|\Psi|}{\partial \theta_j} - \mathbb{E}_{|\Psi|^2}\left[\frac{\partial \log|\Psi|}{\partial \theta_j}\right]\right)\Delta_j\right] \\ &= \mathbb{E}_{|\Psi|^2}\left[\sum_i \Delta_i^T\left(\frac{\partial \log|\Psi|}{\partial \theta_i} - \mathbb{E}_{|\Psi|^2}\left[\frac{\partial \log|\Psi|}{\partial \theta_i}\right]\right)\sum_j\left(\frac{\partial \log|\Psi|}{\partial \theta_j} - \mathbb{E}_{|\Psi|^2}\left[\frac{\partial \log|\Psi|}{\partial \theta_j}\right]\right)\Delta_j\right].\end{aligned} \tag{B.32}$$

This allows a naïve computation of the squared-norm which is quadratic in the number of parameters to be decomposed in to a linear operation of 2 dot-products over the un-scaled natural gradient with the mean-centred derivative of the log-absolute of the wavefunction with respect to its parameters. This 'trick' allows for practical use of such re-scaling methods [20].

B.8 Re-scaling natural gradients with momentum

One extension to first order optimisation methods is to include a 'momentum' term which accumulates previous values of the gradient and uses this information to accelerate convergence [160]. However, this isn't directly applicable for this method as the natural gradient isn't an optimal step only an optimal direction. Therefore, it makes sense to utilise the previous re-scaling technique by also incorporating momentum. This re-defines the re-scaled natural gradient with momentum at epoch t as,

$$\delta_t = \alpha \Delta_t + \mu \delta_{t-1} \tag{B.33}$$

where α and μ are the optimal learning rate and momentum constant respectively. In the case of the first epoch, δ_t is defined as a zero-vector so as to not zero-bias any re-scaled updates with momentum. The methodology of Sec. B.7 can be re-applied with the momentum constant to define an optimal $\alpha^* - \mu^*$ pair such that the QM is minimised which is defined by,

$$\begin{bmatrix}\alpha^* \\ \mu^*\end{bmatrix} = -\begin{bmatrix}\Delta^T \mathcal{F}\Delta + \lambda\|\Delta\|^2 & \Delta^T \mathcal{F}\delta_{t-1} + \lambda\Delta^T\delta_{t-1} \\ \Delta^T \mathcal{F}\delta_{t-1} + \lambda\Delta^T\delta_{t-1} & \delta_{t-1}^T \mathcal{F}\delta_{t-1} + \lambda\|\delta_{t-1}\|^2\end{bmatrix}^{-1}\begin{bmatrix}\nabla\mathcal{L}(\theta)^T\,\Delta \\ \nabla\mathcal{L}(\theta)^T\,\delta_{t-1}\end{bmatrix}. \tag{B.34}$$

This computation is again dominated by the vector-matrix-vector products, however, the 'trick' of Eq. B.32 can be utilised to perform this computation in linear time. This leads to an optimal learning rate and optimal momentum constant which are both real valued. This has the physical interpretation of the learning rate being able to innately step backwards if it minimises the loss, and the momentum constant has the intuitive ability to either accelerate or decelerate convergence such the the loss is minimised more efficiently. In principle, this allows for significantly more efficient optimisation, as measured by the loss, than first-order methods. The QM definition must also be updated in order to accommodate for the momentum term which redefines the QM to,

$$\begin{aligned}\mathcal{M}(\delta) = \frac{\alpha^2}{2}\Delta^T\mathcal{F}\Delta + \frac{\mu^2}{2}\delta_{t-1}^T\mathcal{F}\delta_{t-1} + \alpha\mu\Delta^T\mathcal{F}\delta_{t-1} \\ + \alpha\nabla\mathcal{L}(\theta)^T\Delta + \mu\nabla\mathcal{L}(\theta)^T\delta_{t-1} + \mathcal{L}(\theta)\end{aligned} \tag{B.35}$$

B.9 Restricting natural gradients via a Kullback-Lieber measure

The original KFAC paper of Ref. [20] is targeted towards a specific branch of ML - Supervised Learning (SL). In this use case the goal is to get a model to fit input-output pairs under some data distribution [76]. The associated loss function in this setting could be a mean-squared error (MSE) function whose global minimum is zero, however, for a quantum many-body Hamiltonian the global minimum is unknown. In the use case of VMC calculations we're minimising an expectation value which is bounded by a Variational principle instead of fitting a specific value. This falls under the concept of Reinforcement Learning (RL) in which the expectation value of the Hamiltonian is used as a reinforcement signal to tune the model's performance. Additionally, as our input data is conditionally dependent on our model due to our many-body positions being sampled from the Born probability distribution of the wavefunction this specific application of DNNs to VMC has been referred to as 'self-play' RL [5].

In order to remedy this issue of applying an optimiser specifically designed to SL tasks to an RL task motivates the use of additional constraints to ensure the optimiser performance works well in practice. One such method of constraining updates is to measure how much the parameters change per epoch. The measure chosen is the Kullback-Lieber (KL) divergence which is asymmetric measure between two arbitrary probability distributions. The KL-divergence equals to the squared-norm of the exact QFIM under the re-scaled updates in this setting, i.e. $\frac{1}{2}\delta^T\mathcal{F}\delta$. In Ref. [161] it's stated one can compute this efficiently via dot-products of the raw gradient and un-scaled natural gradients[8], however, it didn't work in my use case more probably due to the fact that KFAC is primarily applied to Linear layers, and not to the layers of this thesis. The KL-divergence can effectively be used as a metric to 'clip' how far the optimiser can step to allow for only so-much KL-divergence to change per epoch. Such a methodology can be succinctly defined as,

$$\tilde{\delta} = \min\left(\delta_{max}, \sqrt{\frac{c}{\delta^T F \delta}}\right), \tag{B.36}$$

[8] This trick allows for the squared-norm term, $\delta^T\mathcal{F}\delta$, to be computed in linear time with respect to the number of parameters. It is explained in full in Sec. 5 of Ref. [161]

with $\tilde{\delta}$ being the KL-clipping factor, δ_{max} is a scalar to denote the maximum allowed change of KL-divergence, and c is the 'norm-constraint' that limits the amount of KL-divergence changing per epoch under the current updates, δ whose value lies within the range 10^{-3} to 10^{-4}. The $\tilde{\delta}$ clipping is multiplied by the effective learning rate to enforce the aforementioned KL-clipping.

B.10 KFAC Pseudocode

A pseudocode for the VMC energy minimisation of this thesis for the KFAC optimiser is shown below. This pseudocode gives a broad overview of the entire algorithmic process and highlights all 'hyperparameters' of the optimiser. All functions are denoted bold, and have their associated equation referenced in the inline comment. In order to improve efficiency of the optimizer, we amortise certain operations within the algorithm. These three sections: updating λ, updating γ, and inverting the KFs have their associated amortised constants, T_1, T_2, and T_3 respectively. In addition, we cache intermediate values of the network via computation objects called hooks. The forward activations and backward sensitivities are cached via `register_foward_pre_hook` and `register_backward_full_hook` respectively.

Algorithm 1 The KFAC algorithm for VMC

```
Require: $\alpha_0, \mu_0, \underline{rescaling}$    // learning rate, momentum, re-scaling boolean
Require: $\lambda_0, \eta, \gamma = \sqrt{\lambda_0 + \eta}$    // Regularisation strengths and $\ell_2$ regularisation
Require: $\lambda_{min}, \lambda_{max}, \gamma_{min}, \gamma_{max}$    // The limits of regularisation
Require: $\omega_1, \omega_2$    // The decay constants for $\lambda$ and $\gamma$ respectively
Require: $t > 0, t_{max} > 0$    // epoch, max epoch
Require: $T_1 > 0, T_2 > 0, T_3 > 0$    // Amortisation constants
Require: $\epsilon_0 = 0, \epsilon_{max} = 0.95$    // KF EMA coefficient with its max value
1: while t < $t_{max}$ do
2:     $X \sim |\Psi|^2$    // Sample many-body positions with MH sampler
3:     $E_L = \frac{\hat{H}\Psi(X)}{\Psi(X)}$    // Compute local energy
4:     $\tilde{E}_L \leftarrow$ Clip$(E_L)$    // Clip local energy
5:     $\mathcal{L}_{\langle E \rangle} \leftarrow 2\mathbb{E}_{|\Psi|^2}\left[\perp\left(\tilde{E}_L - \mathbb{E}_{|\Psi|^2}\left[\tilde{E}_L\right]\right)\log|\Psi|\right]$    // Loss of energy
6:     $\theta_\nabla \leftarrow \nabla\mathcal{L}_{\langle E \rangle}$    // reverse-mode AD for gradients

7:     Cache$(\theta_\nabla)$    // Cache energy-gradients
8:     $\theta_\nabla \leftarrow 0$    // Clear gradient buffers
9:     $\mathcal{L}_{\langle \log|\Psi| \rangle} \leftarrow \mathbb{E}_{|\Psi|^2}[\log|\Psi|]$    // Compute auxiliary loss for hooks
10:    $\theta_\nabla \leftarrow \nabla\mathcal{L}_{\langle \log|\Psi| \rangle}$    // reverse-mode AD to populate hooks
11:    $a \leftarrow \frac{\partial s}{\partial \theta}$    // Populate forward hooks
12:    $e \leftarrow \frac{\partial \log|\Psi|}{\partial s}$    // Populate backward hooks
13:
14:    $\theta_\nabla \leftarrow \nabla\mathcal{L}_{\langle E \rangle}$    // Update gradient buffers with energy gradient from cache
15:
16:    for all layers $\ell$ in DNN do
17:        $\epsilon_t \leftarrow \min\left(1 - \frac{1}{t}, \epsilon_{max}\right)$    // Update KF EMA
18:
19:        $\hat{a}_\ell \leftarrow a_\ell - \mathbb{E}_{|\Psi|^2}[a_\ell]$    // Mean-centre forward activations
20:        $\bar{A}_{\ell,\ell} \leftarrow \mathbb{E}_{|\Psi|^2}\left[a_\ell a_\ell^T\right]$    // Update forward KF
21:        $\bar{A}_{\ell,\ell} \leftarrow \epsilon_t \bar{A}_{(\ell,\ell),t} + (1-\epsilon_t)\bar{A}_{(\ell,\ell),t-1}$    // EMA of forward KF
22:
23:        $\hat{e}_\ell \leftarrow e_\ell - \mathbb{E}_{|\Psi|^2}[e_\ell]$    // Mean-centre backward sensitivities
24:        $\bar{E}_{\ell,\ell} \leftarrow \mathbb{E}_{|\Psi|^2}\left[e_\ell e_\ell^T\right]$    // Update backward KF
25:        $\bar{E}_{\ell,\ell} \leftarrow \epsilon_t \bar{E}_{(\ell,\ell),t} + (1-\epsilon_t)\bar{E}_{(\ell,\ell),t-1}$    // EMA of backward KF
26:
27:        if $t$ mod $T_2 = 0$ then    // Update $\gamma$
28:            for $\gamma \in \{\gamma_t, \omega_2\gamma_t, \gamma_t/\omega_2\}$ do
29:                if $t$ mod $T_3 = 0$ then    // Update KF inverses
30:                    $\pi_\ell \leftarrow$ FactoredTikhonov$\left(\bar{A}_{\ell,\ell}, \bar{E}_{\ell,\ell}\right)$    // Compute $\pi_\ell$ (Eq. B.21)
31:
32:                    $\bar{A}_{\ell,\ell}^{-1} \leftarrow$ CholeskyInverse$\left(\bar{A}_{\ell,\ell} + \gamma\pi_\ell\mathcal{I}\right)$
33:                    $\bar{E}_{\ell,\ell}^{-1} \leftarrow$ CholeskyInverse$\left(\bar{E}_{\ell,\ell} + \frac{\gamma}{\pi_\ell}\mathcal{I}\right)$
34:                end if
35:
36:                $\Delta = \bar{E}_{\ell,\ell}^{-1}\theta_\nabla\bar{A}_{\ell,\ell}^{-1}$    // un-scaled natural gradients
37:
```

Algorithm 2 The KFAC algorithm for VMC. Cont.

```
38:                 if $rescaling$ = True then
39:                     $\alpha^*, \mu^*, \mathcal{M}(\delta) \leftarrow$ CalcRescaling$[\mathcal{M}(\delta)]$        // Eqs. B.29, B.35
40:                     $\alpha \leftarrow \alpha^*$                                  // Computed $\mathcal{M}(\delta)$ for free
41:                     $\mu \leftarrow \mu^*$
42:                 else
43:                     $\alpha \leftarrow \alpha_0$                 // Arb. learning-rate & momentum constants
44:                     $\mu \leftarrow \mu_0$
45:                     $\mathcal{M}(\delta) \leftarrow$ CalcQuadraticModel$(\Delta_t, \delta_t)$          // Not free here!
46:                 end if
47:
48:                 $\delta_t = \alpha\Delta_t + \mu\delta_{t-1}$                // Re-scale the natural gradients to QM
49:
50:                 Cache$(\delta_t, \mathcal{M}(\delta_t))$    // Cache $\delta_t$ and its associated $\mathcal{M}(\delta_t)$ for $\gamma \in \Gamma$
51:             end for                                                     // End of $\Gamma$-loop
52:             $\delta_t \leftarrow$ Argmin$_{\mathcal{M}(\delta)}(\delta_t)$          // Select $\delta_t$ with the lowest $\mathcal{M}(\delta_t)$ value
53:         else
54:             if $t$ mod $T_3 = 0$ then                                    // Update KF inverses
55:                 $\pi_\ell \leftarrow$ FactoredTikhonov$\left(\bar{A}_{\ell,\ell}, \bar{E}_{\ell,\ell}\right)$         // Compute $\pi_\ell$ (Eq. B.21)
56:
57:                 $\bar{A}^{-1}_{\ell,\ell} \leftarrow$ CholeskyInverse$\left(\bar{A}_{\ell,\ell} + \gamma\pi_\ell\mathcal{I}\right)$
58:                 $\bar{E}^{-1}_{\ell,\ell} \leftarrow$ CholeskyInverse$\left(\bar{E}_{\ell,\ell} + \frac{\gamma}{\pi_\ell}\mathcal{I}\right)$
59:             end if
60:
61:             $\Delta = \bar{E}^{-1}_{\ell,\ell}\theta_\nabla\bar{A}^{-1}_{\ell,\ell}$                               // un-scaled natural gradients
62:
63:             if $rescaling$ = True then
64:                 $\alpha^*, \mu^*, \mathcal{M}(\delta) \leftarrow$ CalcRescaling$[\mathcal{M}(\delta)]$            // Eqs. B.29, B.35
65:                 $\alpha \leftarrow \alpha^*$                                      // Computed $\mathcal{M}(\delta)$ for free
66:                 $\mu \leftarrow \mu^*$
67:             else
68:                 $\alpha \leftarrow \alpha_0$
69:                 $\mu \leftarrow \mu_0$
70:                 $\mathcal{M}(\delta) \leftarrow$ CalcQuadraticModel$(\Delta_t, \delta_t)$              // Not free here!
71:             end if
72:
73:             $\delta_t = \alpha\Delta_t + \mu\delta_{t-1}$                    // Re-scale the natural gradients to QM
74:         end if                                                          // End of $T_2$
75:
76:         $\theta_{t+1} \leftarrow \theta_t + \delta_t$                               // Update parameters under KFAC
77:
78:     end for                                                             // End of Layers
79:
80:     if $t$ mod $T_1 = 0$ then                                                  // Update $\lambda$
81:         $\mathcal{M}(\delta) \leftarrow$ CalcQuadraticModel$(\Delta_t, \delta_t)$                  // Eqs. B.29, B.35
82:         $\rho \leftarrow$ CalcReductionRatio$(\mathcal{L}(\theta_t), \mathcal{L}(\theta_t + \delta_t), \mathcal{M}(\delta), \mathcal{M}(0))$     // Eq. B.22
83:         $\lambda_{t+1} \leftarrow$ UpdateLambda$(\rho)$                                         // Eq. B.25
84:     end if
85:     $\lambda_{t+1} \leftarrow$ Clamp$(\lambda_{t+1}, \lambda_{min}, \lambda_{max})$
86:     $\gamma_{t+1} \leftarrow$ Clamp$(\gamma_{t+1}, \gamma_{min}, \gamma_{max})$
87: end while                                                               // End of Epochs
```

B.11 Expanding the Quadratic Model to make a Quasi-Newton KFAC Optimiser

The original work of Ref. [20] was targeted towards a SL task with the loss function being the expectation of a log-probability. A connection is drawn between the Hessian of the objective function and an approximation to the Hessian known as the Generalised Gauss-Newton Matrix (GGN) which is a positive-semi definite (PSD) matrix. The approximation must be a PSD matrix in order for curvature to be strictly positive when updating under natural gradient descent [162]. It was shown by Ref. [152] that the FIM is equivalent to the GGN in certain cases, and thereby an equivalent to the Hessian. The motivation of preconditioning via the FIM emerges purely from the Hessian of this *particular* loss function of Ref. [20] which is approximated well by the FIM under certain conditions.

In the case of this thesis, the loss function we have is the expectation value of the energy whose Hessian is not approximated by the QFIM at all. The QFIM is defined as,

$$\mathcal{F}_{ij} = \mathbb{E}_{|\Psi|^2}\left[\left(\frac{\partial \log|\Psi|}{\partial \theta_i} - \mathbb{E}_{|\Psi|^2}\left[\frac{\partial \log|\Psi|}{\partial \theta_i}\right]\right)\left(\frac{\partial \log|\Psi|}{\partial \theta_j} - \mathbb{E}_{|\Psi|^2}\left[\frac{\partial \log|\Psi|}{\partial \theta_j}\right]\right)\right], \quad \text{(B.37)}$$

with the Hessian of the expectation of the energy being defined as,

$$\begin{aligned}\mathcal{H}_{ij} = {} & 2\mathbb{E}_{|\Psi|^2}\left[\left(E_L - \mathbb{E}_{|\Psi|^2}[E_L]\right)\frac{\partial^2 \log|\Psi|}{\partial \theta_i \partial \theta_j}\right] \\ & + 4\mathbb{E}_{|\Psi|^2}\left[\left(E_L - \mathbb{E}_{|\Psi|^2}[E_L]\right)\frac{\partial \log|\Psi|}{\partial \theta_i}\frac{\partial \log|\Psi|}{\partial \theta_j}\right] + 2\mathbb{E}_{|\Psi|^2}\left[\frac{\partial E_L}{\partial \theta_i}\frac{\partial \log|\Psi|}{\partial \theta_j}\right]. \quad \text{(B.38)}\end{aligned}$$

The work of App. B re-scales the preconditioning from the QFIM within a trust-region based around the QFIM itself. This works in Ref. [20] because re-scaling to the FIM approximates to a PSD equivalent of the Hessian. However, by comparing Eq. B.37 to Eq. B.38 we can clearly see these two terms differ greatly. Additionally, Eq. B.38 isn't even PSD and hence requires an approximation like that of the GGN in Ref. [152] to define a quasi-Hessian.

The quasi-Hessian of this section is the Hessian of Eq. B.38 but without the explicit second-derivative term in the first line of Eq. B.38. The quasi-Hessian is denoted $\mathcal{H}_Q$ and by following the Umrigar-Filippi (UF) estimator for the Hessian of Ref. [163] a more robust estimator[9] for the Hessian can be defined as,

$$\frac{\partial^2 \langle E \rangle}{\partial \theta_i \partial \theta_j} = A_{ij} + B_{ij} + D_{ij} \tag{B.39}$$

$$A_{ij} = 2\mathbb{E}_{|\Psi|^2}\left[(E_L(x) - \langle E \rangle) \cdot \left(\frac{\partial^2 \ln|\Psi|}{\partial \theta_i \partial \theta_j} + 2\frac{\partial \ln|\Psi|}{\partial \theta_i} \cdot \frac{\partial \ln|\Psi|}{\partial \theta_j} \right) \right] \tag{B.40}$$

$$B_{ij} = -2\left(\frac{\partial \langle E \rangle}{\partial \theta_i} \cdot \mathbb{E}_{|\Psi|^2}\left[\frac{\partial \ln|\Psi|}{\partial \theta_j} \right] + \frac{\partial \langle E \rangle}{\partial \theta_j} \cdot \mathbb{E}_{|\Psi|^2}\left[\frac{\partial \ln|\Psi|}{\partial \theta_i} \right] \right) \tag{B.41}$$

$$\begin{aligned} D_{ij} = {} & \mathbb{E}_{|\Psi|}\left[\left(\frac{\partial E_L(x)}{\partial \theta_i} - \mathbb{E}_{|\Psi|}\left[\frac{\partial E_L(x)}{\partial \theta_i} \right] \right) \cdot \left(\frac{\partial \ln|\Psi|}{\partial \theta_j} - \mathbb{E}_{\langle|\Psi|\rangle}\left[\frac{\partial \ln|\Psi|}{\partial \theta_j} \right] \right) \right] \\ & + \mathbb{E}_{|\Psi|}\left[\left(\frac{\partial E_L(x)}{\partial \theta_j} - \mathbb{E}_{|\Psi|}\left[\frac{\partial E_L(x)}{\partial \theta_j} \right] \right) \cdot \left(\frac{\partial \ln|\Psi|}{\partial \theta_i} - \mathbb{E}_{\langle|\Psi|\rangle}\left[\frac{\partial \ln|\Psi|}{\partial \theta_i} \right] \right) \right]. \end{aligned} \tag{B.42}$$

We follow the same statements for the naïve estimator of the Hessian for the UF-estimator of the Hessian and exclude the explicit second-derivative of the log-absolute of the wavefunction from the Hessian to define the quasi-Hessian. In addition to being a more robust estimator for the quasi-Hessian, all statistical expectation are performed under the Born distribution of the wavefunction and therefore utilise the sample many-body positions from the expectation value of the energy. This allows for the samples to be efficiently utilised not only for calculating the energy and its energy gradients, but also its rescaling too. This, in principle, allows for more efficient re-scaling under the QM of Eqs. B.29, and B.35 as the re-scaling more appropriately follows the *actual* curvature of the loss landscape. This allows the KFAC algorithm of Alg. 1 to be efficiently re-used albeit with one key change - changing the QM from the QFIM to the quasi-Hessian of Eq. B.42. These results were discussed in Sec. 5.4. The walltime between optimisers is shown in Tbl. B.1.

Optimiser Walltime					
Particles	Adam	KFAC	QN-KFAC	KFAC/Adam	QN-KFAC/Adam
2	0.075(6)	0.20(70)	0.50(16)	2.70	6.70
3	0.112(6)	0.25(80)	0.57(17)	2.22	5.07
4	0.151(7)	0.30(95)	0.64(19)	1.99	4.26
5	0.204(8)	0.36(12)	0.74(21)	1.79	3.64
6	0.26(1)	0.43(13)	0.86(23)	1.68	3.35

Table B.1: The effective walltime per epoch of all optimisers studied here. The overhead of KFAC and QN-KFAC tend to 1.7 and 3.4 respectively. The walltime is shown for a DNN of $H = 64$, $L = 2$, and $D = 1$.

[9] In this context the 'robustness' can be defined via a numerical stability and variance argument. The variance of this estimator holds significantly less variance due to the mean-centring of its statistical expectation but holds the same expectation values. This allows for more accurate statistical expectations with the same number of finite samples thereby leading to a more robust estimator for the Hessian.

At the time of writing this thesis, the current implementation of QN-KFAC is not possible within ‘standard’ installations of PyTorch due PyTorch not being able to efficiently compute higher order gradients. An example of this higher order gradient is the per-sample local energy gradients that are used within the re-scaling to the quasi-Hessian. The most recent version of PyTorch would require the per-sample gradients of the local energy to be computed sequentially which makes QN-KFAC’s overhead infeasibly large and renders QN-KFAC impracticable. This turns our attention to FuncTorch which is an external library for composable function transformation for PyTorch [164]. This library allows for us to sidestep the aforementioned sequential bottleneck of per-sample gradients by exploiting vectorisation to allow all per-sample gradients to be computed in parallel. This reduces the number of times the autograd engine is evoked in compute per-sample gradients from the batch size, B, to just 1. This allows for approximately three orders of magnitude in speed-up and limits the amount of computation to the amount of memory as opposed to walltime.

In addition to the need of using FuncTorch for efficient vectorisation of per-sample gradients, there’s an inherent issue within PyTorch’s implementation of `torch.linalg.slogdet`. The vectorisation of FuncTorch is performed via `vmap` which pushes any sequential operations down to the levels of C++ primitives in order to minimise overhead. FuncTorch can only work with PyTorch’s operators in order to efficiently vectorise, however, the `backward` method of `torch.linalg.slogdet` contains an `if-else` statement which temporarily leaves the PyTorch environment and hence cannot be vectorised by FuncTorch. In order to circumvent this issue, one must install PyTorch from source and modify the `backward` of `torch.linalg.slogdet` within C++ directly [165]. The current vectorisable implementation of `torch.linalg.slogdet` within my own fork of PyTorch is a rudimentary implementation that lacks some numerical stability components inherent to standard PyTorch. Therefore, we default to running all results in this Appendix to double precision (`double`) rather than single precision (`float`). Accordingly, this results in an overhead from working with doubles over floats and the lack of numerical stability of App. A. This shows an avenue of future work to further improve upon this minimalist implementation of QN-KFAC.

Bibliography

[1] G. Carleo and M. Troyer. “Solving the quantum many-body problem with artificial neural networks”. In: *Science* 355.6325 (2017), pp. 602–606.

[2] H. Saito. “Method to solve quantum few-body problems with artificial neural networks”. In: *Journal of the Physical Society of Japan* 87.7 (2018), p. 074002.

[3] D. R. Entem and R. Machleidt. “Accurate nucleon–nucleon potential based upon chiral perturbation theory”. In: *Physics Letters B* 524.1-2 (2002), pp. 93–98.

[4] D. R. Entem and R. Machleidt. “Accurate charge-dependent nucleon-nucleon potential at fourth order of chiral perturbation theory”. In: *Physical Review C* 68.4 (2003), p. 041001.

[5] D. Pfau et al. “Ab initio solution of the many-electron Schrödinger equation with deep neural networks”. In: *Physical Review Research* 2.3 (2020), p. 033429.

[6] J. S. Spencer et al. “Better, faster fermionic neural networks”. In: *arXiv preprint arXiv:2011.07125* (2020).

[7] J. G. Lietz et al. “Computational nuclear physics and post Hartree-Fock methods”. In: *An Advanced Course in Computational Nuclear Physics*. Springer, 2017, pp. 293–399.

[8] E. Gross, E. Runge, and O. Heinonen. *Many-Particle Theory,* Taylor & Francis, 1991. ISBN: 9780750301558. URL: https://books.google.co.uk/books?id=%5C_6y%5C_72b9HbkC.

[9] P. Ring and P. Schuck. *The nuclear many-body problem*. Springer Science & Business Media, 2004.

[10] K. Choo et al. “Symmetries and many-body excitations with neural-network quantum states”. In: *Physical Review Letters* 121.16 (2018), p. 167204.

[11] M. Troyer and U.-J. Wiese. “Computational complexity and fundamental limitations to fermionic quantum Monte Carlo simulations”. In: *Physical Review Letters* 94.17 (2005), p. 170201.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[13] G. Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[14] C. M. Bishop. “Pattern recognition”. In: *Machine learning* 128.9 (2006).

[15] H. Saito. “Solving the Bose–Hubbard model with Machine Learning”. In: *Journal of the Physical Society of Japan* 86.9 (2017), p. 093001.

[16] M. Holzmann et al. “Backflow correlations for the electron gas and metallic hydrogen”. In: *Physical Review E* 68.4 (2003), p. 046707.

[17] T. Tieleman, G. Hinton, et al. “Lecture 6.5-RMSprop: Divide the gradient by a running average of its recent magnitude”. In: *COURSERA: Neural networks for machine learning* 4.2 (2012), pp. 26–31.

[18] W. L. McMillan. “Ground State of Liquid He^4”. In: *Physical Review* 138 (2A Apr. 1965), A442–A451. DOI: 10.1103/PhysRev.138.A442. URL: https://link.aps.org/doi/10.1103/PhysRev.138.A442.

[19] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[20] J. Martens and R. Grosse. “Optimizing neural networks with Kronecker-factored approximate curvature”. In: *International conference on machine learning*. PMLR. 2015, pp. 2408–2417.

[21] J. E.G. Kessler et al. “The deuteron binding energy and the neutron mass”. In: *Physics Letters A* 255.4-6 (May 1999), pp. 221–229. DOI: 10.1016/s0375-9601(99)00078-x. URL: https://doi.org/10.1016/s0375-9601(99)00078-x.

[22] N. Rodning and L. Knutson. “Asymptotic D-state to S-state ratio of the deuteron”. In: *Physical Review C* 41.3 (1990), p. 898.

[23] R. Hofstadter. “Electron scattering and nuclear structure”. In: *Reviews of Modern Physics* 28.3 (1956), p. 214.

[24] G. Van der Steenhoven et al. “Nuclear-density dependence of the electron-proton coupling”. In: *Physical Review Letters* 58.17 (1987), p. 1727.

[25] S. Kopecky et al. “New measurement of the charge radius of the neutron”. In: *Physical Review Letters* 74.13 (1995), p. 2427.

[26] D. Beachey et al. “The relation between the form factor and the radius of the deuteron: relativistic effects”. In: *Journal of Physics G: Nuclear and Particle Physics* 20.12 (1994), p. L143.

[27] J. Kellogg et al. “An electrical quadrupole moment of the deuteron”. In: *Physical Review* 55.3 (1939), p. 318.

[28] T. E. O. Ericson. “The deuteron properties”. In: *Nuclear Physics A* 416 (1984), pp. 281–298.

[29] M. Garcon and J. Van Orden. “The deuteron: structure and form factors”. In: *Advances in nuclear physics*. Springer, 2001, pp. 293–378.

[30] J. Chadwick. “The existence of a neutron”. In: *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* 136.830 (1932), pp. 692–708.

[31] H. Yukawa. “On the interaction of elementary particles. I”. In: *Proceedings of the Physico-Mathematical Society of Japan. 3rd Series* 17 (1935), pp. 48–57.

[32] R. Machleidt and D. Entem. “Chiral effective field theory and nuclear forces”. In: *Physics Reports* 503.1 (2011), pp. 1–75. ISSN: 0370-1573. DOI: https://doi.org/10.1016/j.physrep.2011.02.001. URL: https://www.sciencedirect.com/science/article/pii/S0370157311000457.

[33] M. Lacombe et al. “Parametrization of the Paris $N-N$ potential”. In: *Physical Review C* 21 (3 Mar. 1980), pp. 861–873. DOI: 10.1103/PhysRevC.21.861. URL: https://link.aps.org/doi/10.1103/PhysRevC.21.861.

[34] R. Machleidt. “The Meson Theory of Nuclear Forces and Nuclear Structure”. In: *Advances in Nuclear Physics*. Ed. by J. W. Negele and E. Vogt. Boston, MA: Springer US, 1989, pp. 189–376. ISBN: 978-1-4613-9907-0. DOI: 10.1007/978-1-4613-9907-0_2. URL: https://doi.org/10.1007/978-1-4613-9907-0_2.

[35] R. Machleidt, K. Holinde, and C. Elster. “The Bonn meson-exchange model for the nucleon—nucleon interaction”. In: *Physics Reports* 149.1 (May 1987), pp. 1–89. DOI: 10.1016/s0370-1573(87)80002-9. URL: https://doi.org/10.1016/s0370-1573(87)80002-9.

[36] F. Myhrer and J. Wroldsen. “The nucleon-nucleon force and the quark degrees of freedom”. In: *Reviews of Modern Physics* 60.3 (1988), p. 629.

[37] S. Weinberg. “Phenomenological lagrangians”. In: *Physics A* 96.1-2 (1979), pp. 327–340.

[38] S. Weinberg. *Nuclear forces from chiral Lagrangians*. Nov. 1990. DOI: 10.1016/0370-2693(90)90938-3.

[39] V. Bernard, N. Kaiser, and U.-G. Meissner. “Chiral prediction for the πN S-wave scattering length a^- to order $O(M_\pi^4)$”. In: *Physical Review C* 52 (4 Oct. 1995), pp. 2185–2187. DOI: 10.1103/PhysRevC.52.2185. URL: https://link.aps.org/doi/10.1103/PhysRevC.52.2185.

[40] P. Buettiker and U.-G. Meißner. “Pion–nucleon scattering inside the Mandelstam triangle”. In: *Nuclear Physics A* 668.1-4 (2000), pp. 97–112.

[41] N. Fettes, U.-G. Meißner, and S. Steininger. “Pion-nucleon scattering in chiral perturbation theory (I): Isospin-symmetric case”. In: *Nuclear Physics A* 640.2 (1998), pp. 199–234.

[42] U. Van Kolck. “Few-nucleon forces from chiral Lagrangians”. In: *Physical Review C* 49.6 (1994), p. 2932.

[43] E. Epelbaum et al. “Three-nucleon forces from chiral effective field theory”. In: *Physical Review C* 66.6 (2002), p. 064001.

[44] C. Ordonez, L. Ray, and U. Van Kolck. “Nucleon-nucleon potential from an effective chiral Lagrangian”. In: *Physical Review Letters* 72.13 (1994), p. 1982.

[45] N. Kaiser. “Chiral 2π-exchange NN potentials: Two-loop contributions”. In: *Physical Review C* 64 (5 Oct. 2001), p. 057001. DOI: 10.1103/PhysRevC.64.057001. URL: https://link.aps.org/doi/10.1103/PhysRevC.64.057001.

[46] N. Kaiser. “Chiral 3π-exchange NN potentials: Results for representation-invariant classes of diagrams”. In: *Physical Review C* 61 (1 Dec. 1999), p. 014003. DOI: 10.1103/PhysRevC.61.014003. URL: https://link.aps.org/doi/10.1103/PhysRevC.61.014003.

[47] R. B. Wiringa, V. Stoks, and R. Schiavilla. “Accurate nucleon-nucleon potential with charge-independence breaking”. In: *Physical Review C* 51.1 (1995), p. 38.

[48] R. Vinh-Mau. *The Paris nucleon-nucleon potential*. 1977. URL: https://cds.cern.ch/record/117855.

[49] M. A. Melkanoff et al. “Diffuse-Surface Optical Model Analysis of Elastic Scattering of 17-and 31.5-MeV Protons”. In: *Physical Review* 106.4 (1957), p. 793.

[50] K. S. Krane. *Introductory nuclear physics*. John Wiley & Sons, 1991.

[51] T. Busch et al. “Two cold atoms in a harmonic trap”. In: *Foundations of Physics* 28.4 (1998), pp. 549–559.

[52] A. Rojo-Francàs, A. Polls, and B. Juliá-Díaz. “Static and Dynamic Properties of a Few Spin 1/2 Interacting Fermions Trapped in a Harmonic Potential”. In: *Mathematics* 8.7 (2020), p. 1196.

[53] F. Pederiva, A. Roggero, and K. E. Schmidt. “Variational and diffusion Monte Carlo approaches to the nuclear few-and many-body problem”. In: *An Advanced Course in Computational Nuclear Physics*. Springer, 2017, pp. 401–476.

[54] M. Born. “Quantenmechanik der stoßvorgänge”. In: *Zeitschrift für Physik* 38.11 (1926), pp. 803–827.

[55] F. Becca and S. Sorella. “Variational Monte Carlo”. In: *Quantum Monte Carlo Approaches for Correlated Systems*. Cambridge University Press, 2017, pp. 103–130. DOI: 10.1017/9781316417041.006.

[56] A. Scemama et al. “An efficient sampling algorithm for Variational Monte Carlo”. In: *The Journal of chemical physics* 125.11 (2006), p. 114105.

[57] F. Becca and S. Sorella. “Monte Carlo Sampling and Markov Chains”. In: *Quantum Monte Carlo Approaches for Correlated Systems*. Cambridge University Press, 2017, pp. 56–84. DOI: 10.1017/9781316417041.004.

[58] N. Metropolis and S. Ulam. “The Monte Carlo method”. In: *Journal of the American Statistical Association* 44.247 (1949), pp. 335–341.

[59] W. K. Hastings. “Monte Carlo sampling methods using Markov Chains and their applications”. In: *Biometrika* 57.1 (Apr. 1970), pp. 97–109. DOI: 10.1093/biomet/57.1.97. URL: https://doi.org/10.1093/biomet/57.1.97.

[60] M. Stedman, W. Foulkes, and M. Nekovee. “An accelerated Metropolis method”. In: *The Journal of Chemical Physics* 109.7 (1998), pp. 2630–2634.

[61] G. P. Lepage. *VEGAS - an adaptive multi-dimensional integration program*. Tech. rep. Ithaca, NY: Cornell Univ. Lab. Nucl. Stud., Mar. 1980. URL: https://cds.cern.ch/record/123074.

[62] G. P. Lepage. “Adaptive multidimensional integration: VEGAS enhanced”. In: *Journal of Computational Physics* 439 (Aug. 2021), p. 110386. DOI: 10.1016/j.jcp.2021.110386. URL: https://doi.org/10.1016%2Fj.jcp.2021.110386.

[63] R. Lee et al. “Strategies for improving the efficiency of Quantum Monte Carlo calculations”. In: *Physical Review E* 83.6 (2011), p. 066706.

[64] M. Wilson et al. “Simulations of state-of-the-art fermionic neural network wave functions with diffusion Monte Carlo”. In: *arXiv preprint arXiv:2103.12570* (2021).

[65] S. A. Cook. “The complexity of theorem-proving procedures”. In: *Proceedings of the third annual ACM symposium on Theory of computing - STOC ’71*. ACM Press, 1971, pp. 151–158. DOI: 10.1145/800157.805047. URL: https://doi.org/10.1145/800157.805047.

[66] J. Hermann, Z. Schätzle, and F. Noé. “Deep-neural-network solution of the electronic Schrödinger equation”. In: *Nature Chemistry* 12.10 (2020), pp. 891–897.

[67] T. Kato. “On the eigenfunctions of many-particle systems in quantum mechanics”. In: *Communications on Pure and Applied Mathematics* 10.2 (1957), pp. 151–177.

[68] R. M. Martin, L. Reining, and D. M. Ceperley. “Many-body wavefunctions”. In: *Interacting Electrons*. Cambridge University Press, 2016, pp. 122–143. DOI: 10.1017/cbo9781139050807.007. URL: https://doi.org/10.1017/cbo9781139050807.007.

[69] J. C. Slater. “The theory of complex spectra”. In: *Physical Review* 34.10 (1929), p. 1293.

[70] R. Jastrow. “Many-Body Problem with Strong Forces”. In: *Physical Review* 98 (5 June 1955), pp. 1479–1484. DOI: 10.1103/PhysRev.98.1479. URL: https://link.aps.org/doi/10.1103/PhysRev.98.1479.

[71] D. Ceperley, G. V. Chester, and M. H. Kalos. “Monte Carlo simulation of a many-fermion study”. In: *Physical Review B* 16.7 (1977), p. 3081.

[72] D. M. Ceperley. “Fermion nodes”. In: *Journal of Statistical Physics* 63.5 (1991), pp. 1237–1267.

[73] Y. Kwon, D. M. Ceperley, and R. M. Martin. “Effects of three-body and backflow correlations in the two-dimensional electron gas”. In: *Physical Review B* 48.16 (Oct. 1993), pp. 12037–12046. DOI: 10.1103/physrevb.48.12037. URL: https://doi.org/10.1103/physrevb.48.12037.

[74] R. Feynman and M. Cohen. “Energy spectrum of the excitations in liquid helium”. In: *Physical Review* 102.5 (1956), p. 1189.

[75] Y. Kwon, D. M. Ceperley, and R. M. Martin. “Effects of backflow correlation in the three-dimensional electron gas: Quantum Monte Carlo study”. In: *Physical Review B* 58.11 (Sept. 1998), pp. 6800–6806. DOI: 10.1103/physrevb.58.6800. URL: https://doi.org/10.1103/physrevb.58.6800.

[76] P. Mehta et al. “A high-bias, low-variance introduction to Machine Learning for physicists”. In: *Physics Reports* 810 (2019), pp. 1–124.

[77] G. E. Hinton et al. “Improving neural networks by preventing co-adaptation of feature detectors”. In: *arXiv preprint arXiv:1207.0580* (2012).

[78] Z. Lu et al. “The Expressive Power of Neural Networks: A View from the Width”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/32cbf687880eb1674a07bf717761dd3a-Paper.pdf.

[79] B. Neal et al. “A modern take on the bias-variance tradeoff in neural networks”. In: *arXiv preprint arXiv:1810.08591* (2018).

[80] S. Hochreiter. “The vanishing gradient problem during learning recurrent neural nets and problem solutions”. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6.02 (1998), pp. 107–116.

[81] V. Nair and G. E. Hinton. “Rectified Linear Units Improve Restricted Boltzmann Machines”. In: *ICML*. 2010, pp. 807–814. URL: https://icml.cc/Conferences/2010/papers/432.pdf.

[82] J. Kolbusz, P. Rozycki, and B. M. Wilamowski. “The study of architecture MLP with linear neurons in order to eliminate the “vanishing gradient” problem”. In: *International Conference on Artificial Intelligence and Soft Computing*. Springer. 2017, pp. 97–106.

[83] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[84] J. Zou, Y. Han, and S.-S. So. “Overview of artificial neural networks”. In: *Artificial Neural Networks* (2008), pp. 14–22.

[85] K. Hornik, M. Stinchcombe, and H. White. “Multilayer feedforward networks are universal approximators”. In: *Neural networks* 2.5 (1989), pp. 359–366.

[86] Irie and Miyake. “Capabilities of three-layered perceptrons”. In: *IEEE 1988 International Conference on Neural Networks*. Vol. 1. 1988, pp. 641–648. DOI: 10.1109/ICNN.1988.23901.

[87] A. N. Kolmogorov. “On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition”. In: *Doklady Akademii Nauk*. Vol. 114. 5. Russian Academy of Sciences. 1957, pp. 953–956.

[88] P. Diaconis and M. Shahshahani. “On nonlinear functions of linear combinations”. In: *SIAM Journal on Scientific and Statistical Computing* 5.1 (1984), pp. 175–191.

[89] S. Carroll. “Construction of neural networks using the Radon transform.” In: *IEEE International Conference on Neural Networks*. Vol. 1. IEEE. 1989, pp. 607–611.

[90] T. Chen, H. Chen, and R.-w. Liu. “Approximation capability in $C(\bar{R}^n)$ by multilayer feedforward networks and related problems”. In: *IEEE Transactions on Neural Networks* 6.1 (1995), pp. 25–30.

[91] A. R. Gallant and H. White. “There exists a neural network that does not make avoidable mistakes”. In: *IEEE International Conference on Neural Networks*. IEEE, 1988, pp. 657–664. DOI: 10.1109/icnn.1988.23903. URL: https://doi.org/10.1109/icnn.1988.23903.

[92] G. Cybenko. *Continuous valued neural networks with two hidden layers are sufficient*. Tech. rep. Technical report, Department of computer science, Tufts. University, 1988.

[93] F. Fogelman-Soulié and Y. Le Cun. “Modèles connexionnistes de l’apprentissage”. In: *Intellectica* 2.1 (1987). Included in a thematic issue : Apprentissage et machine, pp. 114–143. DOI: 10.3406/intel.1987.1804. URL: https://www.persee.fr/doc/intel_0769-4113_1987_num_2_1_1804.

[94] A. Lapedes and R. Farber. “How neural nets work”. In: *Evolution, learning and cognition*. World Scientific, 1988, pp. 331–346.

[95] W. Rudin and K. M. R. Collection. *Functional Analysis*. Higher mathematics series. McGraw-Hill, 1973. ISBN: 9780070542259. URL: https://books.google.co.uk/books?id=ehzvAAAAMAAJ.

[96] G. Palm. “On representation and approximation of nonlinear systems”. In: *Biological Cybernetics* 34.1 (1979), pp. 49–52.

[97] R. B. Ash. *Real analysis and probability: probability and mathematical statistics: a series of monographs and textbooks*. Academic press, 2014.

[98] K. Hornik. “Approximation capabilities of multilayer feedforward networks”. In: *Neural networks* 4.2 (1991), pp. 251–257.

[99] M. L. Minsky and S. A. Papert. *Perceptrons: expanded edition*. MIT press, 1988.

[100] J. Shawe-Taylor. “Symmetries and discriminability in feedforward network architectures”. In: *IEEE Transactions on Neural Networks* 4.5 (1993), pp. 816–826.

[101] J. Shawe-Taylor. “Building symmetries into feedforward networks”. In: *1989 First IEE International Conference on Artificial Neural Networks,(Conf. Publ. No. 313)*. IET. 1989, pp. 158–162.

[102] N. Dodd. *Graph Recognition Strategies, RIPR RSRE, Malvern, UK*. Tech. rep. Report No. RIPREP/1000/28/88, 1988.

[103] J. Wood and J. Shawe-Taylor. “Representation theory and invariant neural networks”. In: *Discrete applied mathematics* 69.1-2 (1996), pp. 33–60.

[104] M. Zaheer et al. “Deep sets”. In: *arXiv preprint arXiv:1703.06114* (2017).

[105] A. Sannai, Y. Takai, and M. Cordonnier. “Universal approximations of permutation invariant/equivariant functions by deep neural networks”. In: *arXiv preprint arXiv:1903.01939* (2019).

[106] C. Adams et al. “Variational Monte Carlo calculations of $A \leq 4$ nuclei with an artificial neural-network correlator ansatz”. In: *Physical Review Letters* 127.2 (2021), p. 022502.

[107] K. 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.

[108] I. Shavitt. “The method of configuration interaction”. In: *Methods of electronic structure theory*. Springer, 1977, pp. 189–275.

[109] M. Hutter. “On representing (anti) symmetric functions”. In: *arXiv preprint arXiv:2007.15298* (2020).

[110] P. Blanchard, D. J. Higham, and N. J. Higham. “Accurate computation of the log-sum-exp and softmax functions”. In: *arXiv preprint arXiv:1909.03469* (2019).

[111] W. Fedus, B. Zoph, and N. Shazeer. “Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity”. In: *arXiv preprint arXiv:2101.03961* (2021).

[112] I. Wolfram Research. *Mathematica, Version 13.0.0*. Champaign, IL, 2021. URL: https://www.wolfram.com/mathematica.

[113] B. W. Char et al. *Maple V library reference manual*. Springer Science & Business Media, 2013.

[114] A. Meurer et al. “SymPy: symbolic computing in Python”. In: *PeerJ Computer Science* 3 (Jan. 2017), e103. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.103. URL: https://doi.org/10.7717/peerj-cs.103.

[115] J. F. Nolan. “Analytical differentiation on a digital computer”. PhD thesis. Massachusetts Institute of Technology, 1953.

[116] L. M. Beda et al. *Programs for automatic differentiation for the machine BESM*. Technical Report. (In Russian). Moscow, USSR: Institute for Precise Mechanics and Computation Techniques, Academy of Science, 1959.

[117] L. B. Rall. *Automatic differentiation: Techniques and applications*. Springer, 1981.

[118] R. E. Wengert. “A simple automatic derivative evaluation program”. In: *Communications of the ACM* 7.8 (1964), pp. 463–464.

[119] H. M. Bücker et al. *Automatic differentiation: applications, theory, and implementations*. Vol. 50. Springer Science & Business Media, 2006.

[120] M. Berz. *Computational differentiation: techniques, applications, and tools*. 89. Society for Industrial & Applied, 1996.

[121] G. Corliss et al. *Automatic differentiation of algorithms: from simulation to optimization*. Springer Science & Business Media, 2002.

[122] A. Griewank. “Automatic Differentiaion of Algorithms: Theory, Implementation and Application”. In: *SIAM* 1991 (1991).

[123] A. Griewank and A. Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

[124] M. Iri. “Simultaneous computation of functions, partial derivatives and estimates of rounding errors—Complexity and practicality—”. In: *Japan Journal of Applied Mathematics* 1.2 (1984), pp. 223–252.

[125] A. E. Bryson and Y.-C. Ho. *Applied Optimal Control: Optimization, Estimation, and Control*. Blaisdell Publishing Company or Xerox College Publishing, May 1969. DOI: 10.1201/9781315137667. URL: https://doi.org/10.1201/9781315137667.

[126] P. Werbos. “Beyond Regression : New Tools for Prediction and Analysis in the Behavioral Sciences”. In: *Ph. D. dissertation, Harvard University* (1974). URL: https://cir.nii.ac.jp/crid/1572824499094264320.

[127] D. Parker. *Learning-logic: Casting the Cortex of the Human Brain in Silicon*. Technical report: Center for Computational Research in Economics and Management Science. Center for Computational Research in Economics and Management Science, Alfred P. Sloan School of Management, Massachusetts Institute of Technology, 1985. URL: https://books.google.co.uk/books?id=2kS9GwAACAAJ.

[128] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323.6088 (1986), pp. 533–536.

[129] A. G. Baydin et al. “Automatic differentiation in machine learning: a survey”. In: *Journal of Machine Learning Research* 18 (2018).

[130] A. Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[131] G. Andreas and W. Andrea. “Evaluating derivatives”. In: *Society for Industrial and Applied Mathematics* (2008).

[132] M. Clifford. “Preliminary sketch of biquaternions”. In: *Proceedings of the London Mathematical Society* 1.1 (1871), pp. 381–395.

[133] W. R. Hamilton. *Theory of conjugate functions, or algebraic couples: with a preliminary and elementary essay on algebra as the science of pure time*. PD Hardy, 1835.

[134] B. A. Pearlmutter. “Fast exact multiplication by the Hessian”. In: *Neural computation* 6.1 (1994), pp. 147–160.

[135] Y. A. LeCun et al. “Efficient BackProp”. In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by G. Montavon, G. B. Orr, and K.-R. Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 9–48. ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_3. URL: https://doi.org/10.1007/978-3-642-35289-8_3.

[136] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[137] J. Keeble and A. Rios. “Machine learning the deuteron”. In: *Physics Letters B* 809 (2020), p. 135743.

[138] X. Glorot and Y. Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 249–256.

[139] P. Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[140] M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Applied mathematics series v. 55, no. 1972. U.S. Government Printing Office, 1968. ISBN: 9780160002021. URL: https://books.google.co.uk/books?id=ZboM5tOFWtsC.

[141] J. Carlson et al. “Quantum Monte Carlo methods for nuclear physics”. In: *Reviews of Modern Physics* 87.3 (2015), p. 1067.

[142] J. R. Sarmiento, J. Keeble, and A. Rios. “Machine learning the deuteron: new architectures and uncertainty quantification”. In: *arXiv preprint arXiv:2205.12795* (2022).

[143] S.-X. Zhang, Z.-Q. Wan, and H. Yao. “Automatic Differentiable Monte Carlo: Theory and Application”. In: *arXiv preprint arXiv:1911.09117* (2019).

[144] P.-O. Löwdin. “Quantum theory of many-particle systems. I. Physical interpretations by means of density matrices, natural spin-orbitals, and convergence problems in the method of configurational interaction”. In: *Physical Review* 97.6 (1955), p. 1474.

[145] M. Valiente. “Bose-Fermi dualities for arbitrary one-dimensional quantum systems in the universal low-energy regime”. In: *Physical Review A* 102 (5 Nov. 2020), p. 053304. DOI: 10.1103/PhysRevA.102.053304. URL: https://link.aps.org/doi/10.1103/PhysRevA.102.053304.

[146] M. Girardeau, H. Nguyen, and M. Olshanii. "Effective interactions, Fermi–Bose duality, and ground states of ultracold atomic vapors in tight de Broglie waveguides". In: *Optics Communications* 243.1 (2004). Ultra Cold Atoms and Degenerate Quantum Gases, pp. 3–22. ISSN: 0030-4018. DOI: https://doi.org/10.1016/j.optcom.2004.09.079. URL: https://www.sciencedirect.com/science/article/pii/S0030401804010582.

[147] H. J. Schulz. "Wigner crystal in one dimension". In: *Physical Review Letters* 71 (12 Sept. 1993), pp. 1864–1867. DOI: 10.1103/PhysRevLett.71.1864. URL: https://link.aps.org/doi/10.1103/PhysRevLett.71.1864.

[148] M. Girardeau and M. Olshanii. "Fermi-Bose mapping and N-particle ground state of spin-polarized fermions in tight atom waveguides". In: *arXiv preprint cond-mat/0309396* (2003).

[149] E. Wigner. "On the interaction of electrons in metals". In: *Physical Review* 46.11 (1934), p. 1002.

[150] T. Cheon and T. Shigehara. "Fermion-Boson Duality of One-Dimensional Quantum Particles with Generalized Contact Interactions". In: *Physical Review Letters* 82 (12 Mar. 1999), pp. 2536–2539. DOI: 10.1103/PhysRevLett.82.2536. URL: https://link.aps.org/doi/10.1103/PhysRevLett.82.2536.

[151] T. Kinoshita, T. Wenger, and D. S. Weiss. "Observation of a one-dimensional Tonks-Girardeau gas". In: *Science* 305.5687 (2004), pp. 1125–1128.

[152] J. Martens. "New insights and perspectives on the natural gradient method". In: *arXiv preprint arXiv:1412.1193* (2014).

[153] L. Tonks. "The complete equation of state of one, two and three-dimensional gases of hard elastic spheres". In: *Physical Review* 50.10 (1936), p. 955.

[154] C. Chen et al. "Continuous-time flows for efficient inference and density estimation". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 824–833.

[155] J. Köhler, L. Klein, and F. Noé. "Equivariant flows: exact likelihood generative learning for symmetric densities". In: *International conference on machine learning*. PMLR. 2020, pp. 5361–5370.

[156] S.-I. Amari. "Natural gradient works efficiently in learning". In: *Neural computation* 10.2 (1998), pp. 251–276.

[157] S.-i. Amari and H. Nagaoka. *Methods of Information Geometry*. Vol. 191. American Mathematical Soc., 2000.

[158] J. Martens and I. Sutskever. "Training deep and recurrent networks with Hessian-free optimization". In: *Neural networks: Tricks of the trade*. Springer, 2012, pp. 479–535.

[159] R. Kiros. "Training neural networks with stochastic Hessian-free optimization". In: *arXiv preprint arXiv:1301.3641* (2013).

[160] I. Sutskever et al. “On the importance of initialization and momentum in deep learning”. In: *International conference on machine learning*. PMLR. 2013, pp. 1139–1147.

[161] J. Ba, R. Grosse, and J. Martens. “Distributed Second-Order Optimization Using Kronecker-Factored Approximations”. In: *International Conference on Learning Representations* (2016).

[162] M. Toussaint. *Lecture Notes: Some notes on gradient descent*. Tech. rep. Machine Learning & Robotics lab, FU Berlin, 2012.

[163] J. Toulouse and C. J. Umrigar. “Optimization of quantum Monte Carlo wave functions by energy minimization”. In: *The Journal of Chemical Physics* 126.8 (2007), p. 084102.

[164] R. Z. Horace He. *functorch: JAX-like composable function transforms for PyTorch*. https://github.com/pytorch/functorch. 2021.

[165] R. Zou. personal communication. 2021.