Variational Monte Carlo Algorithms

for Finding Ground States

Evan Wonisch
Supervisor: Filippo Vicentini

Project Report

Centre du Physique Théorique
École Polytechnique
France
May 15, 2026

Contents

1 Introduction

Finding the ground state of a physical system is a frequent task of paramount importance in physics and also physical chemistry. The ground state of a given Hamiltonian corresponds to the state of lowest energy and is thus frequently used to deduce the low-energy properties of systems. In this project, we focus on determining the ground state of molecular systems and later lattices of spins. Starting with most simple and trivial examples, our method is demonstrated to work. Passing on to more complicated problems, our method quickly runs into issues which will be discussed in detail.

The presented method is based on the widely used variational principle in quantum mechanics. It states that you can find the ground state of a system as the state which minimises the expected energy. Thus, in principle, one could try out all possible wavefunctions and check, which one has the lowest energy. It would be the wavefunction of the ground state. Obviously this is not practical as we have infinitely many wavefunctions to try. The main idea to make this feasible, is to parametrise a wavefunction with a finite amount of real scalar parameters by choice of an Ansatz. Then, one just has to find the set of finite parameters which minise the expected energy. If the Ansatz was chosen well, the result is the acutal ground state of the system. In practice however, no one knows how the correct Ansatz should look like and all calculations are mere approximations, which can only be compared to other methods which try to compute the ground state.

A major point of comparison therein, is the ground state energy. Better Ansätze and better calculations obtain lower ground state energy and the lowest one wins. Where analytical solutions become unavailable (which happens very quickly in molecular physics), we thus only compare our obtained energies to density functional calculations and full configuration interaction calculations in selected professional basis sets.

This research project focusses on a novel choice of Ansätze, which are constructed by using neural networks from machine learning. Machine learning has shown promising advancements in wide areas of data driven modelling. Their strength lies in their ability to parametrise classes of functions rather efficiently. This means, that by only choosing a finite amount of parameters $\theta \in \mathrm{R}^{\mathrm{p}}$, neural networks can cover large parts of function spaces like $C^\infty(\mathbb{R}^{\mathrm{d}}, \mathbb{R})$. Obviously, there is no isomorphism between $\mathbb{R}^p$ and $C^\infty(\mathbb{R}^{\mathrm{d}}, \mathbb{R})$, which highlights that neural networks are just universal function approximators. In our application to quantum chemistry, we try to find neural networks which can represent molecular orbitals. Instead of being completely agnostic about past advances in that field, we combine standard methods like linear combination of atomic orbitals (where the coefficients become parameters) and neural networks acting as corrections to those traditional Ansätze, which are shown to work well in certain cases. Thus, our wavefunctions consist out of standard physical Ansätze, multiplied with neural network corrections, which are trained to lower the expected energy.

An important distinction to many machine learning applications is the lack of data in this project. As opposed to learning a function or probability distribution by minimising a statistical distance to a dataset (common in machine learning), we are having no data points to learn the molecular energies from. This may seem unnatural to many machine learning engineers, though it comes natural to physicists, as the variational principle offers the possibility to learn the ground state wavefunction without ever measuring any data in the real world, just by minimising the expected energy (provided the Schrödinger equation is the accurate model of reality). The computational methods to minimse this energy are also of concern in this project. Next to standard numerical methods like gradient descent, we investigate further strategies, by adding memory to the minimisation procedure. These algorithms are also generalised further to work on Riemannian manifolds, which will be the main mathematical object we are dealing with in the following.

Overall, we are able to efficiently compute ground state energies of the dihydrogen molecule and the helium atom with better accuracy than DFT and FCI calculations in the selected basis sets. Crucial aspects here are in curing divergences of energy density, which are brought about by the Coulomb potential. In short: the potential energy at the positions of the nuclei (or when two electrons coincide) diverges because of the Coulomb potential. The kinetic energy has to counteract this, coming into the Schrödinger equation via the Laplacian. This is a difficult dance between diverging energy densities and the constant energy density, the true ground state exhibits. It is one of the main issues we encounter with our method. Furthermore, we turn to spin lattices to investigate our novel optimisation methods, which we formulated for Riemannian manifolds. Our new method though makes the optimisation much more unstable and has not provided any

benefits yet, eventhough similar methods have vastly improved the performance of optimisation algorithms in other cases. This needs further investigation, which could be undertaken as part of a continuation of this research project.

2 Theoretical Background

Given a Hamiltonian H which is a linear operator on a suitable function space over $\mathbb{R}^3$, one tries to solve the following eigenvalue equation, also known as the time-independent Schrödinger equation:

$$H\Psi = E\Psi \tag{1}$$

Here $\Psi \in \mathcal{H}$ is a so-called wavefunction, which must be square integrable. $\mathcal{H}$ is the corresponding Hilbert space of square-integrable functions. For our physical systems, we assume that the spectrum of H is bounded from below and there exists a ground state energy E_0.

2.1 Variational Principle

To find this energy, along with its eigenfunction Ψ_0, a variational principle can be applied: Ψ_0 and E_0 follow from the minimisation of the expected energy functional:

$$E[\Psi] = \frac{\langle\Psi|H|\Psi\rangle}{\langle\Psi|\Psi\rangle} \tag{2}$$

$$= \frac{\int_{\mathbb{R}^3} d^3r\, \Psi^*(r) H\Psi(r)}{\int_{\mathbb{R}^3} d^3r\, |\Psi(r)|^2} \tag{3}$$

Thus we are posed with an optimisation problem:

$$\Psi_0 \in \underset{\Psi\in\mathcal{H}}{\operatorname{argmin}} \frac{\langle\Psi|H|\Psi\rangle}{\langle\Psi|\Psi\rangle} \quad \text{and} \quad E_0 = E[\Psi_0] \tag{4}$$

This optimisation problem is hard to tackle, as the Hilbert space $\mathcal{H}$ is infinite dimensional. However, restricting the optimisation to a simpler subset $M \subset \mathcal{H}$ is helpful, as long as the ground state lies still within it. There are plenty of choices for such a subset M, but particularly useful are the ones which make M isomorphic to some $\mathbb{R}^p$ with a given number p, thus allowing to make M a differential manifold of dimension p. The perks of this are that traditional minimisation algorithms can now be employed to minimise the above energy functional. As we later leave the frame of molecular systems and turn towards spin lattices in the end, the integrals over all spaces would have to be replaced by sums over all spin configurations, i.e. the basis vectors of the Hilbert space.

2.2 Differentiable Manifold

A easy way to find such a subset M is the parametrisation of the variational wave function Ψ with parameters $\theta \in \mathbb{R}^p$. Here, each $\Psi \in M$ is assumed to be normalised. Thus, the manifold is a subset of the Hilbert space $\mathcal{H}$ of wavefunctions, more precisely $M \subset \mathcal{H}_0 := \{\Psi \in \mathcal{H} \mid ||\Psi|| = 1\}$. The manifold is naturally equipped with the chart map θ, which results from the parametrisation:

$$\theta : M \to \mathbb{R}^p \tag{5}$$

$$\Psi \mapsto \theta(\Psi) \tag{6}$$

It is assumed to be bijective, even though this might not be the case automatically when using certain types of parametrisations (e.g. a deep neural network). The inverse θ^{-1} will also be denoted Ψ_θ. Using this chart map, one obtains a coordinate induced set of basis vectors $\frac{\partial}{\partial\theta_i}$ of the T_pM for each point $p \in M$ (Each T_pM is a vector space attached to the point p on the manifold. Therein lie all the tangent vectors to the manifold. In the intrinsic framework of differential geometry, these are directional derivatives which can act on scalar functions on the manifold).

2.3 Riemannian Manifold

The original Hilbert space $\mathcal{H}$ of course was equipped with an inner product $\langle\cdot|\cdot\rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{C}$. This inner product gave rise to the notion of distance in the Hilbert space. We want to inherit this notion of distance onto the manifold. Thereby, we want that the normalisation of the wavefunction doesn't matter. The manifold M will be equipped with a metric g, thus becoming a Riemannian manifold:

$$g : T_pM \times T_pM \to \mathbb{R} \tag{7}$$

The metric is derived from the squared distance function d^2 which lives on $\mathcal{H}_0$ (the subset of normalised wavefunctions) (also called the Bures distance):

$$d^2 : \mathcal{H}_0 \times \mathcal{H}_0 \to \mathbb{R} \tag{8}$$

$$(\Psi, \Phi) \mapsto 2 - 2\,\mathrm{Re}\langle\Psi|\Phi\rangle \tag{9}$$

The components of the metric tensor in the coordinate chart θ at point $\Psi \in M$ will be taken as:

$$g_{ij}(\Psi) = \lim_{\theta \to \theta(\Psi)} \frac{1}{2} \frac{\partial}{\partial\theta_i} \frac{\partial}{\partial\theta_j} d^2(\Psi, \Psi_\theta) \tag{10}$$

Executing this calculation and respecting that the wavefunctions are normalised, thus:

$$\mathrm{Re}\langle\Psi_\theta|\frac{\partial}{\partial\theta_i}\Psi_\theta\rangle = 0 \tag{11}$$

leads to:

$$g_{ij} = -\,\mathrm{Re}\langle\Psi|\frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}\Psi_\theta\rangle\Big|_{\theta(\Psi)} \tag{12}$$

$$= \mathrm{Re}\langle\frac{\partial}{\partial\theta_i}\Psi_\theta|\frac{\partial}{\partial\theta_j}\Psi_\theta\rangle\Big|_{\theta(\Psi)} \tag{13}$$

Along this way, we also retrieve the quantum information tensor S and find the metric as its real part:

$$g_{ij} = \mathrm{Re}\, S_{ij} = \mathrm{Re}\langle\frac{\partial}{\partial\theta_i}\Psi_\theta|\frac{\partial}{\partial\theta_j}\Psi_\theta\rangle\Big|_{\theta(\Psi)} \tag{14}$$

2.4 Natural Gradient Descent

After reducing the size of the space from the entire Hilbert space $\mathcal{H}$ to just a subset, which is made to be a Riemannian manifold, the minimisation of the energy functional on the manifold M can be performed by a standard method: Gradient Descent. As we now live on a manifold equipped with a metric, all these structures will be respected and the resulting method is called Natural Gradient Descent: One starts out with a random position on the manifold and one follows a trajectory which at each point heads towards the minimum of energy. The change of energy along a direction $v \in T_pM$ is given by the gradient. We aim to minimise this change by choosing the best direction v, while keeping the norm of v fixed (as to not do too big steps such that the first order approximation stays valid, one can also formulate this with a Lipschitz constant when knowing that the second derivative is bounded from above). This can be formulated as minimising the following Lagrangian with Lagrange multiplier λ:

$$L(v) = \nabla E(v) + \frac{\lambda}{2}||v||^2 \tag{15}$$

Here, the gradient of the energy function ∇E makes appearance. It is a covector, meaning:

$$\nabla E : T_pM \to \mathbb{R} \tag{16}$$

$$v \mapsto \nabla E(v) \tag{17}$$

It does nothing else then taking a tangent vector (which is a directional derivative) and applying it to the scalar function E. Writing v in the coordinate basis

$$v = v^i \frac{\partial}{\partial \theta_i} \tag{18}$$

the above expression becomes

$$L(v) = \frac{\partial E}{\partial \theta_i} v^i + \frac{\lambda}{2} g_{ij} v^i v^j \tag{19}$$

This is minimised for:

$$\frac{\partial E}{\partial \theta_i} + \lambda g_{ij} v^j = 0 \tag{20}$$

leading to:

$$v^i = -\frac{1}{\lambda} g^{ij} \frac{\partial E}{\partial \theta_j} \tag{21}$$

with g^{ij} being the inverse metric tensor. During our optimisation procedure, we hence wish to follow a curve which is tangent to this vector v all along the way. This curve shall be denoted γ:

$$\gamma : \mathbb{R}^+ \to M \tag{22}$$

$$\tau \mapsto \gamma(\tau) \tag{23}$$

satisfying:

$$\frac{d\gamma^i}{d\tau} = -\frac{1}{\lambda} g^{ij} \frac{\partial E}{\partial \theta_j} \tag{24}$$

Discretising the equation in time with time step ϵ, the trajectory γ i.e. the update rule of the parameters θ is thus:

$$\theta_{n+1} = \theta_n - \frac{1}{\lambda \epsilon} g^{-1} \nabla E \tag{25}$$

Having everything in place, one can now follow numerically the trajectory $\gamma(\tau)$ until convergence to the minimum of the energy (It can be shown, that using Natural Gradient Descent actually converges to the global minimum as it is equivalent to imaginary time evolution of the wavefunction, which has guaranteed convergence, of course only when the manifold covers a suitable area around the ground state and most importantly the ground state itself). The main problem now is, that the parametrisation of the wavefunction has to cover the true ground state, in order to find it. This is yet another hard problem we will discuss below.

2.5 Natural Gradient Descent with Momentum

In analogy to the well known momentum optimisation algorithm for scalar functions $E : \mathbb{R}^{\mathrm{d}} \to \mathbb{R}$, we devise a covariant formulation of this, adapted to the Riemannian manifold at hand. First, we reiterate how the original algorithm works on the $\mathbb{R}^{\mathrm{d}}$ i.e. a flat metric.

2.5.1 Standard Formulation of Momentum

Gradient descent with momentum describes a trajectory $\gamma : \mathbb{R} \to \mathbb{R}^{\mathrm{d}}$ which tries to converge to a global minimum (if that exists) of the scalar function E. It is given by:

$$\ddot{\gamma} = -\kappa \dot{\gamma} - \nabla E|_\gamma \tag{26}$$

Discretizing this equation in time gives rise to a recursion formula with which one can evaluate $\gamma(n\epsilon) = \gamma_n \, \forall n \in \mathbb{N}$. Thus, ϵ functions as the discrete time-integration step. There are multiple

ways to do this, as one has the choice between forward-, backward- or central-type finite differences (amongst others). Here we chose:

$$\ddot{\gamma}(t) = \frac{1}{\epsilon^2}(\gamma(t+\epsilon) - 2\gamma(t) + \gamma(t-\epsilon)) \tag{27}$$

$$\dot{\gamma}(t) = \frac{1}{\epsilon}(\gamma(t+\epsilon) - \gamma(t)) \tag{28}$$

A further choice is given concerning when to evaluate $\nabla E|_\gamma(t)$. Naturally, one might pick the given one, a more stable alternative, also called Nesetrov-Accelerated Gradient descent, uses $\nabla E|_\gamma(t+\epsilon)$, which technically makes the scheme implicit. Upon insertion of the finite differences, one obtains the recursion formula:

$$\gamma_{n+1} = \gamma_n + \frac{1}{1+\kappa\epsilon}(\gamma_n - \gamma_{n-1}) - \frac{\epsilon^2}{1+\kappa\epsilon}\nabla E|_{\gamma_{n+1}} \tag{29}$$

By redefining:

$$\beta = \frac{1}{1+\kappa\epsilon} \tag{30}$$

$$\alpha = \frac{\epsilon^2}{1+\kappa\epsilon} \tag{31}$$

and declaring:

$$\Delta\gamma_n = \gamma_n - \gamma_{n-1} \tag{32}$$

one obtains the following scheme:

$$\Delta\gamma_{n+1} = \beta\Delta\gamma_n - \alpha\nabla E|_{\gamma_{n+1}} \tag{33}$$

This can be easily interpreted: The new momentum $\Delta\gamma_{n+1}$ is the old one, but damped by factor β. Also, the gradient of the energy acts as a force, changing the momentum at each time step. The speciality is that we evaluate the gradient ∇E at γ_{n+1}. This means, we would have to solve for γ_{n+1}, but we want to avoid this. Thus we do a Dyson-type series expansion and truncate it after the first order. To do this, we realize that:

$$\gamma_{n+1} = \gamma_n + \Delta\gamma_{n+1} \tag{34}$$

If we now insert equation (33) into itself and truncate, we finally arrive at:

$$\Delta\gamma_{n+1} = \beta\Delta\gamma_n - \alpha\nabla E|_{\gamma_n+\beta\Delta\gamma_n} \tag{35}$$

which can be used numerically.

2.5.2 Covariant Reformulation

We now generalise the continuous time equation (26) onto the Riemannian manifold (M, g) described before. The only difference is, that we now have a notion of parallel transport and geodesics.

Levi-Civita Connection On a Riemannian manifold (M, g) which we have at hand, one can define a way to not only differentiate scalar fields (this is done by tangent vectors, or by the gradient) or to measure distances (this is why we have g), one can also define the directional derivative of a vector field. This concept is tightly related to parallel transport, as a vector field is called parallel if and only if its directional derivatives vanish (like one would imagine a constant vector field on $\mathbb{R}^d$ where a vector at one position is parallel to all the others). There is an almost canocial choice of connection, the Levi-Cevita connection, which makes the straight line connecting points p_1 and p_2 also be the line of shortest length. It can be directly obtained out of the metric tensor g. This information of this directional derivative comes encompassed in the form of Christoffel-Symbols Γ which at each point of the manifold $p \in M$ have the following signature in our coordinate chart:

$$\Gamma_p : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}^p \tag{36}$$

$$(u, v) \mapsto w \quad \text{linearly} \tag{37}$$

where p is the number of parameters we have (i.e. the dimension of the manifold M). More intuitively, each Γ-symbol is a hyper-matrix. Given a vector field v, which we want to derive in the direction u, $\Gamma(u, v)$ is the correction to the naïve directional derivative $D_u v$ one needs, to account for the curved space. It acts linearly in u and v. The components of this hypermatrix can be obtained out of the metric tensor g and its partial derivatives:

$$\Gamma^i_{jk} = \frac{1}{2} g^{il} \left(\partial_k g_{lj} + \partial_j g_{lk} - \partial_l g_{jk} \right) \tag{38}$$

To define what the second derivative of a curve $\gamma : \mathbb{R} \to M$ is (i.e. the acceleration), the Christoffel symbols appear. Why? Recall that the Christoffel-symbols carry the information of what the straight lines on the manifold are. The second derivative of a curve (the acceleration) is exactly the deviation from that straight line. The definition of the second order time derivative of a trajectory thus looks like:

$$\frac{d^2}{dt^2}\gamma(t) = \ddot{\gamma} + \Gamma(\dot{\gamma}, \dot{\gamma}) \tag{39}$$

The equation from setting the above to zero is the geodesic equation, which describes straight trajectories of constant velocity.

Covariant Formulation of Momentum In our momentum algorithm, we deviate from a geodesic by a damping force $-\kappa\dot{\gamma}$ and an attractive force directing us towards the minimum of the energy $-\nabla E^{\#}$. The hashtag means that we have to raise the index of the gradient via the inverse metric tensor, as seen before. Recall again, that the gradient by its definition is a covector, written in the covector basis with lower case components $\partial_i E$. When contracting with the inverse metric (that's what the hashtag means, also called musical isomorphism) we obtain our natural gradient $g^{ij}\partial_j E$, which is a vector and can thus be equated with the acceleration of the curve γ, which is a vector as well. The continuous time version now reads:

$$\ddot{\gamma} = -\kappa\dot{\gamma} - \nabla E^{\#}|_{\gamma} - \Gamma(\dot{\gamma}, \dot{\gamma}) \tag{40}$$

When now discretising, we use the same type of finite differences as before, except for the Geodesic correction, where we use a backward first difference:

$$\Gamma(\dot{\gamma}(t), \dot{\gamma}(t)) = \frac{1}{\epsilon^2}\Gamma(\gamma(t) - \gamma(t-\epsilon), \gamma(t) - \gamma(t-\epsilon)) \tag{41}$$

Rearranging leads to:

$$\Delta\gamma_{n+1} = \beta \left[\mathbb{1} - \bar{\Gamma}\right] \Delta\gamma_n - \alpha \nabla E|_{\gamma_n + \beta\left[\mathbb{1} - \bar{\Gamma}\right]\Delta\gamma_n} \tag{42}$$

where we used $\bar{\Gamma}$ being the matrix $\Gamma(\Delta\gamma_n, \cdot)$. (It is indeed of question at which $p \in M$ this matrix should be evaluated. Standardly, one might use $\Gamma|_{\gamma}(t)$, but similarly to the Nesterov Gradient, one could try $\Gamma|_{\gamma}(t + \epsilon)$). Great! Also, this equation is nicely interpretable. Therefore, one has to remind oneself on how to parallel transport a vector v along itself infinitesimally. This is given by the linear relationship

$$v' = [\mathbb{1} - \bar{\Gamma}]v \tag{43}$$

where v is the vector to be transported along itself, $\bar{\Gamma} = \Gamma(v, \cdot)$ and v' being the result of that transport. We thus see, that the new momentum $\Delta\gamma_{n+1}$ is given by the damped old momentum, but parallel transported along itself. Furthermore, the gradient is evaluated at a certain midpoint, as in the Nesterov accelerated version. Additionaly, if we set $\beta = 0$, we obtain the overdamped limit, which corresponds exactly to the ordinary natural gradient descent method. Thus, by exploring hyperparameters ranging from $0 < \beta < 1$, one investigates both the standard natural gradient descent, as well as a variable amount of additional momentum.

3 Numerical Methods

In this section, we go into detail about how to compute the expected energy, the gradient of the expected energy, the metric tensor and the christoffel symbols. Also, we will go through different methods of how to parametrise the wavefunction Ψ and thus choose a manifold M on which to perform the optimisation.

3.1 Geometric Computations

The optimisation algorithms above look very promising, but we have to actually calculate these objects in our coordinate chart. Our manifold is high-dimensional, so we have to resort to numerical calculations. Also, all our geometric objects can be formulated as inner products between elements of the underlying Hilbert space. These inner products (i.e. integrals) are very costly or nearly impossible to calculate with discretisation. We have to be able to express all geometric quantities via expectations over the Born-probability of the wavefunction, such that we can make use of Monte-Carlo estimates for those integrals.

Gradient of the Energy As can be seen, the gradient of the energy functional must be computed. Using the formula for the expectation value of the energy, one moves the gradient into the integral and applies the chain rule. After rearranging and formulating the integral as an expectation value, one obtains:

$$\frac{\partial}{\partial\theta_i}E = \frac{\partial}{\partial\theta_i}\frac{\int_{\mathbb{R}^3} d^3r\, \Psi_\theta(r)^* H\Psi_\theta(r)}{\int_{\mathbb{R}^3} d^3r\, |\Psi_\theta(r)|^2} \tag{44}$$

$$= 2\ \mathrm{Re}\int_{\mathbb{R}^3} d^3r\, \frac{|\Psi(r)|^2}{\langle\Psi|\Psi\rangle}\left(\frac{\partial}{\partial\theta_i}\log\Psi(r)\right)[H_{\mathrm{loc}}(r) - E[\Psi]] \tag{45}$$

Thereby, H_{loc} is the local energy, defined as:

$$H_{loc}(r) = \frac{H\,\Psi(r)}{\Psi(r)} \tag{46}$$

If one is handed samples r_i for $i = 1...N$ from the Born-probability distribution $p(r) = |\Psi(r)|^2/\langle\Psi|\Psi\rangle$, these integrals can be approximated effectively. They can be written as expectation values:

$$\frac{\partial}{\partial\theta_i}E = 2\ \mathrm{Re}\ \mathbb{E}\left[\left(\frac{\partial}{\partial\theta_i}\log\Psi_\theta\right)^*(H_{\mathrm{loc}} - \mathrm{Re}\,\mathbb{E}[H_{\mathrm{loc}}])\right] \tag{47}$$

As an approximation, we will substitute them with empirical averages over the samples. Numerically, we can make use of JAX auto-differentiation, allowing to efficiently calculate products of vectors or covectors with the Jacobian of a function. Given N samples, we define a proxy function:

$$\textbf{logpsi} : \mathbb{R}^p \to \mathbb{R}^N \tag{48}$$

$$\theta \mapsto (\log\Psi_\theta(r_i))_{i=1...N} \tag{49}$$

The above gradient can be efficiently calculated as a vector-jacobian product. The function jax.vjp takes in a function f and a vector v and computes $v^T J_f$, where J_f is the Jacobian of f.

$$\nabla E = \mathrm{jax.vjp}(\textbf{logpsi}, H_{loc} - \langle E\rangle) \tag{50}$$

where $H_{loc} - \langle E\rangle$ has components $(H_{loc} - \langle E\rangle)_i = H_{loc}(r_i) - \langle E\rangle$.

Quantum Geometric Tensor In the expression derived so far, the parametrised wavefunctions were restricted to the subset of normalised wavefunctions. If this condition is relaxed, the computations change slightly. As the wave functions we are parametrising are not normalised, one has to substitute $\Psi_\theta \to \frac{\Psi_\theta}{K_\theta}$, where K_θ is the missing normalisation constant. K_θ can be chosen as real. Putting this into the above calculation:

$$g_{ij} = \mathrm{Re}\langle\frac{\partial}{\partial\theta_i}\frac{\Psi_\theta}{K_\theta}|\frac{\partial}{\partial\theta_j}\frac{\Psi_\theta}{K_\theta}\rangle \tag{51}$$

$$= \mathrm{Re}\,\frac{1}{K_\theta^2}\langle\frac{\partial}{\partial\theta_i}\Psi_\theta|\frac{\partial}{\partial\theta_j}\Psi_\theta\rangle - \left(\frac{\partial}{\partial\theta_i}\log K_\theta\right)\left(\frac{\partial}{\partial\theta_j}\log K_\theta\right) \tag{52}$$

The first constituent can be written as:

$$\mathrm{Re}\,\frac{1}{K_\theta^2}\langle\frac{\partial}{\partial\theta_i}\Psi_\theta|\frac{\partial}{\partial\theta_j}\Psi_\theta\rangle = \mathrm{Re}\int_{\mathbb{R}^d} d^dr\frac{|\Psi(r)|^2}{K_\theta^2}\left(\frac{\partial}{\partial\theta_i}\log\Psi_\theta\right)^*\left(\frac{\partial}{\partial\theta_j}\log\Psi_\theta\right) \tag{53}$$

And the other ones as:

$$\left(\frac{\partial}{\partial\theta_i}\log K_\theta\right) = \mathrm{Re}\int_{\mathbb{R}^d} d^d r \frac{|\Psi(r)|^2}{K_\theta^2}\left(\frac{\partial}{\partial\theta_i}\log\Psi_\theta\right) \tag{54}$$

We see again, that those integrals allow for approximation via sampling. At first, we will not compute the entire metric tensor at once, but allow for evaluation of products of g with vectors v. The components of the result are

$$g_{ij}v^j = \mathrm{Re}\,\mathbb{E}\left[\left(\frac{\partial}{\partial\theta_i}\log\Psi_\theta(r)\right)^*\left(\frac{\partial}{\partial\theta_j}\log\Psi(r) - \mathrm{Re}\,\mathbb{E}\left[\frac{\partial}{\partial\theta_j}\log\Psi\right]\right)v^j\right] \tag{55}$$

We can also formulate this in JAX code, using first jax.jvp and afterwards jax.vjp. We define

$$k = \text{jax.jvp}(\mathbf{logpsi}, v) \tag{56}$$

Then we can evaluate the metric tensor as

$$g\cdot v = \text{jax.vjp}(\mathbf{logpsi}^*, k - \mathrm{Re}\,\frac{1}{N}\sum_{i=1} k_i) \tag{57}$$

where $k - \mathrm{Re}\,\frac{1}{N}\sum_{i=1} k_i$ indicates the vector k with the same scalar subtracted from each component.

Christoffel Symbols A new aspect on top of calculating the gradient of the energy and the metric tensor is to calculate the Christoffel-symbols, or more precisely Christoffel-vector products. The following expression can appear:

$$\Gamma(u,v)^i = \Gamma^i_{jk}u^j v^k \tag{58}$$

Inserting equation (38), we obtain:

$$\Gamma^i_{jk}u^j v^k = \frac{1}{2}g^{il}\left(\partial_k g_{lj} + \partial_j g_{lk} - \partial_l g_{jk}\right)u^j v^k \tag{59}$$

The last bit we redefine to be the covector b_{ljk} given as

$$b_{ljk} = \frac{1}{2}(\partial_k g_{lj} + \partial_j g_{lk} - \partial_l g_{jk}) \tag{60}$$

If we assume again that the wavefunctions we are parametrising are normalised, we can use the simple expression given in section (2.3). The covector b_{ljk} boils down to

$$b_{ljk} = \mathrm{Re}\langle\partial_l\Psi|\partial_j\partial_k\Psi\rangle \tag{61}$$

To get into our setting we have to substitute $\Psi_\theta \to \frac{\Psi_\theta}{K_\theta}$ as before and apply the chain rule. One finds that b_{ljk} can be written as:

$$b_{ljk} = \mathrm{Re}\,\frac{1}{<\Psi|\Psi>}\left\langle\partial_l\Psi - \Psi\frac{\mathrm{Re}<\Psi|\partial_l\Psi>}{<\Psi|\Psi>}\middle|\partial_j\partial_k\Psi - \partial_k\Psi\frac{\mathrm{Re}<\Psi|\partial_j\Psi>}{<\Psi|\Psi>} - \partial_j\Psi\frac{\mathrm{Re}<\Psi|\partial_k\Psi>}{<\Psi|\Psi>}\right\rangle \tag{62}$$

As in our algorithm so far only $b_{ljk}v^j v^k$ make appearance, we can simplify slightly further by noticing that the last two terms are equal when contracted.

$$b_{ljk}v^j v^k = \mathrm{Re}\,\frac{1}{<\Psi|\Psi>}\left\langle\partial_l\Psi - \Psi\frac{\mathrm{Re}<\Psi|\partial_l\Psi>}{<\Psi|\Psi>}\middle|\partial_j\partial_k\Psi - 2\partial_j\Psi\frac{\mathrm{Re}<\Psi|\partial_k\Psi>}{<\Psi|\Psi>}\right\rangle v^j v^k \tag{63}$$

We can expand this into an expectation value

$$b_{ljk}v^j v^k = \mathrm{Re}\int_{\mathbb{R}^d} dr^d \frac{|\Psi(r)|^2}{<\Psi|\Psi>}\left(\partial_l\log\Psi - \frac{\mathrm{Re}<\Psi|\partial_l\Psi>}{<\Psi|\Psi>}\right)^*\left(\frac{\partial_j\partial_k\Psi}{\Psi(r)} - 2\partial_j\log\Psi\frac{\mathrm{Re}<\Psi|\partial_k\Psi>}{<\Psi|\Psi>}\right)v^j v^k \tag{64}$$

The term with second-order derivatives can be rewritten as

$$\frac{1}{\Psi}\partial_j\partial_k\Psi = (\partial_j \log\Psi)(\partial_k \log\Psi) + \partial_j\partial_k \log\Psi \tag{65}$$

Inserting this into the above gives

$$b_{ljk}v^j v^k = \mathrm{Re}\,\mathbb{E}\Bigg[\left(\partial_l \log\Psi - \frac{\mathrm{Re} < \Psi|\partial_l\Psi >}{< \Psi|\Psi >}\right)^* \tag{66}$$

$$\left(\partial_j\partial_k \log\Psi + (\partial_j \log\Psi)\left(\partial_k \log\Psi - 2\frac{\mathrm{Re} < \Psi|\partial_k\Psi >}{< \Psi|\Psi >}\right)\right) v^j v^k\Bigg] \tag{67}$$

Defining

$$u(r) = \partial_j \log\Psi(r) v^j \tag{68}$$

and noticing

$$\frac{\mathrm{Re} < \Psi|\partial_j\Psi >}{< \Psi|\Psi >} v^j = \mathrm{Re}\,\mathbb{E}[\partial_j \log\Psi v^j] = \mathrm{Re}\,\mathbb{E}[u] \tag{69}$$

The terms in the second parentheses become:

$$a(r) = \left(\partial_j\partial_k \log\Psi + (\partial_j \log\Psi)\left(\partial_k \log\Psi - 2\frac{\mathrm{Re} < \Psi|\partial_k\Psi >}{< \Psi|\Psi >}\right)\right) v^j v^k \tag{70}$$

$$= \partial_j\partial_k \log\Psi(r) v^j v^k + u(r)\,(u(r) - 2\,\mathrm{Re}\,\mathbb{E}[u]) \tag{71}$$

u can be evaluated easily with a jacobian-vector product. The expectation values are replaced by an average over the samples. The second order derivatives are the costly part and can be evaluated as a hessian sandwich. Having calculated this, the entire expression for the Christoffels becomes:

$$b_{ljk}v^j v^k = \mathrm{Re}\,\mathbb{E}\left[\left(\partial_l \log\Psi - \frac{\mathrm{Re} < \Psi|\partial_l\Psi >}{< \Psi|\Psi >}\right)^* a(r)\right] \tag{72}$$

$$= \mathrm{Re}\,\mathbb{E}\left[\partial_l \log\Psi(r)^* (a(r) - \mathbb{E}\,[a])\right] \tag{73}$$

This can easily be evaluated with a vector-jacobian product.

3.2 Use Neural Networks for Parametrisation

As discussed above, one has to choose a nice subset of the Hilbert space $\mathcal{H}$. This is nicely done by assuming some explicit form of the wavefunction, and then introducing parameters. Of course, the symmetries of a given system can help us guess the form of the wavefunction, and will be incorporated into our algorithm. Still, one does not avoid the question of how to parametrise an arbitrary square-integrable function. Here we use neural networks, as they are universal function approximators, to build such functions. A neural network can be constructed by concatenation of very simple, yet non-linear functions. Such a building block is called layer and shall be denoted $f_{m,n,\theta}$:

$$f_{m,n,\theta} : \mathbb{R}^m \to \mathbb{R}^n \tag{74}$$

$$x \mapsto \sigma(Ax + b) \tag{75}$$

Here, σ may be a nonlinear function you pick, A a linear map from $\mathbb{R}^m$ to $\mathbb{R}^n$ and $b \in \mathbb{R}^n$. We call the parameters of this layer $\theta = \{A, b\}$. One obtains a neural network by concatenation of many of these layers and collecting all parameters in one $\theta \in \mathbb{R}^p$, where p is the number of values we need, to store all the matrices A and vectors b.

$$N_\theta : \mathbb{R}^m \to \mathbb{R}^n \tag{76}$$

$$x \mapsto f_1 \circ \cdots \circ f_N \tag{77}$$

And the choice of f_1 to f_N has of course to be such, that the concatenation of all of them works and results in the signature indicated for N_θ. We call N_θ a neural network, it looks schematically like in figure (1).

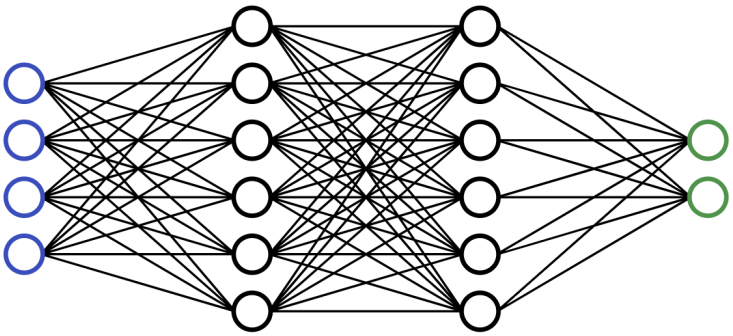

Figure 1: A simple feed-forward neural network

How to build a wavefunction out of this? To build a wavefunction, we simply build a neural network with the corresponding signature:

$$N_\theta : \mathbb{R}^3 \to \mathbb{R}^2 \tag{78}$$

Here, the choice of how many building blocks and of what size is suppressed, this is to be figured out later. We convert the $\mathbb{R}^2$ output of the neural network to a complex number by simply introducing the imaginary unit to one of its components:

$$\Psi(r) = N_\theta^{(1)}(r) + iN_\theta^{(2)}(r) \tag{79}$$

If one uses a C^∞ non-linear function σ, also Ψ will be C^∞ with respect to r, and also θ. Yet, when using many layers, Ψ can be very general and approximate a wide variety of functions. Alternatives are numerous, the above might not be normalisable and thus not a good wavefunction, one can mitigate this: Making sure, that the non-linearity σ is always positive, one can have a better candidate by exponentiating it with a minus sign:

$$\Psi(r) = \exp\Big[-N_\theta^{(1)}(r) + iN_\theta^{(2)}(r)\Big] \tag{80}$$

Here, we allow for an arbitrary complex phase, and if the neural network goes to infinity quick enough when $r \to \infty$ (linearly is already enough), then the wavefunction is nicely normalisable.

3.3 Problems with Non-diferentiable Wavefunctions

When violating against the prescription of smooth (at least C^2) neural networks, a fatal breakdown already occurs. This smoothness is provided by choosing an appropriate nonlinearity σ, also called activation function. It is common practise in many machine learning to use activation functions like Relu (which are differentiable almost everywhere, expect at the origin), as they provide a stable non-linearity. The gradient can be computed at almost every point, thus most numerical schemes still work. In our case, the method breaks down, as the Monte Carlo estimate of the local energy H_{loc} becomes biased. Suppose, for a single electron, the Hamiltonian looks as follows:

$$H = -\frac{\hbar^2 \Delta}{2m} + V(r) \tag{81}$$

The local energy H_{loc} becomes:

$$H_{loc}(r) = -\frac{\hbar^2}{2m}\frac{\Delta\Psi(x)}{\Psi(x)} + V(r) \tag{82}$$

As only the non-singular parts of H_{loc} (these are the parts where the Laplacian can actually act on the wavefunction) can be sampled by the monte carlo method, the singular parts create a bias. This allows kinetic energy to 'hide' in the non-smooth parts of the wavefunction, making the estimated energy decrease below the ground state energy (This is a breakdown as this can't happen for any wavefunction as long as the calculation of the expected energy works, they point here is that is doesn't work when the wavefunction is not C^2). An example is illustrated in figure (2):

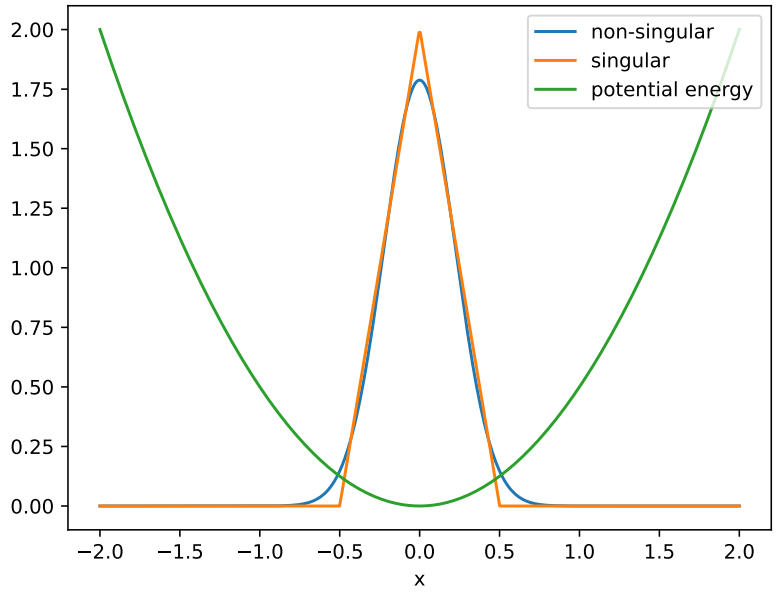

Figure 2: Two different wavefunctions which asympotically minimse the expected potential energy $V(x) = x^2$ by narrowing down to the delta distribution $\delta(x)$. The piece-wise linear (and thus singular in second derivative) wavefunction fails to provide kinetic energy, as taking the Laplacian of this piecewise linear function is zero almost everywheren. The gaussian wavefunction does provide the correct kinetic energy, resulting in the correct expected energy.

4 Results: Continuous Systems with Natural Gradient Descent

4.1 Harmonics Oscillator in 3D

We are finally ready to find our first ground state. In this case we make it very easy and just consider the harmonic oscillator in 3D:

$$H = -\frac{\hbar^2}{2m}\Delta + \frac{1}{2}m\omega^2(x^2 + y^2 + z^2) \tag{83}$$

The energies are known:

$$E = \hbar\omega\left(n_x + n_y + n_z + \frac{3}{2}\right) \tag{84}$$

Setting $\hbar = m = \omega = 1$, the ground state energy is:

$$E = \frac{3}{2} = 1.5 \tag{85}$$

We use a very good Ansatz which basically takes away all the work:

$$\Psi = \exp\left\{-\frac{1}{2}r^2\theta\right\} \tag{86}$$

Here we just have one single parameter θ. Taking 500 samples of the Born-probability, a step-size of 0.01 and 1000 iterations, we easily converge to the ground state with the natural gradient descent method. This could also have been calculated by hand. The expected energy over the course of iterations is depicted in figure (3). A good criterion to diagnose if one has found the ground state if the actual ground state energy is unknown is, that also the variance of the expected energy vanished. This is simply due to the fact, that if Ψ is eigenstate to H with eigenvalue E, it

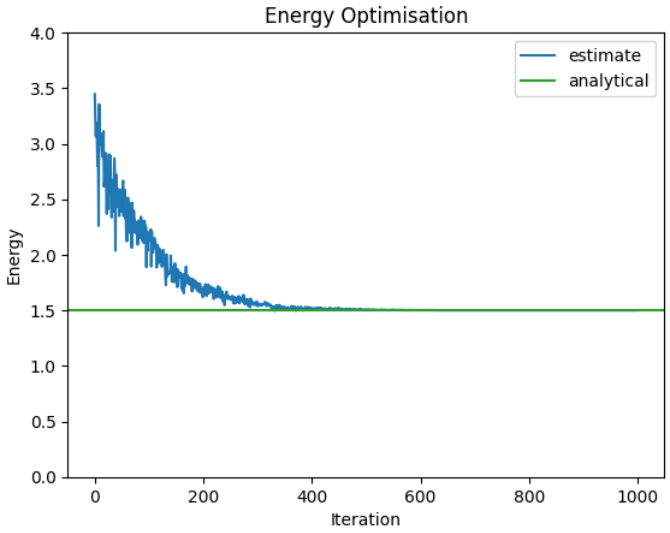

Figure 3: The energy decreases and reaches the ground state energy without problems

also is an eigenstate of H^2 with eigenvalue E^2, making the variance of the energy disappear. The variance of the energy can be calculated by monte carlo sampling as:

$$(\Delta E)^2 = \frac{1}{N}\sum_{i=1}^{N} H_{\text{loc}}(r_i)^2 - \left(\frac{1}{N}\sum_{i=1}^{N} H_{\text{loc}}(r_i)\right)^2 \tag{87}$$

In this example, we find it going to zero with almost machine precision. It worked!

4.2 Hydrogen Atom and Naïve Ansatz

We now are ready to apply our Ansatz to a simple hydrogen atom. The Hamiltonian looks like the following:

$$H = -\frac{1}{2}\Delta - \frac{1}{r} \tag{88}$$

As a wavefunction we use a simple neural network consisting out of 4 layers. No symmetries of the system were incorporated, the wavefunction is build as:

$$\Psi(r) = \exp[-N_\theta(r)] \tag{89}$$

The non-linearity was chosen to be $\sigma = \log\cosh$, which is always positive. During the optimisation with natural gradient descent (4), the expected energy decreases and comes close to the ground state energy of $E_0 = -0.5\,\text{Hartree}$. Though, the variance doesn't go to zero. We don't converge. A resulting slice of the wavefunction along the z axis is shown in figure (5). Looking at the local energy H_{loc}, we see this problem manifesting (6). The local energy in this case is:

$$H_{\text{loc}}(r) = -\frac{1}{2}\frac{\Delta\exp[-N_\theta(r)]}{\exp[-N_\theta(r)]} - \frac{1}{r} \tag{90}$$

As we chose a wavefunction which is smooth, the left term can never compensate for the diverging coulomb potential. But this is indeed necessary. If one computes the local energy H_{loc} for the actual ground state, one finds that it is just a constant with the value of the ground state energy. Thus, an important understanding is, that we have to allow the Ansatz to make H_{loc} a constant.

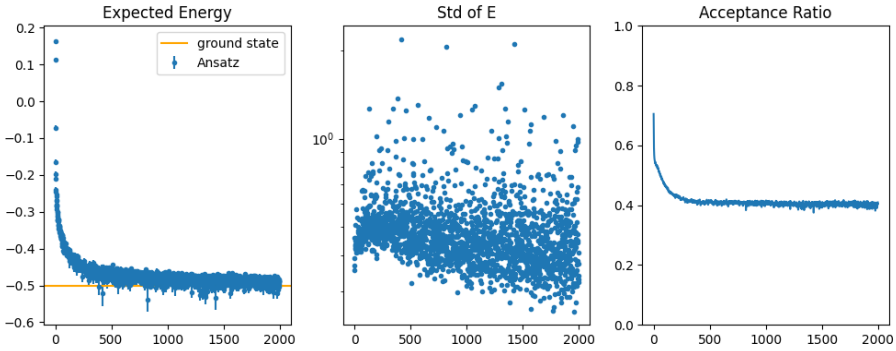

Figure 4: On the left, the expected energy during the optimisation, in the center, the variance of the expected energy, and on the right, the acceptance ratio of our monte-carlo sampling. The expected energy after this optimisation is $E = -0.488 \pm 0.0035$ Hartree.

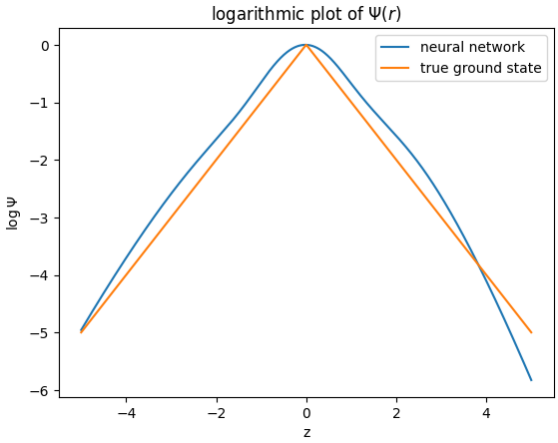

Figure 5: A slice through the z-axis reveals, that the neural network is approaching the true ground state. Though, as it is a smooth function, it cannot replicate the cusp the ground state displays at $r = 0$.

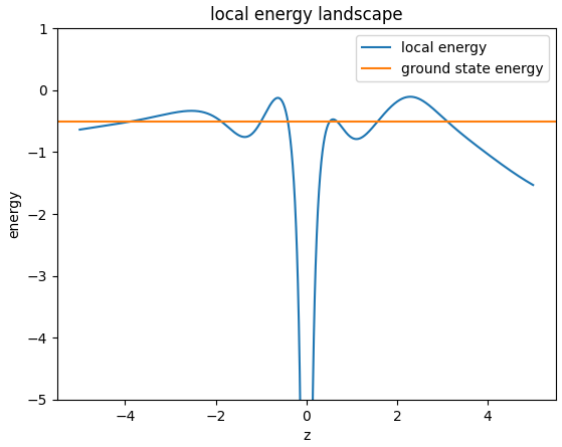

Figure 6: We see, that the local energy actually diverges around the nucleus. This is due to the coulomb potential. The true ground state compensates for the diverging potential energy by a diverging local kinetic energy, though by our choice of smooth neural network, we can never have a diverging Laplacian.

4.3 Hydrogen Atom and Refined Ansatz

As seen above, one of the main problems was, that the cusp, and the resulting divergent local kinetic energy could not be produced by our smooth ansatz. This is because we ruled out any non-differentiability of our Ansatz as this can make the monte carlso sampling break down and produce expected energies below the ground state. This is a dilemma. However, if the potential is non-diverging, the wavefunction is smooth in that area. For a single atom, it is thus sufficient to correct the cusp in the vicinity of the nucleus and let the neural network do the rest. We thus refine the Ansatz by placing the cusp where it belongs and adding the neural network:

$$\Psi(r) = \exp[-N_\theta(r) - r] \tag{91}$$

Of course this reduces the neural network part to triviality, as the correct ground state is given if N_θ just becomes zero everywhere. Still, we can check, that the problems from before disappear by calculating the local energy:

$$H_{\text{loc}}(r) = -\frac{1}{2}\frac{\Delta\Psi(r)}{\Psi(r)} - \frac{1}{r} \tag{92}$$

Taking Ψ as a product (as above) of $\Phi = \exp[-r]$ and $\chi = \exp[-N_\theta(r)]$ we can write H_{loc} as:

$$H_{\text{loc}}(r) = -\frac{1}{2}\frac{\Delta\Phi(r)}{\Phi(r)} - \frac{1}{2}\frac{\Delta\chi(r)}{\chi(r)} - \frac{(\nabla\Phi)\cdot(\nabla\chi)}{\Phi(r)\chi(r)} - \frac{1}{r} \tag{93}$$

This is great. Now taking something smart for Φ, which ideally cancels the $\frac{1}{r}$ potential energy around the origin should help us remove the singularity. In this case, it amounts to:

$$H_{\text{loc}}(r) = -\frac{1}{2}\frac{\Delta\chi(r)}{\chi(r)} - \frac{(\nabla\Psi)\cdot(\nabla\chi)}{\Phi(r)\chi(r)} - \frac{1}{2} \tag{94}$$

Also from here we see, what is the best state for the neural network. If it identically goes to zero everywhere, the local energy becomes -0.5, which is constant and also the ground state energy. This would mean convergence. Let's verify that it also works in practise (7) Of course, we have out-sourced the problematic bits into the underlying Ansatz $\Phi(r) = \exp[-r]$, which already is the true ground state. Also the local energy becomes constant (8).

4.4 The Dihydrogen Molecule

As it turns out, to cure the divergences in local energy for multi-nuclei atoms is not easy. Our attempts are showcased in (A.2). Here we present a simpler Ansatz which doesn't cure these

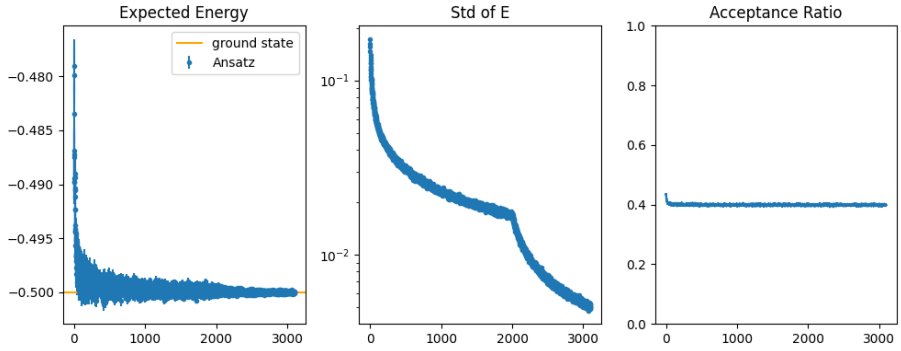

Figure 7: One can see, that the ground state energy is reached very quickly, and that afterwards the variance of the expected energy shrinks. There is no end in sight, and the optimisation procedure can be interrupted when a desirable accuracy is reached. The expected energy at the end of this optimisation is $E = -0.500027 \pm 5.0841 \cdot 10^{-5}$ Hartree. So much better than before.

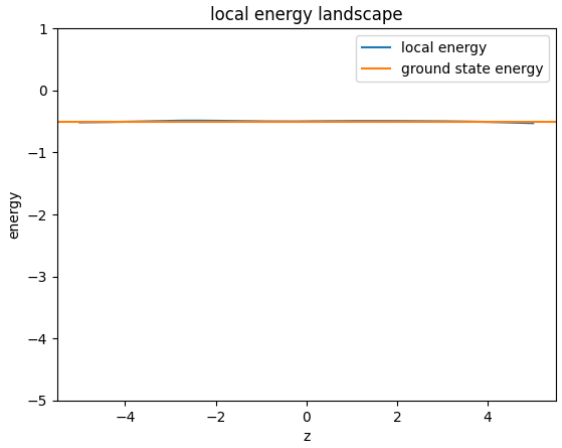

Figure 8: Also, the local energy is converging to a constant equalling to the ground state energy.

divergences but still allows us to obtain ground state energies below the FCI calculations in the augCCPVTZ basis set. These energies can be easily calculated using the python-package *pyscf* [1]. The new Ansatz is based on an approximate solution to the H_2^+ molecule, which goes beyond a simple LCAO Ansatz. It is most easily expressed in elliptical coordinates. Defining the two nuclei at positions $\vec{R}_i$ for $i = 1, 2$ the distances of the electron to the nuclei naturally are given by:

$$r_i = |\vec{r} - \vec{R}_i| \tag{95}$$

We then introduce the elliptical coordinates

$$\lambda = r_1 + r_2 \tag{96}$$

$$\mu = r_1 - r_2 \tag{97}$$

The approximate wavefunction of the H_2^+ molecule is then written as:

$$\Psi_0(\vec{r}) = \exp[-\alpha\lambda]\cosh(\beta\mu) \tag{98}$$

The numbers α and β will be fit-parameters. To obtain an Ansatz for the singlett state of the dihydrogen molecule, we symmetrically combine the orbital Ψ_0 (i.e. multiplying) and adding two further things: A so-called Jastrow correlation factor which is supposed to account for the electron-electron repulsion in the following form:

$$J_\theta(\vec{r_1}, \vec{r_2}) = \exp[-N_\theta(\vec{r_1}, \vec{r_2})] \tag{99}$$

The function N_θ is here a neural network with parameters θ. In similar fashion as before, one can see that the local energy will diverge for $\vec{r_1} \to \vec{r_2}$. In contrast to before, this can be fixed more easily by adding an electron-electron cusp like in [2]:

$$\chi(\vec{r_1}, \vec{r_2}) = \exp\left[-\frac{1}{2}\frac{1}{1 + |\vec{r_1} - \vec{r_2}|}\right] \tag{100}$$

The full Ansatz is then given by:

$$\Psi(\vec{r_1}, \vec{r_2}) = \Psi_0(\vec{r_1})\Psi_0(\vec{r_2})\chi(\vec{r_1}, \vec{r_2})J_\theta(\vec{r_1}, \vec{r_2}) \tag{101}$$

Minimising the energy via natural gradient descent leads to optimisation depicted in figure (9). This result is much better than the one trying to fix the local energy divergences in (A.3). Apparently, even though these divergences matter, when using sufficiently flexible neural networks and good Ansätze, the overall approximation still can be accurate.

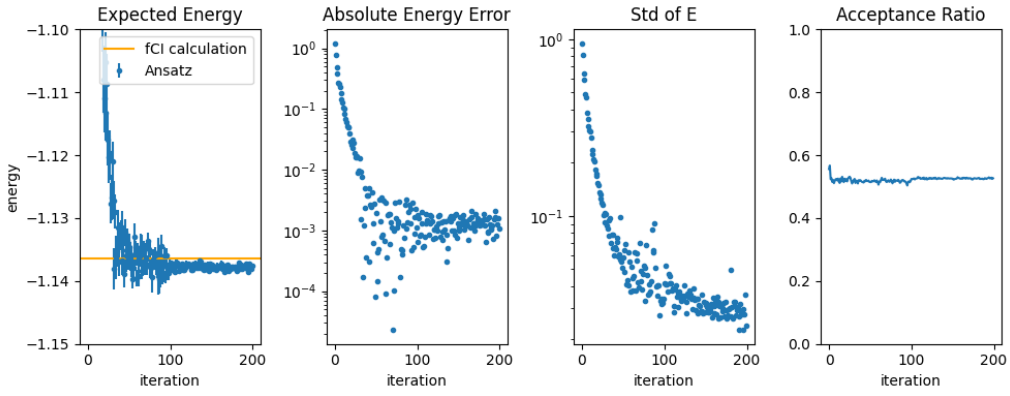

Figure 9: One can observe that the estimated energy for the H_2 molecule drops below the estimate obtained with FCI via *pyscf*. The energy error is now meaningless, as the pyscf threshold obviously is no longer a benchmark for this Ansatz.

4.5 Helium-Triplet

In a similar way, we can obtain a good Ansatz for the Helium atom in its triplet state, meaning, that the space-like wavefunction is antisymmetric with respect to particle exchange. Again, we

use a Jastrow correlation factor J_θ and electron-electron-cusps as defined before. Instead of the symmetric dihdrogen molecular orbital, we use a antisymmetric combination of atomic Ψ_{1s} and Ψ_{2s} orbitals:

$$\Psi = J_\theta(\vec{r_1}, \vec{r_2})[\Psi_{1s}(\vec{r_1})\Psi_{2s}(\vec{r_2}) - \Psi_{2s}(\vec{r_1})\Psi_{1s}(\vec{r_2})]\chi(\vec{r_1}, \vec{r_2}) \tag{102}$$

Also this Ansatz proves to be very good when comparing to pyscf calculations, as we can again obtain lower energies, as in figure (10).

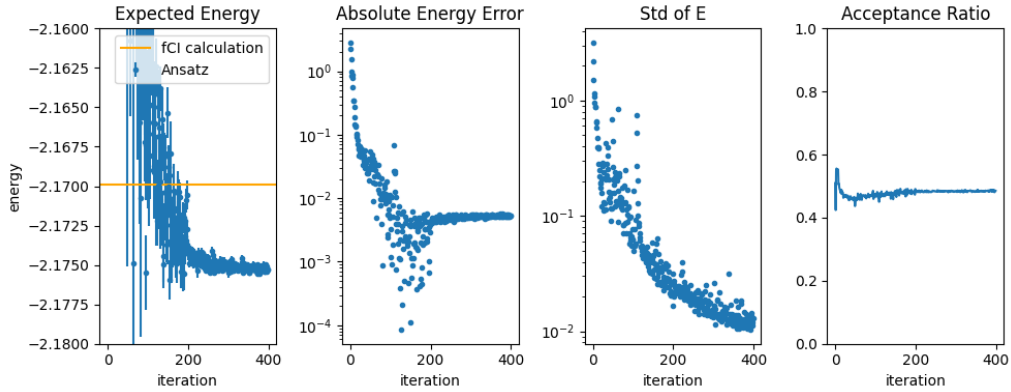

Figure 10: With the described Ansatz, we can beat the FCI calculations for the Helium atom in its triplet state.

5 Results: Heisenberg Model and Natural Momentum

One can also apply the variational methods to spins interacting on lattices. Those kinds of models are of great interest as they can be interpreted as effective descriptions of interacting many-particle problems found in solid state physics. A typical example is the Heisenberg spin chain with N spins. We take $N = 16$. The Hilbert space is $\mathcal{H} = (\mathbb{C}^2)^{\otimes N}$ and we chose the eigenbasis of the total spin-z operator $|\sigma>$ where σ is a bit string of up/down referring to each site. The Heisenberg Hamiltonian is then given by:

$$H = -J \sum_{<i,j>} \vec{S_i} \cdot \vec{S_j} \tag{103}$$

When setting $J < 0$, we obtain the antiferromagnetic Heisenberg model, which will be the object of our study. In this section we are interested in how the two different optimisation algorithms (natural gradient with or without momentum) compare. To do this, we launched a hyperparameter search. Thy hyperparameters in play were the learning rate α, the momentum parameter β and also a condition value r which we use to pseudo-invert the metric tensor. As an Ansatz, we use a simple densely connected feed-forward neural network, meaning that the input, which is a bit-string of length 16 is transformed to a 48 dimensional vector by a parametrised affine linear transformation (also called dense layer). Afterwards, it is transformed non-linearly by applying $x \mapsto \log\cosh x$ on each component individually. Then, it is transformed linearly into a vector space of dimension 2. Finally, the two components are combined into a complex number, as discussed before. All the coefficients of the linear or affine-linear transformations are parameters. It is observed, that all the algorithms have convergence issues in this problem. Thus, for each hyperparameter, we launch the optimisation 10 times with different random seeds to obtain a rough estimate of a survival rate (meaning that the optimisation is stable and does not explode for a significant time of 1500 iterations). Then, we compute the relative errors of all runs and plot the minimal errors for each hyperparameter configuration. Thus we get an impression of which hyperparameters work and how good the result is shown in figure (11). As it is apparent in this figure, using higher β deteriorates the stability of the optimisation, rendering our method not beneficial. The region of stability moves to the left (the area of small learning rate α). This can be understood as following. Using higher β while keeping the learning rate α fixed results in larger steps taken during optimisation. One should counteract this by using smaller learning rate α when incrementing β. This should

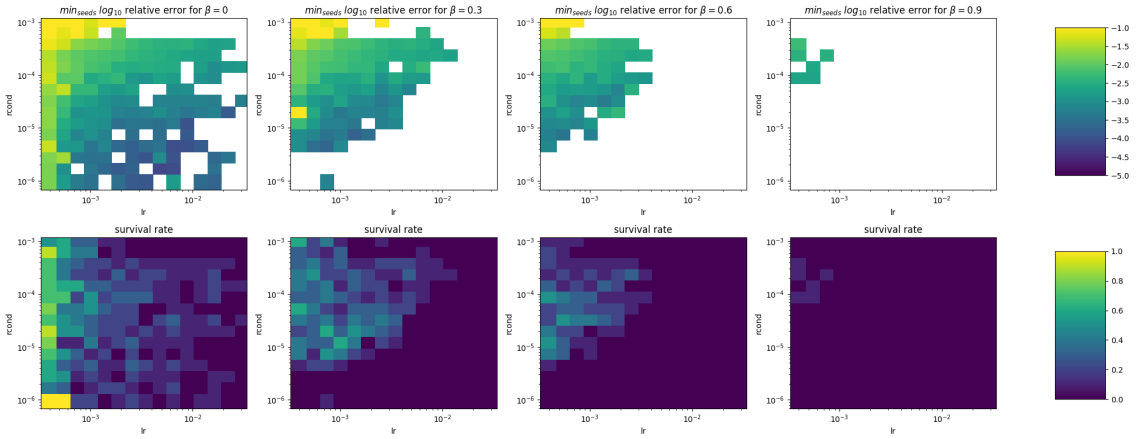

Figure 11: When increasing the momentum parameter β, the survival rate diminishes on the scanned patch of hyperparameters. It appears that the entire feasible area moves to the left. By increasing β, this also has the side effect of taking bigger steps during the optimisation, i.e. a bigger learning rate. One should thus scan the areas more to the left of the plots.

undo the shift of the stability region to the left. This should be done in the future. A second observation is, that the conditioning of the inverse metric tensor also influences the stability. This is commonly seen, also in other algorithms. As we have to compute the inverse of a matrix, which do to overparametrisation or numerical error can be non-invertible, we use a pseudo-inverse. Using higher conditioning, the determinant of the pseudo-inverse is reduced, making the computations more stable. Also, applying this momentum-based optimisation method to molecular orbitals would be very interesting, as those problems suffer less from ill-conditioned metric tensors, due to their continuous nature and continuous smapling procedures.

6 Outlook

In this research project, we embarked on a journey to tackle one of the most fundamental challenges in quantum mechanics and quantum chemistry: determining the ground state of physical systems. The ground state, representing the state of lowest energy, is not only central to understanding the low-energy properties of molecular and lattice systems but also crucial for predicting various physical phenomena. Our approach combined the time-honored variational principle with modern advancements in machine learning, particularly the use of neural networks, to develop a novel method for approximating ground states.

The foundation of our method rests on the variational principle, a well-established concept in quantum mechanics. The principle asserts that the ground state of a system can be found by identifying the wavefunction that minimizes the expected energy. However, the vastness of the function space, where an infinite number of possible wavefunctions exist, makes a brute-force search for the ground state impractical. To overcome this, we adopted a strategy based on parametrizing wavefunctions using a finite set of real scalar parameters. This approach reduces the problem to finding the optimal parameters that yield the lowest energy, thus approximating the ground state.

A key innovation in our work was the introduction of neural networks to construct these parametrized wavefunctions, or Ansätze. Neural networks, known for their universal function approximation capabilities, allowed us to efficiently explore large sections of the function space with a finite number of parameters. This approach provided a flexible and powerful tool for representing complex wavefunctions, which are often challenging to describe using traditional methods.

In developing our Ansätze, we did not disregard the advances made in quantum chemistry. Instead, we integrated traditional methods, such as the linear combination of atomic orbitals, with neural network corrections. This hybrid approach leveraged the strengths of both worlds: the physical intuition and established success of traditional quantum chemistry methods and the adaptability and computational efficiency of neural networks. The resulting wavefunctions were therefore not purely machine-learned constructs but rather informed by decades of physical theory

and empirical success.

One of the distinguishing features of our approach was the lack of reliance on data, which sets it apart from most machine learning applications. Typically, machine learning models are trained on extensive datasets, optimizing their parameters by minimizing a statistical distance to known data points. In contrast, our method required no such data. Instead, it relied entirely on the variational principle to guide the learning process. The neural networks in our models learned to minimize the expected energy directly, allowing us to approximate the ground state wavefunction without needing any empirical data. This data-free learning process is not only elegant but also mirrors the theoretical nature of quantum mechanics, where the Schrödinger equation serves as the ultimate model of reality.

Our research produced promising results, particularly in the context of molecular systems. We successfully computed the ground state energies of simple yet fundamental systems, such as the dihydrogen molecule and the helium atom. Remarkably, our method achieved better accuracy than traditional approaches, such as Density Functional Theory (DFT) and Full Configuration Interaction (FCI), when using selected professional basis sets. This achievement underscores the potential of our neural network-based Ansatz to surpass existing computational methods, at least for certain classes of problems.

However, the journey was not without challenges. One of the significant hurdles we encountered was related to the divergences in energy density, particularly those caused by the Coulomb potential. The potential energy at nuclear positions or when two electrons coincide tends to diverge, creating a complex interaction with the kinetic energy, which is represented in the Schrödinger equation via the Laplacian. Balancing these divergent energies with the constant energy density of the true ground state proved to be a delicate and challenging task. Our approach had to carefully navigate this intricate landscape, where small miscalculations could lead to significant deviations from the true ground state energy.

In addition to molecular systems, we extended our investigation to spin lattices, which represent a different class of quantum systems with their unique complexities. Here, we explored advanced optimization techniques that incorporated memory and operated on Riemannian manifolds. The idea was to generalize our method and potentially improve the stability and efficiency of the energy minimization process. Unfortunately, this extension introduced unexpected instability into the optimization process. While similar methods have shown substantial improvements in other contexts, in our case, they did not yet provide the desired benefits. This highlighted the challenges of applying these sophisticated mathematical tools in the context of quantum mechanical optimization and indicated the need for further refinement.

In summary, our research demonstrated the potential of neural network-based Ansätze in accurately determining the ground state of molecular systems, offering improvements over some traditional methods. However, it also revealed significant challenges, particularly in handling energy density divergences and extending the method to more complex systems like spin lattices. The mixed results suggest that while our approach holds promise, further investigation and development are necessary to fully harness its potential. This work lays the groundwork for future studies that could refine the neural network Ansatz, improve optimization techniques, and extend the applicability of our method to a broader range of quantum systems.

References

[1] pyscf package for quantum chemistry computations, *https://pyscf.org/*

[2] Deep-neural-network solution of the electronic Schrödinger equation, Hermann, Jan and Schätzle, Zeno and Noé, Frank, *https://arxiv.org/abs/1909.08423*

A Appendix

A.1 Metric Tensor in Detail

Non-normalised Wavefunctions As the wave functions we are parametrising are not normalised, one has to substitute $\Psi_\theta \to \frac{\Psi_\theta}{K_\theta}$, where K_θ is the missing normalisation constant. K_θ can be chosen as real. Putting this into the above calculation:

$$g_{ij} = \mathrm{Re}\langle \frac{\partial}{\partial\theta_i}\frac{\Psi_\theta}{K_\theta} | \frac{\partial}{\partial\theta_j}\frac{\Psi_\theta}{K_\theta}\rangle \tag{104}$$

$$= \mathrm{Re}\,\frac{1}{K_\theta^2}\langle \frac{\partial}{\partial\theta_i}\Psi_\theta | \frac{\partial}{\partial\theta_j}\Psi_\theta\rangle - 2\left(\frac{\partial}{\partial\theta_i}\log K_\theta\right)\left(\frac{\partial}{\partial\theta_j}\log K_\theta\right) \tag{105}$$

The first constituent can be written as:

$$\mathrm{Re}\,\frac{1}{K_\theta^2}\langle \frac{\partial}{\partial\theta_i}\Psi_\theta | \frac{\partial}{\partial\theta_j}\Psi_\theta\rangle = \mathrm{Re}\int_{\mathbb{R}^d} d^d r \frac{|\Psi(r)|^2}{K_\theta^2}\left(\frac{\partial}{\partial\theta_i}\log\Psi_\theta\right)^*\left(\frac{\partial}{\partial\theta_j}\log\Psi_\theta\right) \tag{106}$$

And the other ones as:

$$\left(\frac{\partial}{\partial\theta_i}\log K_\theta\right) = \mathrm{Re}\int_{\mathbb{R}^d} d^d r \frac{|\Psi(r)|^2}{K_\theta^2}\left(\frac{\partial}{\partial\theta_i}\log\Psi_\theta\right) \tag{107}$$

A.2 Dihydrogen Molecule and LCAO Ansatz

We could now try to generalise this to a system of two nuclei and try to find its ground state. The hamiltonian now looks as the following:

$$H = -\frac{1}{2}\Delta - \frac{1}{|r-R_1|} - \frac{1}{|r-R_2|} \tag{108}$$

We chose the nuclei to be aligned along the x-axis by setting:

$$R = \pm\frac{d}{2}e_x \tag{109}$$

One finds that this Hamiltonian commutes with complex conjugation, rotation and inversion along the x-axis. We deduce that the wavefunction of the ground state should only depend on the distances to the nuclei r_1, r_2:

$$\Psi = \Psi(r_1, r_2) \tag{110}$$

with:

$$r_i = |r - R_i| \tag{111}$$

From what we learned so far, it would make much sense to build a wavefunction as a product of something which takes care of the diverging Coulomb potentials and a variational rest. A common first idea would be to use a standard LCAO wavefunction, here we will show that this is not desirable but that it has to be modified slightly. We calculate the local energy of the following wavefunction:

$$\Psi(r) = e^{-r} + e^{-|r-R|} \tag{112}$$

Here we set the origin of the coordinate system to one of the nuclei. The local energy is now:

$$H_{\mathrm{loc}}(r) = \frac{1}{\Psi(r)}\left[-\frac{1}{2}\Delta\Psi(r) - \frac{1}{r}\Psi(r) - \frac{1}{|r-R|}\Psi(r)\right] \tag{113}$$

$$= \frac{1}{\Psi(r)}\left[-\frac{1}{2}\Delta e^{-r} - \frac{1}{2}\Delta e^{-|r-R|} - \frac{1}{r}e^{-r} - \frac{1}{r}e^{-|r-R|} - \frac{1}{|r-R|}\Psi(r)\right] \tag{114}$$

$$\tag{115}$$

As e^{-r} is the ground state of the one-nuclei hydrogen problem, two terms of the above simplify a lot:

$$-\frac{1}{2}\Delta e^{-r} - \frac{1}{r}e^{-r} = -\frac{1}{2}e^{-r} \tag{116}$$

It is to note that here, a part of the Coulomb potential got absorbed, which is what we want. The local energy becomes:

$$H_{\text{loc}}(r) = \frac{1}{\Psi(r)}\left[-\frac{1}{2}e^{-r} - \frac{1}{2}\Delta e^{-|r-R|} - \frac{1}{r}e^{-|r-R|} - \frac{1}{|r-R|}\Psi(r)\right] \tag{117}$$

Performing the same procedure with the other nucleus, one obtains:

$$H_{\text{loc}}(r) = \frac{1}{\Psi(r)}\left[-\frac{1}{2}e^{-r} - \frac{1}{2}e^{-|r-R|} - \frac{1}{r}e^{-|r-R|} - \frac{1}{|r-R|}e^{-r}\right] \tag{118}$$

We now see, that this local energy still diverges when approaching $r \to 0$ or $r \to R$, because of the mixed-terms, where the potential of the one nucleus meets a non-zero wavefunction of the other nucleus. So just this plain LCAO is not in our favour, one can in principle fix this (see appendix A.3). We went down an alternative route though.

A.3 Dihydrogen with cutoff-LCAO Ansatz

If the wavefunction of the respective other nucleus vanished in the vicinity of the respective nucleus, the above would be fine. Thus, we propose an alternative to the LCAO by introducing a smooth version of indicator function:

$$\mathbb{O}_A(x) = \begin{cases} 0 \quad \text{for} \quad x \in A \\ 1 \quad \text{if } x \text{ is more far than } \epsilon \text{ from } A \\ \text{otherwise smooth} \end{cases} \tag{119}$$

Here we also define:

$$\mathbb{1}_A(x) = 1 - \mathbb{O}_A(x) \tag{120}$$

The proposed alternative to the LCAO now reads:

$$\Psi(r_1, r_2) = e^{-r_1} \cdot \mathbb{O}_{\{r_2<k\}} + e^{-r_2} \cdot \mathbb{O}_{\{r_1<k\}} \tag{121}$$

By doing this, the diverging terms in the local energy are removed, as when coming closer than k to one of the nuclei, the Ansatz just looks like the hydrogen ground state of that nucleus without further addition. Let's look at the two Ansätze and their local energy in (12) and (13).

Now we could multiplicatively add (i.e. multiply) a neural network. We first take our cutoff-LCAO Ansatz, which we call Φ and an exponentiated neural network χ:

$$\Phi(r_1, r_2) = e^{-r_1} \cdot \mathbb{O}_{\{r_2<k\}} + e^{-r_2} \cdot \mathbb{O}_{\{r_1<k\}} \tag{122}$$

$$\chi(r_1, r_2) = e^{-N_\theta(r_1, r_2)} \tag{123}$$

Our Ansatz is now:

$$\Psi(r_1, r_2) = \Phi(r_1, r_2)\chi(r_1, r_2) \tag{124}$$

We already know what the local energy looks like for a product, let's formulate it in this case:

$$H_{\text{loc}}(r) = -\frac{1}{2}\frac{\Delta\Phi(r)}{\Phi(r)} - \frac{1}{2}\frac{\Delta\chi(r)}{\chi(r)} - \frac{(\nabla\Phi)\cdot(\nabla\chi)}{\Phi(r)\chi(r)} - \frac{1}{|r-R_1|} - \frac{1}{|r-R_2|} \tag{125}$$

We shall now move closer to R_1 and more fare to R_2 than k, thus $\Phi(r_1, r_2) = e^{-r_1}$. The first and the fourth term of the above then equate to $-\frac{1}{2}e^{-r_1}$, resulting in:

$$H_{\text{loc}}(r) = -\frac{1}{2}e^{-r_1} - \frac{1}{2}\frac{\Delta\chi(r)}{\chi(r)} - \frac{(\nabla e^{-r_1})\cdot(\nabla\chi)}{e^{-r_1}\chi(r)} - \frac{1}{|r-R_2|} \tag{126}$$

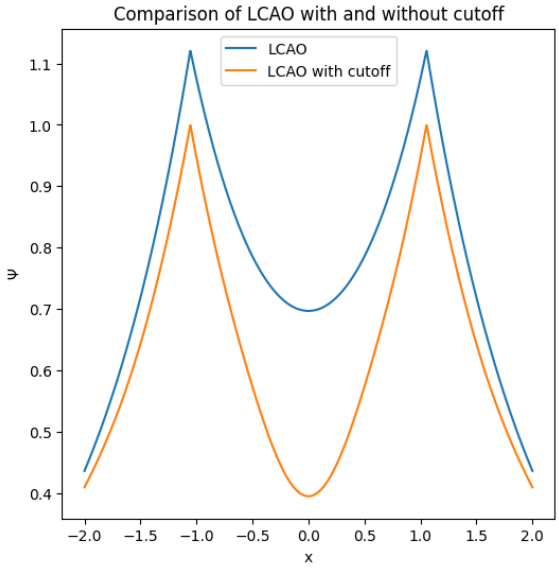

Figure 12: Here the two Ansätze are compared.

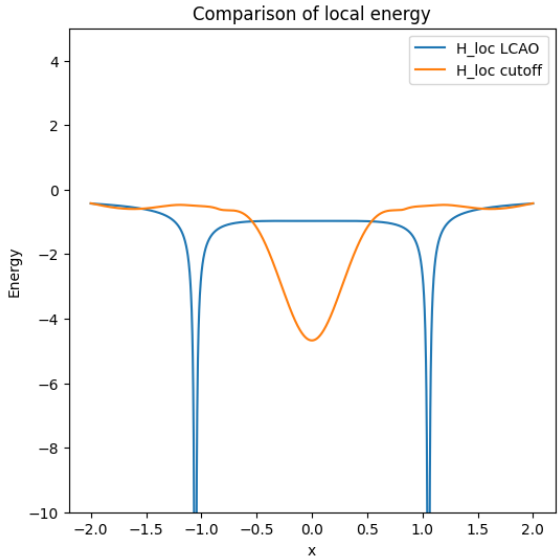

Figure 13: The local energy divergences are fixed when using the cutoff LCAO Ansatz.

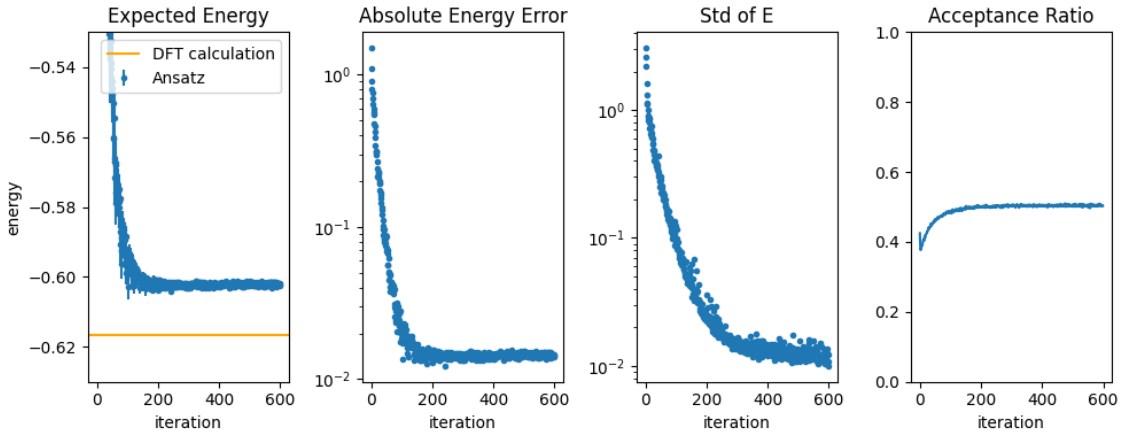

Figure 14: The optimisation converges to an energy value above the ground state energy which was obtained by DFT calculations.

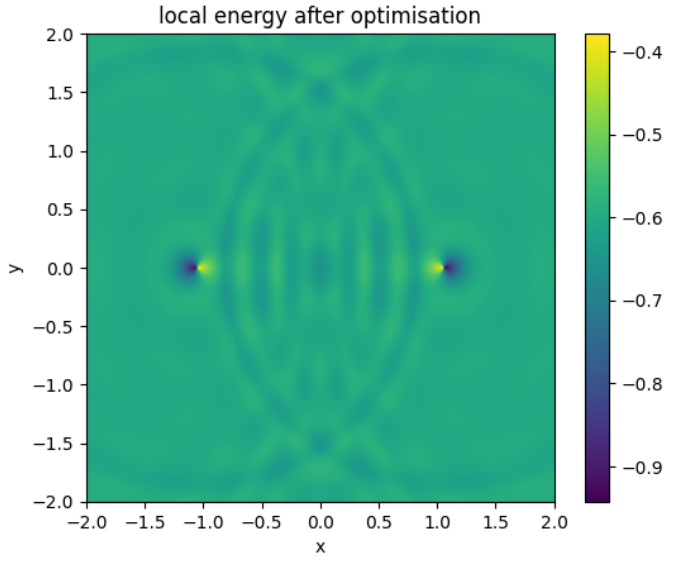

Figure 15: The local energy on a slice of $z = 0$ after optimisation. The local energy approaches a constant outside the vicinity of the nuclei. Though, we see two discontinuities at the nuclei positions.

Great! We don't see any diverging terms. The optimisation procedure is depicted in figure (14). The obtained ground state energy is compared to DFT calculations and we still deviate $1.5 \cdot 10^{-2}$ Hartree from it. Judging from the variance, we have not converged to the ground state. We shall investigate the local energy to find out why, see figure (15). The discontinuities we observe can be calculated analytically, as they result from the gradient term in the local energy:

$$H_{\text{loc}}^{\text{problem}}(r_1, r_2) = -\frac{(\nabla e^{-r_1}) \cdot (\nabla \chi)}{e^{-r_1}\chi(r_1, r_2)} \tag{127}$$

Calculating the gradient of Φ (when again being close to R_1) and assuming ξ is just a linear function:

$$\nabla e^{-r} = -e^{-r} e_r \tag{128}$$

$$\chi = \alpha x \tag{129}$$

The problematic term becomes:

$$H_{\text{loc}}^{\text{problem}}(r) = \frac{\alpha e^{-r} e_r \cdot e_x}{\Psi(r)} = \frac{\alpha e^{-r} \sin\theta \cos\phi}{\Psi(r)} \tag{130}$$

These are exactly the terms we observe in the above. The neural network cannot compensate for these terms in the local energy, as they are not continuous, and the neural network is restricted to differentiable functions for reasons we explained in the beginning. The most current progress, is to construct neural networks whose gradients are vanishing when reaching one of the nuclei positions, as to make this discontinuity disappear. Though, this seems to be a difficult task. One can multiply the network with some squared distance from the nuclei, resulting in a locally flat neural network. This has the drawback, that the entire wavefunction would be zero. A more recent idea is to multiply the network with our smooth indicator functions, to set them zero around the nuclei and then to add a constant to them in this area such that their gradient is zero but the wavefunction is non-zero. Though also, these approaches haven't come to fruition yet.

A.4 $\Psi = 0$ Problem

Choosing a simple toy model, one can demonstrate one of the drawbacks of monte carlo sampling: biased estimates. In combination with a wavefunction which, and thus also the Born probability, has a node $\Psi = 0$, this can result in the gradient optimisation to fail. To demonstrate this in a two-dimensional Hilbert space, we place a $\frac{1}{2}$ spin into a magnetic field and try to optimise for the ground state. Choosing the basis $\{|\uparrow\rangle, |\downarrow\rangle\}$ and a parameter $\theta \in [0; \pi]$ we get a variational ansatz by:

$$\Psi : \mathbb{R} \to \mathcal{H} \tag{131}$$

$$\theta \mapsto \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} \tag{132}$$

As hamiltonian, we take the x component of the spin operator:

$$H = -S_x = -\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{133}$$

As initial parameter we choose $\theta_0 = \pi/8$.

Calculating the expected energy analytically, one obtains the expressions:

$$E = -\sin 2\theta \tag{134}$$

and respectively

$$\nabla E = -2\cos 2\theta \tag{135}$$

for the gradient. Indeed, imposing the continuous time gradient descent scheme,

$$\partial_t \theta = -\nabla E|_\theta \tag{136}$$

One converges smoothly to the ground state, which is given by $\theta = \pi/4$. Following this smooth trajectory, θ crosses through 0. This point shall be of interest, as the second component of the wavefunction becomes zero. Along with it comes the problem, that the respective Born probability becomes arbitrarily small and that the Monte Carlo sampling will not accurately sample those small probabilities. Instead, it will effectively sample from a distribution where the tails are truncated. Thus introducing a bias. We shall now investigate how the expressions for the sampled gradient look like and what this implies for the optimisation procedure.