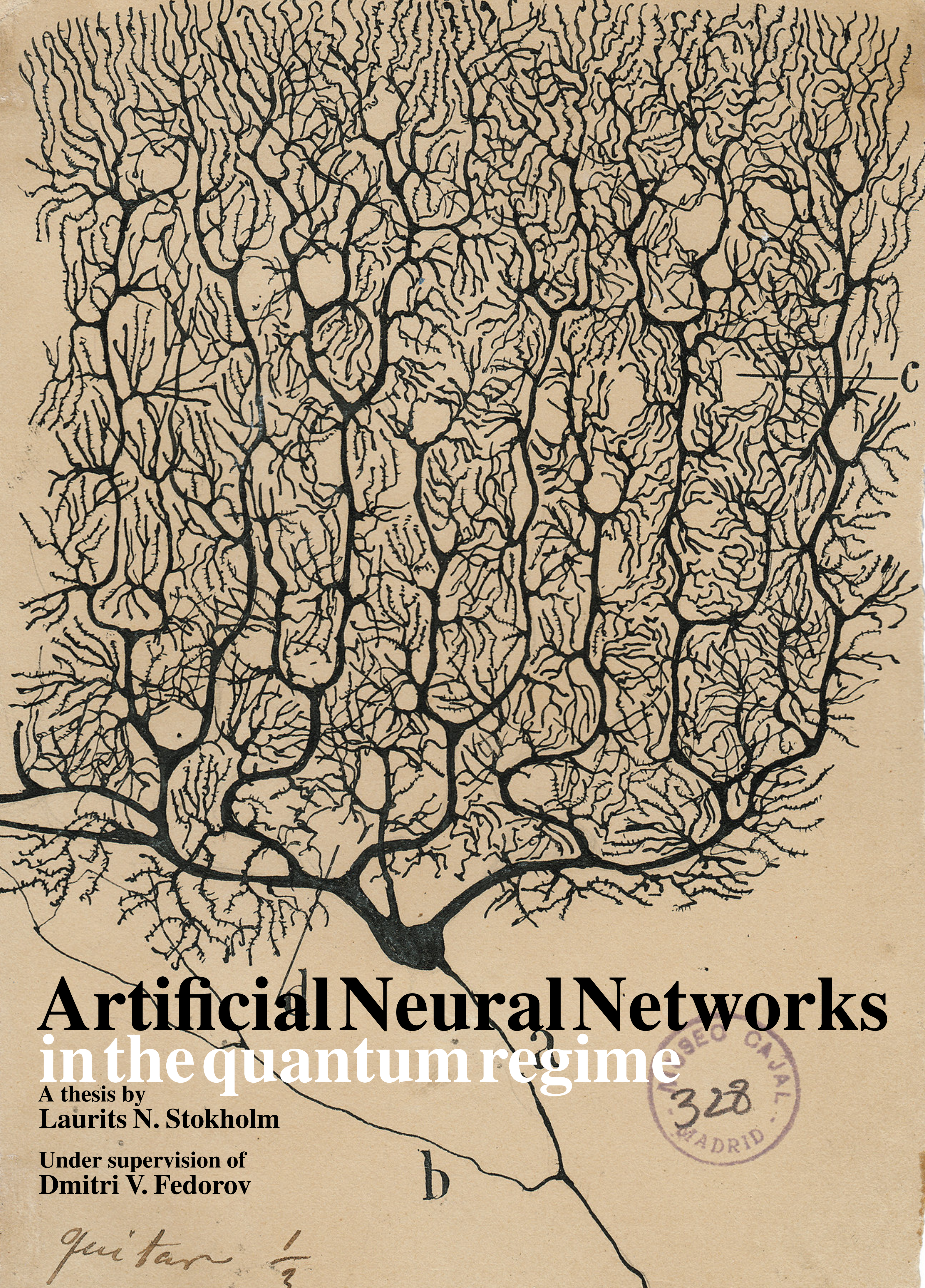


Artificial Neural Networks in the quantum regime

A thesis by
Laurits N. Stokholm

Under supervision of
Dmitri V. Fedorov

University of Aarhus

A BACHELOR'S THESIS IN THE SUBJECT OF PHYSICS

Submitted to the Faculty of Science and Technology
at the Department of Physics and Astronomy

Presenting

ARTIFICIAL NEURAL NETWORKS
as trial wave function
to numerically solve quantum systems ab initio

Thesis by
Laurits N. Stokholm

Under supervision of
Dmitri V. Fedorov

December 26, 2019

Laurits Nikolaj Stokholm, 2019.
Bachelor's project written under supervision of Dmitri Vladimir Fedorov at the
Subatomic Physics group, Department of Physics and Astronomy, Aarhus University.

This document is typeset with LaTeX using the memoir class. The text is set with *Latin Modern* in 11 point size. The choice of layout with the limit of 66 characters per line is quite deliberate since the author believes that the chosen layout improves readability, and with approximately 36 lines per page it adheres to the requirement that a page of text contains approximately 2400 characters.

Printed at Aarhus University.

The cover illustration shows a network of Purkinje neurons from pigeon cerebellum as first drawn by Nobel laureate Santiago Ramón y Cajal, who is known as the father of modern neuroscience. Credit: Santiago Ramón y Cajal, 1899. Instituto Santiago Ramón y Cajal, Madrid, Spain. (Online version in colour.)

Preface

This report was written to conclude my bachelor's degree at the Department of Physics and Astronomy at Aarhus University.

In the process, I have been confronted with complex concepts that I have never come across before, which has made me take the time to further immerse myself into the theory of Numerical Physics. I completed the curriculum of the *10 ECTS* course *Practical Programming and Numerical Physics*, alongside my attended courses. This taught me the language of C — a powerful general-purpose programming language — in which the entire project has been implemented. This experience has led me deeper into the art of computer programming. I have become comfortable in building *Makefiles* and the command-line driven graphing utility of *gnuplot*, in which all of the programs of this project have been generated and the graphics have been created. I have enjoyed spending time diving into the literature of *Artificial Neural Networks*. In doing so, I have encountered a variety of methods where especially the *Diffusion Monte Carlo* and the *Markov Chain Monte Carlo* have been interesting to read about. This might inspire future research. I am not sure I would have studied these concepts had it not been for the initiative of my supervior. To recognize my grattitude, I would like to thank Dmitri V. Fedorov for his help and patience. I would also like to give a sincere thank to Emma Pallesen Ib, Mathias Rav and my sister, Amalie Louise Stokholm for giving helpful suggestions and valuable advise on this thesis.

Abstract

Inspired by the universal approximation theorem and the widespread adoption of artificial neural network techniques in a diversity of scientific fields, this bachelor's project investigates feed-forward neural networks as general-purpose trial wave functions for solving eigenvalue problems of differential and integro-differential operators. Using system-specific activation functions of either Gaussian, Gaussian wavelet, or exponential characteristics, this method relies upon the optimisation of the parameters of the trial wave functions according to the Rayleigh-Ritz variational principle and an other method as proposed by [9]. Starting off with the Quantum Harmonic Oscillator (QHO), the ground state and the succeeding two excited states are solved with exact energies. Subsequently, the perturbed potential of the QHO — that of the Morse potential — is investigated. The Coulombic problem of the hydrogen atom is also solved using this scheme. Lastly, by adding one more electron to the hydrogen atom, this method solves the three-body problem of the negative hydrogen ion. The method in all the treated cases proves to be in fine agreement with the theoretical results and it can be concluded that this novel ab initio method is highly efficient. It is hence a promising tool for tackling problems of higher complexity and dimensionality.

Resumé

Inspireret af den universelle tilnærmelsessætning samt den voksende anvendelse af kunstige neurale netværk indenfor diverse videnskabelige felter vil dette bachelorprojekt undersøge et såkaldt fremad-propagerende (eng: feed forward) neuralt netværks egenskab som en almen anvendt bølgefunktion til løsning af egenværdiproblemer af differential og integrodifferentiale operatorer. Ved anvendelse af systemspecifikke aktiveringsfunktioner af Gaussisk, Gaussiske små-bølger samt eksponentiel karakteristik vil metoden optimere parametrene efter Rayleigh-Ritz' variationsprincip samt en anden metode, som fremvist i [9]. De undersøgte systemer er den kvantemekaniske harmoniske oscillator (QHO), hvor grundtilstanden og de to derpå følgende exciterede tilstande løses med de eksakte energier. Derefter undersøges Morse potentialet, efterfulgt af et Coulombproblem for hydrogenatomet. Til sidst tilføjes en elektron i problemet, hvorved trelegemeproblemet løses. Den anvendte metode løser de undersøgte problemer, og det kan konkluderes, at denne tilgang er effektiv. Neurale netværk ser derfor ud til at være et lovende redskab til løsningen af komplekse og høj-dimensionelle problemer.

Contents

Preface **i**

Abstract **ii**

Resumé **ii**

Contents **iii**

1 Introduction **1**

2 Implementations **5**

2.1 Methods for Numerical Approximations 5

2.2 Solving the Excited States 8

3 Results **9**

3.1 Quantum Harmonic Oscillator 9

3.2 Hydrogen Atom 14

3.3 The Hydrogen Anion 17

4 Discussion **19**

5 Conclusion **21**

Bibliography **23**

6 Appendix **25**

6.1 Morse Potential 25

6.2 Orthogonality of Stationary States of the Hermitian Discrete Operator 26

6.3 Fall-off Rates 27

6.4 Code Snippets 29

1 Introduction

Since the development of wave mechanics [16], the Schrödinger equation has led to an enormously deeper understanding of the attributes and behaviour of quantum mechanics. In its non-relativistic time-independent spatial form, it is written as

$$\mathbb{H}|\Psi(r)\rangle = E|\Psi(r)\rangle, \tag{1.1}$$

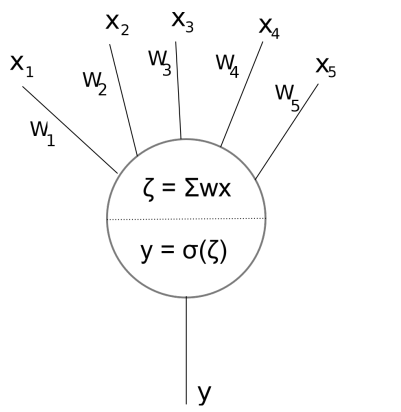
where $\mathbb{H}$ is the Hamiltonian operator, E denotes the energies of the system, r is the spatial coordinate, and the Dirac notation has been used to specify the state vectors. It is considered to be the fundamental equation of non-relativistic quantum mechanics and solving it can be regarded as an eigenvalue problem with the eigenvalues being energy. It is apparent from eq. (1.1) that the spectrum of the stationary wave functions $\{\Psi_{\text{i}} \mid i \in \mathbb{N}\}$ is exactly the eigenstates of the Hamiltonian operator $\mathbb{H}$. The real-valued eigenvalues E_{i} are the energies of the i'th state. The energy is computed by performing an inner product followed by an extraction of the scalar value E from the inner product. By simply rearranging the norm $\langle\Psi|\,\Psi\rangle$, one obtains

$$E = \frac{\langle\Psi||\mathbb{H}||\Psi\rangle}{\langle\Psi|\,\Psi\rangle}, \tag{1.2}$$

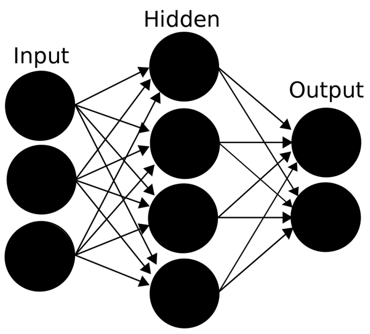
which is the expectation value of the Hamiltonian operator. In common nomenclature, the state of lowest energy is called the ground state, which is denoted by $|\Psi_0(r)\rangle$.

A lot of information in quantum mechanics can be gained from analysing the closed-form solutions to eq. (1.1). However, exact analytic solutions to eq. (1.1) with special mathematical functions can only be obtained for a limited number of physical systems. Most of the realistic systems and their associated Hamiltonians are too complex and thus a solution to eq. (1.1) is impossible to achieve. The exact spectra of eigenstates are thus undeterminable by these methods and so the problem is to find an accurate approximate representation of Ψ_{i}. This motivates the use of numerical methods for simulating the wave function. By postulating a trial wave function, $\Psi_{\text{T}}(r;\beta)$ — which is a parametrised function of spatial coordinates r and parameters β — one can try to learn the true parametrisation that reflects the nature of the quantum system, and thus approximate a solution to eq. (1.1). For now, *learn the true parameters* is equivalent to a variational based method, which for instance minimises the energy of the system. This will be described further in the next section.

In this report, the methods of artificial neural networks (ANNs) are investigated with the purpose to construct general-purpose trial wave functions to simulate the



(a) Visualisation of the multiple-input single-output neuron in the McCulloch-Pitts model [4]. The neuron applies a non-linear activation function, σ to a linear combination of its inputs.



(b) A visual aid to geometrically understand the simple feed-forward neural network structure of the multilayer perceptron with a single hidden layer.

wave function in eq. (1.1). It has been about three quarters of a century since the first mathematical model of an ANN was presented by McCulloch and Pitts [13]. ANN algorithms belong to the field of machine learning and it is a technique for information processing, which mimics the human brain. It is composed of a network of neurons — also called perceptrons, computing units, or just units. A neuron is a multiple-input multiple-output unit that applies a non-linear activation function to a linear combination of its inputs. The resultant signal is then send forward. The activation function resembles whether a neuron is activated ("*fired*") or not in the biological neural network. Instead of a simple binary-valued step-function, often a smooth activation function is chosen, with a gradual range from *off* to *on*. The choice of activation function can be important — especially the non-linearity will be necessary for the network to solve complex problems. See fig. 1.1a for a visualisation of the model of a neuron in the McCulloch-Pitts model.

There are many different network architectures and to confine the analysis, this project only treats a subcategory of the *feed-forward neural network* (FFNN) — that of the *multilayer peceptron* (MLP). The MLP is composed, as its name suggests, of multiple layers of perceptrons — at the very least three layers: an input layer, a hidden layer, and an output layer. An illustration of the MLP structure can be found in fig. 1.1b. Except for the nodes of the input layer, each node is a neuron. The input layer nodes sends an input to the hidden layer. Then each neuron send their signal to the nodes of the output layer. The output layer nodes are equivalent to a weighted sum of the given signals. Consequently, information flows from the input (the coordinate r) to the output layer (value of the trial wave function $\Psi_{\mathrm{T}}(r)$). This illustrates the philosophy of a feed-forward neural network.

Only the most elementary MLP is adopted: one with a single hidden layer and a single output layer node. The reason for this choice is its simplicity combined with its property as a universal approximator [6] [7], that is; that a FFNN with a single hidden layer and a finite number of processing units can approximate any function to arbitrary accuracy by appropriately increasing the number of units in the hidden layer.

Considering a FFNN with m hidden neurons to be mathematically equivalent to a finite weighted sum, we can write

$$N(x,\lambda) = \sum_{i=1}^{m} w_{\mathrm{i}} \sigma_i(x,\beta) = w \cdot \sigma, \tag{1.3}$$

where x is the given input vector, λ contains the parameters of the network, and σ is the activation function. In this notation, $\beta, w \in \lambda$ are the parameters of the activation function and the vector of weights respectively. These will change according to the *learning methods* described in the next section. However, the structure will remain the same. The weights resemble the synapse strenghtening in the biological brain. In other words, the desired traits will be scaled appropriately. Although there exist a great variety of options, the activation function in this project is chosen to be non-linear and smooth ($\sigma \in C^\infty$). Notably a FFNN is smooth given smooth activation functions, in accordance to eq. (1.3), and this is also expected of a physical wave function for a finite potential energy [8]. Therefore it should be possible to approximate the physical wave function to arbitrary precision by an appropriately large FFNN.

Since the very first implementation of an ANN, the field of machine learning has shown rapid advancement and today ANNs are used as essential solutions to many technological desires such as signal processing or pattern recognition. They have also been used to solve both ordinary and partial differential equations as well as the eigenvalue problem. [10] [9]. The first successful applications of ANNs as trial wave functions for quantum systems in one, two, and three dimensions were published about 20 years ago [10] [9]. However, the bulk of the work has been done over the last couple of years, where especially 2017–2019 shows an immense increase in field research. Most notably is the work of Carleo and Troyer presenting applications of ANNs in the quantum many-body problem [2].

The majority of the work draws upon the *universal approximation property* of the *feed forward neural network* (FFNN) architecture, relying upon the proof of Cybenko [3] for sigmoid activation functions and later — but for more general activation functions — of Hornik [6] [7]. A FFNN with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of Euclidian space under mild assumptions on the activation function.

This is the magnum opus for the research undertaken in this project. Several quantum mechanical systems will be considered, each resting on certain cardinal assumptions, namely that they can be described canonically by an integro-differential operator called the Hamiltonian $\mathbb{H}$ of the system. By virtue of the Rayleigh-Ritz variational principle and a method employing FFNN as trial functions proposed by Lagaris[9] (see next chapter), the parameters will be optimised to solve the Schrödinger equation, thus obtaining a numerical approximation for the eigenstates and eigenvalues of the quantum system. These solutions, Ψ, are wave functions fully describing the quantum state — be it a single particle or a complex molecule — and hence of fundamental nature in quantum mechanics. Therefore, it is of fundamental and practical interest to understand whether an ANN can learn, modify, and adapt itself to fully describe and analyse any quantum system. This ability would reveal solutions to many problems in those regimes so far inaccesible to existing numerical approaches.

In the next chapter, I will give a brief introduction to the different variationel methods which will be used in this project. Furthermore, I will explain my implementation in more detail. In Chapter 3, the results of the analysis will be presented and discussed, and finally, a summary and a discussion of future improvements can be found in Chapter 4.

2 Implementations

In the following, the reader will be made acquainted with the two methods used in this project for solving eq. (1.1). In both methods, a *cost* function is introduced. This is a mapping of a set of parameters onto a real number, which represents a *loss* associated with the parameters. This is then used as a measure for inaccuracy. By implementing a multidimensional optimisation routine, the cost function will be minimised. In this project it is the *Simplex algorithm* of Nelder and Mead that will be utilised. It considers a hypercube in parameterspace, and then transforms the corners of the hypercube such that the cost function is minimised. When the process has converged, the network is said to have *learned* the parameters for the approximate wave function.

2.1 Methods for Numerical Approximations

Method 1: Rayleigh-Ritz Variational Principle

To solve for the energy of the ground state in a systematic way, it can be convenient to take advantage of the Rayleigh-Ritz variational principle,

$$E_0 \leq E_{\mathrm{T}}, \tag{2.1}$$

where E_0 is the true ground state energy, and E_{T} is the energy associated to the trial wave function. The latter is calculated by

$$E_{\mathrm{T}} = \frac{\langle \Psi_{\mathrm{T}} | \mathbb{H} | \Psi_{\mathrm{T}} \rangle}{\langle \Psi_{\mathrm{T}} | \, \Psi_{\mathrm{T}} \rangle}, \tag{2.2}$$

where $\Psi_{\mathrm{T}} = \Psi(R; \beta)_{\mathrm{T}}$ is a parametrised trial wave function of parameters β and coordinates R. This suggests that if Ψ_{T} is optimised to minimise E_{T} by varying β then Ψ_{T} converges to the true wave function corresponding to the true ground state energy E_0. Thus, a natural choice of cost function is the energy:

$$\mathrm{Cost}(\beta) = E_{\mathrm{T}}. \tag{2.3}$$

By minimising the energy, one obtains well approximating parameters for the solution of eq. (1.1). This is an important method popularly used in the literature (e.g. [2]).

Method 2: The Residual Approach of Lagaris

Inspired by the technique for solving differential equations proposed by Lagaris [9], a collocation method is utilised. A discretisation of the domain for the wave function, D, into a set of points, R_i is required. The domain is further assumed bounded, and the wave function is assumed smooth to the highest order of derivatives of the Hamiltonian. The problem of solving the Schrödinger equation is then transformed into the more straight-forward problem of minimising the following error quantity with respect to the parameters β.

$$\mathrm{Cost}(\beta) = \sum_{R_i \in D} \frac{(\mathbb{H}\Psi_\mathrm{t} - E\Psi_\mathrm{t})^2}{\langle \Psi_\mathrm{T} |\, \Psi_\mathrm{T} \rangle} \tag{2.4}$$

where again, $\Psi_\mathrm{T} = \Psi_\mathrm{T}(R_i;\beta)$ is the parametrised trial wave function. If the obtained minimum has a value close to zero, an approximate solution can be considered found.

In contrast to the first method, the energy calculated in eq. (1.2) using the trial wave functions obtained by the second method will not be an upper limit to the natural energy of the system.

Implementation in C

As the trial wave functions Ψ_T are functions of N particles in D-dimensional space, computing the inner products in the Hilbert space ($\mathbb{L}^2$) requires integrating over a $(D \cdot N)$-dimensional space resulting in solving high-dimensional integrals. In this project, the integrals will be operated by the *QUADPACK* routines for low-dimensional problems as well as the *Monte Carlo* algorithms for both low- and high-dimensional problems. The *Monte Carlo* method is utilised with the adaptive algorithm of Lepage named *VEGAS*, which is based on the combination of importance sampling and stratified sampling. This was chosen for its favorable scaling with dimensionality. Each method will be assigned separate integration routines; the Variational Principle in which the energy is minimised will adopt the *Monte Carlo* algorithm, while the Residual Method of Lagaris, in which the difference between sides of equalty of the Schrödinger equation is minimised, will have assigned to it the *Adaptive Quadratrues* from the *QUADPACK* routines. Then, to optimise the parameters of the trial wave function, it is common in the litterature to resort to a gradient-based stochastic optimisation algorithm, such as the *Newton* or *steepest descent* method. In this thesis however, the *Simplex* algorithm of Nelder and Mead is applied. All of the above described methods are as defined in the *GSL*

— *GNU Scientific Library*. Evenmore, all programs are written in the language of C. Examples of code snippets can be found in the appendix.

Choice of Activation Functions

Among the benefits of neural networks is the flexibility to change the activation function. Each system considered in this thesis will be tested for different activation functions and compared.

In this project, a triplet of activation functions has been found convenient for testing. That is the Gaussian, Gaussian wavelet and exponential activation functions. They can each be written in the form of eq. (1.3) for the one-dimensional cases,

$$N_{\text{Gaussian}}(x,a,b,w) = \sum_{i}^{m} w_{\text{i}} \exp\left(- \left(\frac{x-a_{\text{i}}}{b_{\text{i}}} \right)^2 \right), \tag{2.5}$$

for the Gaussian,

$$\begin{aligned} N_{\text{wavelet}}(x,a,b,w) &= \sum_{i}^{m} x w_{\text{i}} \exp\left(- \left(\frac{x-a_{\text{i}}}{b_{\text{i}}} \right)^2 \right) \\ &= x N_{\text{Gaussian}}(x,a,b,w), \end{aligned} \tag{2.6}$$

for the Gaussian wavelet, and

$$N_{\text{exponential}}(x,a,b,w) = \sum_{i}^{m} w_{\text{i}} \exp\left(- \left(\frac{x-a_{\text{i}}}{b_{\text{i}}} \right) \right), \tag{2.7}$$

for the exponential function. For the quantum three-body problem, the Gaussian is rewritten:

$$N(r_1,r_2,\alpha,\beta,\gamma,w) = \sum_{i}^{n} w_{\text{i}} \exp\left(-\alpha {r_1}^2 - \beta {r_2}^2 - \gamma (r_1 - r_2)^2 \right), \tag{2.8}$$

where r_1 and r_2 is two spatial coordinate vectors with respect to the third particle. By introducing a single input vector $\hat{r} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}$, and the positive definite matrix

$$A = \begin{bmatrix} \alpha+\beta & -\gamma \\ -\gamma & \beta+\gamma \end{bmatrix}, \tag{2.9}$$

one can rewrite the exponent as

$$\begin{aligned} \alpha {r_1}^2 + \beta {r_2}^2 + \gamma (r_1 - r_2)^2 &= (\alpha+\gamma){r_1}^2 + (\beta+\gamma){r_2}^2 - 2\gamma(r_1 \cdot r_2) \\ &= \hat{r}^{\dagger} A \hat{r}. \end{aligned} \tag{2.10}$$

2.2 Solving the Excited States

When the ground state is solved for, it is natural to look for methods to solve for excited states. To do so, one can take advantage of the Gram-Schmitt process to extract from the trial wave function the previously computed levels, thus obtaining a trial wave function ϕ_{i} orthogonal to ϕ_{j} for any $j \neq i$. This is as required for an operator with a discrete spectrum. The proof for this requirement can be found in the appendix. For instance, let $\phi_0(r)$ denote a normalised ground state, then the trial wave function of the first excited level would be [9]

$$\phi_{\mathrm{t}}(r) = \hat{\phi}_{\mathrm{t}}(r) - \hat{\phi}_0(r)\langle\phi_0(r)|\,\hat{\phi}_{\mathrm{t}}(r)\rangle \tag{2.11}$$

where $\hat{\phi}_{\mathrm{t}}(r)$ is parametrised in the same way as before and r is the spatial coordinate vector. Now $\phi_{\mathrm{t}}(r)$ is orthogonal to $\phi_0(r)$ by construction.[1] This process is easily extended recursively to obtain higher excited states.

[1]To give a geometrical intuition for this result, the projection component of the state onto $\phi_0(r)$ has been removed. This is just as any ordinary vector space. However, as the inner product space is a Hilbert space ($\mathbb{L}^2$), this is computed as an overlap integral. It is also readily verified by the defn. of orthogonality, if one takes the inner product of ϕ_{t} with ϕ_0.

3 Results

Now that the methods have been described, we start off with a toy model: the one-dimensional quantum harmonic oscillator (QHO).

3.1 Quantum Harmonic Oscillator

A QHO is among the most important model systems in quantum mechanics as any arbitrary potential can be approximated in the vicinity of a stable equilibrium point as a harmonic potential. Many problems are then transformed into a local one akin the approximation of a Taylor series to its second-order term. The QHO is described in any introductory book on quantum mechanics (i.e. [5]), and in the following context it is only considered as a simple boundary value differential equation with eigenvalues. Firstly, consider the simple scaling of units to the systems natural length and energy scales, easily achieved by requiring non-dimensionality of units. This is achieved by letting the mass, m, and angular frequence ω be set to one, $m = \omega = 1$. Then the governing differential equation — the Schrödinger equation — is canonically described by the Hamiltonian given by

$$\mathbb{H} = -\frac{1}{2}\nabla^2 + \frac{1}{2}x^2, \tag{3.1}$$

where ∇^2 is the Laplacian operator and x is the spatial coordinate. This differential equation has boundary conditions $\psi(\pm\infty) = 0$. Since one cannot integrate numerically to infinity, a smart workabout is to do a substitution of a reasonably large number L, much larger than the typical size of the oscillator, but still manageable for the numerical integrator. The boundary conditions is then transformed into $\psi(\pm L) = 0$.[1] The energy levels are known analytically and are given in natural energy scales by

$$E_\text{n} = \left(n + \frac{1}{2}\right). \tag{3.2}$$

Using a FFNN with one hidden layer and m hidden units of Gaussian activation functions; the two cost function, as described in eq. (2.1) and eq. (2.4), are minimised to obtain a set of two parameter vectors, λ_1 and λ_2, each corresponding to an approximate solution. For both methods, the energy eigenvalue for the ground state is calculated by eq. (1.2). Inspired by [9], the excited states are solved as well. This was done specifically for the residual approach of Lagaris.

[1] In practice, the value chosen was $L = 10$ in the units of natural lengths.

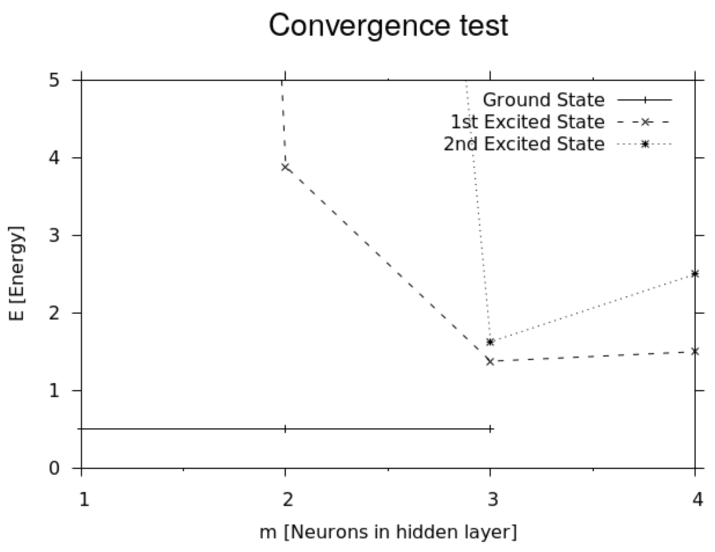


Figure 3.1: Testing a feed-forward neural network for its convergence values as it approximates the quantum harmonic oscillator energies with varying number of computing units in the hidden layer.

Afterwards, the two ground state energies for each method are compared. The results can be found in table 3.1. The two methods are found to be similar, but the integration routines are very different. The Variational Monte Carlo is often found in the literature to be more approachable for higher dimensional integrals due to the so-called *curse of dimensionality* — a phenomena coined by Richard E. Bellman [1], which refers to the problems when the dimensionality increases and the volume of the space increases so fast that the available data become sparse. This is a problem for any method that requires statistical significance as the amount of data needed to support the result often grows exponentially with the dimensionality.

Test of Convergence

Test 1: A Naive Method

Before a precise result is sought, the convergence for different network sizes is tested, in order to obtain an optimised value of m. The test is written in a simple BASH script with the purpose to run the program with a variable number of hidden

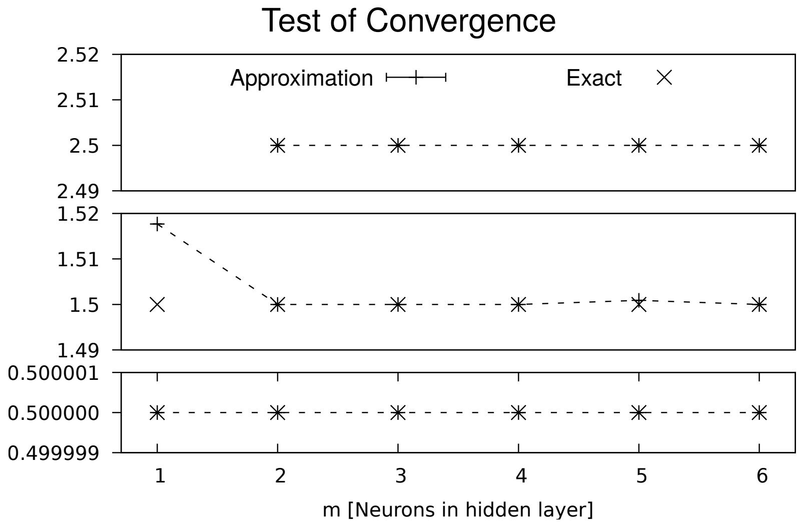


Figure 3.2: An optimised test of a feed-forward neural network for its convergence values as it approximates the quantum harmonic oscillator with varying number of computing units in the hidden layer. The three plots is for each of the ground state (lower), first (mid) and second (top) excited states.

computing units.[2] The results for the second method can be seen on fig. 3.1. It is found that the ground state is well approximated by $m = 1$ hidden computing units. This reveals a truth in the nature of the system. The exact solution to the ground state is a Gaussian function, whereas the excited states amount to Hermite polynomials[5]. Thus it is expected that the excited states need more than a single neuron. Both states need $m = 4$ hidden neurons, as seen on fig. 3.1.

As the reader might have noticed on fig. 3.1, the energies of the two excited states are both lower than the real solution for $m = 3$. This is not in violation of any principle as the method does not involve the Rayleigh-Ritz variational principle. The energy of the latter method is *not* an upper bound for the real energy and is allowed to exceed the natural energy.

Test 2: An Optimised Method

A way to optimise this test is to use the last solution for m neurons as an initial guess for the next size of $m+1$ neurons. The result is shown on fig. 3.2. Furthermore, a different number of hidden neurons for each state was tried, (say $n, l, m \in \mathbb{N}$). This

[2]An example of this BASH script can be found in the appendix

	Monte Carlo	Adaptive Quadrature		
	Ground state	Ground state	1st excited state	2nd excited state
Energy	0.50000000(3)	0.5000000000000(1)	1.50000000034993(5)	2.50000043864(7)
# Iterations	241	3637	3168	8461

Table 3.1: Comparison of energy eigenvalues and number of iterations for the quantum harmonic oscillator using a Gaussian activation function for each of the applied methods.

way each state was solved seperately. The number, n, is the size of the network as an approximation to the ground state, which after finding an optimised value is held fixed. Then, l, is the number of neurons used for calulating the first excited state, and is fed the previously optimised ground state representation for the inner products of eq. (2.11). When l is solved for, both n and l are held fixed. This method can then be repeated as necessary. As seen on fig. 3.2, each state is solved with an almost exact accuracy and for fewer neurons. However, the second excited state was off the chart for $m = 1$ and removed for visualisation purposes. To explain this deviation, it is known that the real solution for the second excited state has two nodes, which would correspond to two Gaussians. Evenmore, the error propagates, and as seen on fig. 3.2, the first excited state is imprecise for $l = 1$. This is as far as the analysis goes. For the second method, using the variational principle, only the ground state was solved. As in fig. 3.2 a single unit was enough.

Now, a simple comparison between the two methods and their solved energy eigenvalues is tabulated in table 3.1 along with the number of iterations it took the process to converge. Both methods converge to the analytical ground state energy rapidly.

It is worth mentioning that for the *Monte Carlo* method, the domain integrated over was $[-10, 10]$ and the number of sampling points — which amounts to a number of random number generator function calls —was chosen for two distinct values. First test was 2000 calls. This resulted in an energy of $0.500024(2)$. Then the number generator was called 100.000 times, thus obtaining a change of resolution of grid from 0.01 to 0.0002. The energy can be seen in table 3.1. The obtained energy for the ground state was $E = 0.5$, which is in perfect agreement with the exact value from eq. (3.2). The computed wave functions are shown in fig. 3.3 along with the analytical solutions and their residuals.

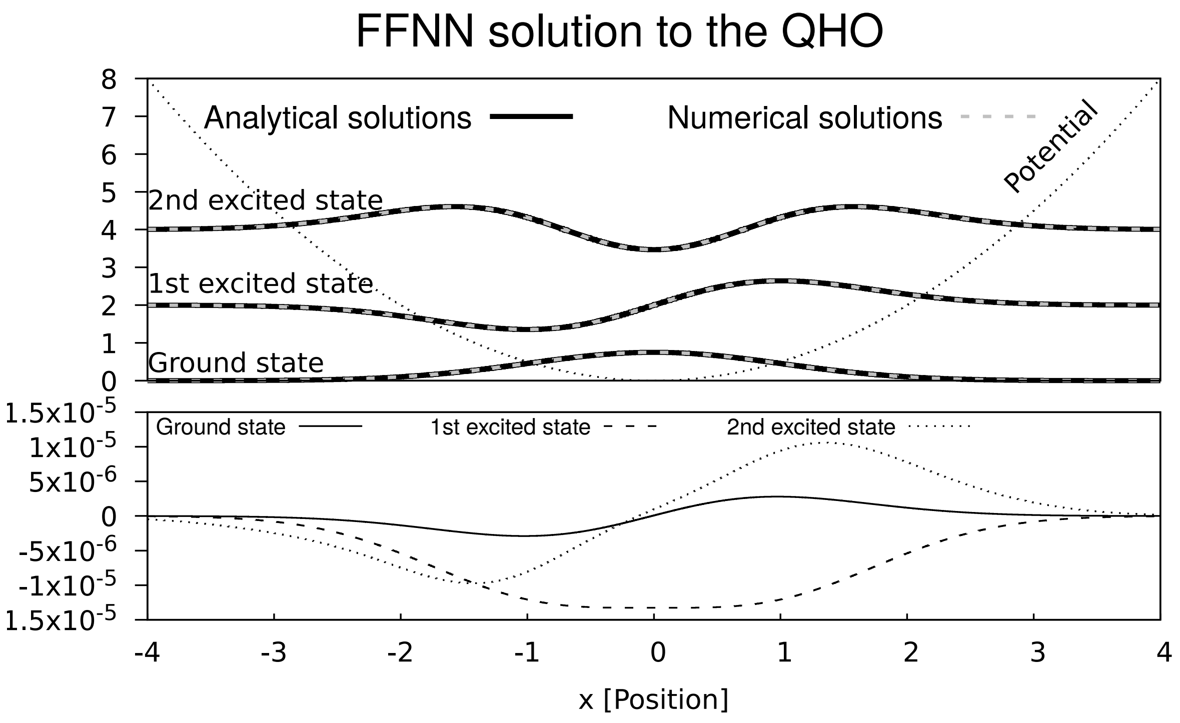


Figure 3.3: The solutions of the lowest three energy states of the quantum harmonic oscillator (upper) and their residuals with respect to the exact state (lower).

3.2 Hydrogen Atom

Now that the QHO has been solved, we move on to the more realistic and complex system; that of the hydrogen atom.

The spatial Schrödinger equation for the three-dimensional hydrogen atom can be solved exactly (see for instance [5]). Using the spherical symmetry of the Coulomb potential, the problem obtains three sets of ordinary second-order differential equations, which can be solved analytically. A typical analysis of the hydrogen atom determines the spherical harmonic functions as solutions to the azimuthal and polar angular equations. Due to the spherical symmetry of the system, only the radial equation is investigated. The radial Schrödinger equation for the hydrogen atom with no external magnetic field is described by the Hamiltonian

$$\mathbb{H} = -\frac{1}{2}\frac{\partial^2}{\partial\rho^2} - \frac{1}{\rho} + \frac{l(l+1)}{2\rho^2}, \quad \rho \in \mathbb{R}^+, \tag{3.3}$$

in dimensionless variables with boundary conditions $\psi(0) = 0 = \psi(+\infty)$. Here ρ is the radial coordinate. In the following, only the s-state is solved, corresponding to $l = 0$. In contrast to the previously solved systems, this system is better described by a Gaussian wavelet than of the Gaussian. This is due to the fall-off rate, where the Gaussian wavelet has a longer tail in contrast to the Gaussian, which falls more quickly. A short proof can be found in the appendix. Also the Gaussian wavelet already solves the boundary conditions of the system. Thus, a change of activation function is in order. The network is then described by the sum of eq. (2.6). Even further, the exact solution is known to be exponentially decaying and of the form $\Psi \propto \rho \exp(-\rho)$ [5]. For this reason, the system will be tested against a set of exponential activation functions. This will show how integrating physical insight into the network can help the convergence of the method. This type of bias is typically introduced in other numerical methods as well such as the *Quantum Variational Monte Carlo* to cleverly choose a set of trial wave functions [8].

Notice that the hydrogen atom has $\rho \in [0, \infty]$, whereas $x \in (-\infty, \infty)$ for the harmonic oscillator. This is essential to the generator of random positions for any sampling methods used in the *Monte Carlo* integrations. Also, the activation functions are distinct in nature and an initialisation of the first guess should reflect this. For the exponential, the domain was chosen strictly positive, whereas the Gaussian wavelet was preferred symmetrically about the origin. This was chosen to favor the positive pulse of the wavelet, which is similar to the wave function found analytically.

Test of Convergence

As before — to obtain a good size for the hidden layer — a test of convergence for different number of hidden computing units is performed. The results for the residual method of Lagaris can be seen on fig. 3.4. It is found that the ground state is well approximated by $m = 1$ (a single hidden computing unit) for the exponential function, whereas the Gaussian wavelet activation function varies more distinctly. A choice of $m = 3$ is best for the latter. This demonstrates yet again that the natural function is much better than a general purpose trial function just as expected.

Final results

Now, a simple comparison of the solved ground state energy eigenvalues for each method is tabulated in table 3.2. It is seen that both methods converge within an adequate tolerance to the exact ground state energy. It is worth mentioning that for the *Variational Monte Carlo* method, the domain was chosen as $[0, 10]$ with 5000 sampling points. The resulting resolution is 0.02.

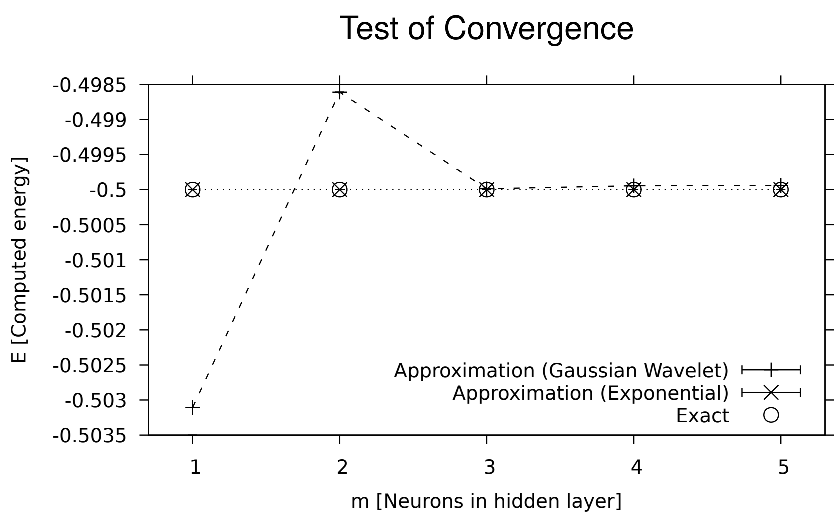


Figure 3.4: Testing of a feed-forward neural network for its convergence values as it approximates the radial wave function for the hydrogen atom with varying number of computing units in the hidden layer.

	Monte Carlo		*QUADPACK*	
	Exponential	Wavelet	Exponential	Wavelet
Energy	$-0.49997(3)$	$-0.500003(6)$	$-0.50000000000(3)$	$-0.49998651431(9)$
Iterations	1	727	1114	1

Table 3.2: A comparison of energies for the hydrogen atom, as calculated by a feed-forward neural network with two different activation functions as well as distinctive cost functions.

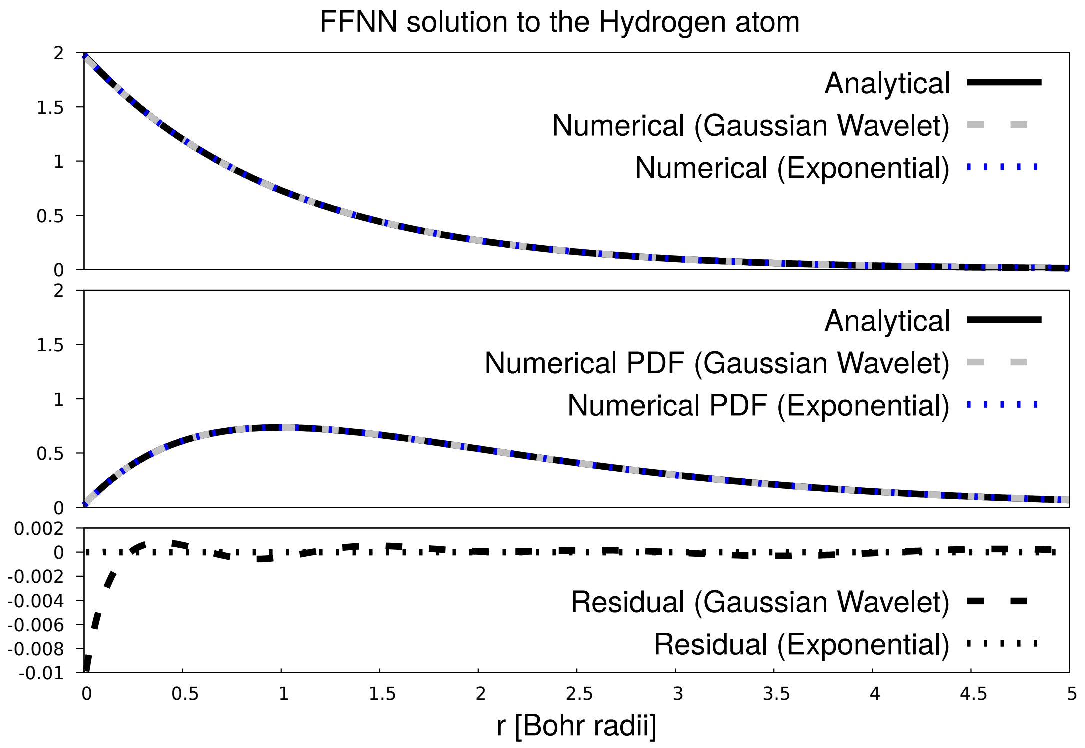


Figure 3.5: (Upper) The wave functions— (mid) The probability distribution function — as numerically approximated compared to the analytical solution. (Bottom) The residual of each method with respect to the exact solution.

3.3 The Hydrogen Anion

Moving on to a new Coulomb potential problem, the next step is the three-body problem. There are many ways to add another particle to the already-considered Coulombian problem[3] To mention a few, there is the hydrogen molecular ion containing two protons and one electron. Another is the helium atom, which although it consists of two protons and two electrons can be described under the Born-Oppenheimer approximation as a fixed centre with two elecrons. Not to mention all of the more exotic systems such as the muonic atoms. However, the simplest three-body system coming to mind is adding an electron to the hydrogen atom, creating the negatively charged hydrogen ion also called the hydrogen anion.

Now solving the electronic spatial wave functions of this system demands a new setup as described by eq. (2.8). The Hamiltonian for the hydrogen ion is

$$\begin{aligned}\mathbb{H} &= -\frac{1}{2}({\nabla_1}^2 + {\nabla_2}^2) - \left(\frac{1}{r_1} + \frac{1}{r_2} - \frac{1}{r_{12}}\right) \\ &= \mathbb{H}_1 + \mathbb{H}_2 + \frac{1}{r_{12}}, \end{aligned} \tag{3.4}$$

in dimensionless variables, where r_1 and r_2 are the spatial coordinates of each electron with respect to the proton, and r_{12} is the internal distance between the two electrons. It is easily recognised that if the distance between the two electrons is large, the $\frac{1}{r_{12}}$ can be neglected and the resulting Hamiltonian becomes a sum of two Hamiltonians of the same form as found in the hydrogenic problem eq. (3.3).

Unlike the previously solved systems, the exact ground state of the hydrogen anion is not known. Here a reference energy of $E = -0.5227715$ is used based on the energy computed in [11].

The energy as computed by this method is $-0.52(3)$, which results in a relative error of 0.41%. An analysis of the convergence of the network with a variable hidden computing units is shown on fig. 3.6.

[3]Coulombian: of or relating to the discoveries or laws of C.A. de Coulomb. `https://www.merriam-webster.com/dictionary/coulombian`

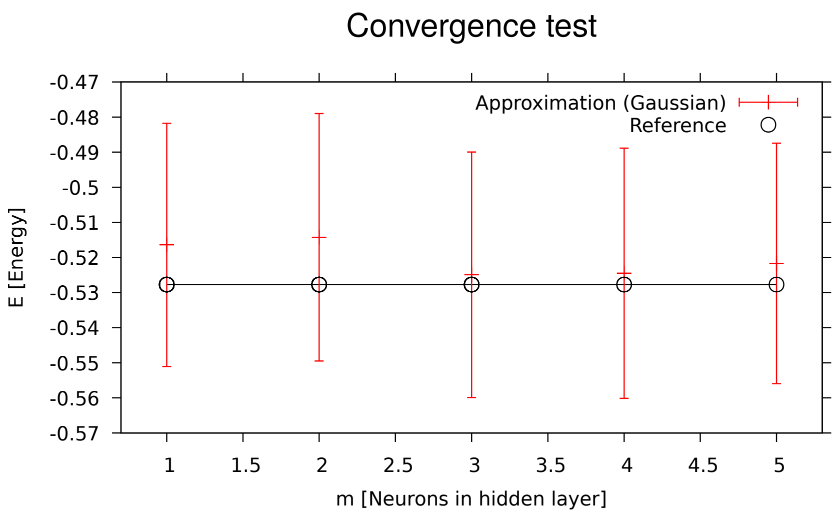


Figure 3.6: Testing of a feed-forward neural network for its convergence values as it approximates the solution for the hydrogen anion with varying number of computing units in the hidden layer.

4 Discussion

Four quantum systems have been solved with a feed-forward neural network by applying up to three different system-specific activation functions and adopting two numerical integrational routines. In the following section, the results will be discussed and the shortcomings of the method will be listed.

QHO The quantum harmonic oscillator seemed best solved by the *Adaptive Quadrature* routines in comparison to a *Monte Carlo* integration. However, the difference was miniscule. The ground state was optimised for the number of hidden computing neurons, and this number was determined to be one. This was of no surprise as the exact wave function is of Gaussian form. The two excited states were solved by the residual approach of Lagaris and best approximated by two hidden neurons. Chaining the previously solved parameter vector for $n-1$ neurons to the parameter vector of size n with last four parameters set to the default values seemed to reduce the number of neurons necessary. All energies were determined with a close to exact accuracy to the values as given:

$$\begin{aligned} E_0 &= 0.50000000000000(1) \\ E_1 &= 1.50000000034993(5) \\ E_2 &= 2.50000043864(7) \end{aligned}$$

for the *Adaptive Quadrature* and $E_0 = 0.50000000(3)$ for the *Monte Carlo* integration routine. These energies were however for a fixed precision of both the integration and optimisation routine and can be considered solved satisfactory.

Hydrogen atom The hydrogen atom was solved with both *Adaptive Quadratures* and *Monte Carlo* integration. Furthermore, activation functions of both the Gaussian wavelet and the exponential characteristics were applied. It was found to be best described using the exponential activation function. This was anticipated as the exponential activation function matches the analytical description.

The energy was solved satisfactorily. For Gaussian activation function the energy was determined to be $-0.49998651431(9)$ and $-0.500003(6)$ for the *Adaptive Quadrature* and *Monte Carlo* routines respectively. This deviates from the exact energy by 0.00269714% and 0.0006%. This however is adjustable by varying the condition of convergence for the optimisation routine and is therefore considered an accurate approximate solution. For the exponential activation function the energies were found to be almost exact.

Hydrogen anion The hydrogen anion was the only case with no full analytical description, and with its six spatial dimensions, it was only solvable using a multi-dimensional integration method. The chosen routine was that of the *Monte Carlo* using the adaptive algorithm of Lepage named *VEGAS*, based on the combination of importance sampling and stratified sampling. The energy was determined to be $-0.52(3)$ which deviates from the reference energy by an error of 0.41%. By testing the networks capability with a varying number of hidden computing units, the system seemed best solved by 3 neurons.

5 Conclusion

A method for solving quantum mechanical systems using FFNN-based trial wave functions in continuous space has been presented. This has then been employed to approximate the ground states of the quantum harmonic oscillator (QHO), Morse potential, hydrogen atom, and hydrogen anion. The accuracy of the numerical solutions were checked by comparing to analytically known results whenever possible and tabulated reference energies otherwise.

It was shown that by introducing knowledge for the respective quantum system in form of a cleverly chosen activation function, the neural network solved the eigenvalue problem with uncanny precision. Using a more general purpose activation function, the network could for some systems show undesirable features. However, this was only seen when testing for robustness and efficiency. This suggests that the method — as implemented — is not as robust as envisioned. The reason is anticipated by the author to lie in the multi-dimensional optimisation routine. It seemed to be a sensitive module with respect to initial guess, step size, and criterion for convergence.

For each system, comparisons of both cost functions and integration routines were performed. Adopting both the popular *Variational Monte Carlo* and a less pubslished method as proposed by [9], each system was solved. In doing so, both *Adopted Quadratures* and *Monte Carlo* were used. For the QHO, the method proceeded to solve for the excited states. In all cases the energies were approximated with adequate precision.

In conclusion, it has been demonstrated that a FFNN can very well represent a flexible trial wave function that can be applied to a variety of Hamiltonians, without many requirements. The structure defined in this work was only a simple architecture with only a single layer. Despite the lack of complexity, it can be considered a well approximator for all systems investigated within this project.

Future Prospects In this thesis, ANNs have been investigated as an approximator to the Schrödinger equation. However, there is still room for improvements, which the author would have liked to improve upon.

1) For future research, it would be a natural step to extend the network with a gradient-based multidimensional optimisation method, such as the *newton* or *steepest descent* method. This could then be compared with the used *Simplex* algorithm by Nelder and Mead. Based on previous work done in the field, it is expected to be more reliable in finding the correct global minima and quicker in its speed of convergence.

2) Moreover, it is expected for the proposed method to be efficiently implemented on parallel architectures.

3) Furthermore, a linear regression — before the training algorithm has begun — can find a decent initial guess. This would optimise the speed of convergence, and more safely find the correct minima in the high-dimensional parameter space, as the distance to be traveled is shorter, and the number of local minima in the way is less.

When above-mentioned improvements have been achieved, another prospect would be to look further upon the molecular hydrogen ion and more complex systems such as the helium atom. This would be a gateway into quantum molecular dynamics — an excessively investigated field of research, which is still built upon today.

Bibliography

[1] Richard E Bellman. *Adaptive control processes: a guided tour*. Vol. 2045. Princeton university press, 2015.

[2] Giuseppe Carleo & Matthias Troyer. “Solving the quantum many-body problem with artificial neural networks”. In: *Science* 355.6325 (2017), pp. 602–606. arXiv: `1606.02318 [cond-mat.dis-nn]`.

[3] George Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of Control, Signals and Systems* 2 (1989), pp. 303–314.

[4] Candace Gilet. *Artificial Neural Networks/Neural Network Basics*. Last accessed the 25th of December, 2019. 2019. URL: `https://en.wikibooks.org/wiki/Artificial_Neural_Networks/Neural_Network_Basics`.

[5] D.J. Griffiths. *Introduction to Quantum Mechanics*. Cambridge University Press, 2017.

[6] K. Hornik, M. Stinchcombe & H. White. “Multilayer Feedforward Networks Are Universal Approximators”. In: *Neural Netw.* 2.5 (1989), pp. 359–366.

[7] Kurt Hornik. “Approximation Capabilities of Multilayer Feedforward Networks”. In: *Neural Netw.* 4.2 (1991), pp. 251–257.

[8] Jan Kessler, Francesco Calcavecchia & Thomas D. Kühne. “Artificial Neural Networks as Trial Wave Functions for Quantum Monte Carlo”. In: *arXiv e-prints*, arXiv:1904.10251 (2019), arXiv:1904.10251. arXiv: `1904.10251 [physics.comp-ph]`.

[9] I. E. Lagaris, A. Likas & D. I. Fotiadis. “Artificial Neural Network Methods in Quantum Mechanics”. In: *Computer Physics Communications* 104.1 (1997), pp. 1–14. arXiv: `quant-ph/9705029 [quant-ph]`.

[10] I. E. Lagaris, A. Likas & D. I. Fotiadis. “Artificial Neural Networks for Solving Ordinary and Partial Differential Equations”. In: *Trans. Neur. Netw.* 9.5 (1998), pp. 987–1000.

[11] K. R. Lykke, K. K. Murray & W. C. Lineberger. “Threshold photodetachment of H^-”. In: *Phys. Rev. A* 43 (11 1991), pp. 6104–6107.

[12] Sergei Manzhos & Tucker Carrington. “An improved neural network method for solving the Schrödinger equation”. In: *Canadian Journal of Chemistry* 87.7 (2009), pp. 864–871. eprint: `https://doi.org/10.1139/V09-025`.

[13] Warren S. McCulloch & Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[14] Yosuke Nagaoka & Tunemaru Usui. “Symmetry of the Ground-State Wave Function of Many Particle System”. In: *Progress of Theoretical Physics Supplement* E68 (1968), pp. 392–402. eprint: `http://oup.prod.sis.lan/ptps/article-pdf/doi/10.1143/PTPS.E68.392/5216006/E68-392.pdf`.

[15] Krzysztof Pachucki. “Born-Oppenheimer potential for H_2”. In: *Phys. Rev. A* 82 (3 2010), p. 032509.

[16] E Schrödinger. *Collected Papers on Wave Mechanics*. New York: Chelsea, 1982.

6 Appendix

6.1 Morse Potential

The Morse potential is a convenient interatomic interaction model for the potential energy of a diatomic molecule. It is more successful in approximating the vibrational structure of the molecule than the quantum harmonic oscillator is. However, a review of Morse potential problems is outside the scope of this project, and the importance of the subject for both theory and applications in quantum mechanics may be found in the literature. For the Morse potential, we consider the Hamiltonian in atomic units

$$\mathbb{H} = -\frac{1}{2\mu}\nabla^2 + V(x), \tag{6.1}$$

where $V(x) = D(\exp(-2\alpha x) - 2\exp(-\alpha x) + 1)$. Now as calculated in [9], we choose the I_2 molecule with $D = 0.0224$, $\alpha = 0.9374$ and $\mu = 119406$. The energy levels are known analytically to be given as

$$E_\mathrm{n} = \left(n + \frac{1}{2}\right)\left(1 - \frac{n+1/2}{\zeta}\right)\xi \tag{6.2}$$

with $\zeta = 156.047612535$ and $\xi = 5.741837286 \cdot 10^{-4}$. The ground state energy is $E_0 = 0.28617979 \cdot 10^{-3}$. Testing the FFNN with $n = 2$, the ground state energy obtained is $0{,}286\,179\,81 \cdot 10^{-3}$. This deviates from the analytical energy with an error of 0.00007985%. The normalised unit-free wave function with the corresponding unit-free potential is shown in a fig. 6.1.

Discussion

Morse potential The Morse potential was only solved with the *adoptive quadrature* integration. It was best approximated by 2 hidden neurons and took a number of 1182 iterations to converge. The energy was solved with a relative error of 0.00007975%. This is considered a satisfactory result. The system was considered, only to compare with [9] and was therefore not investigated comprehensively.

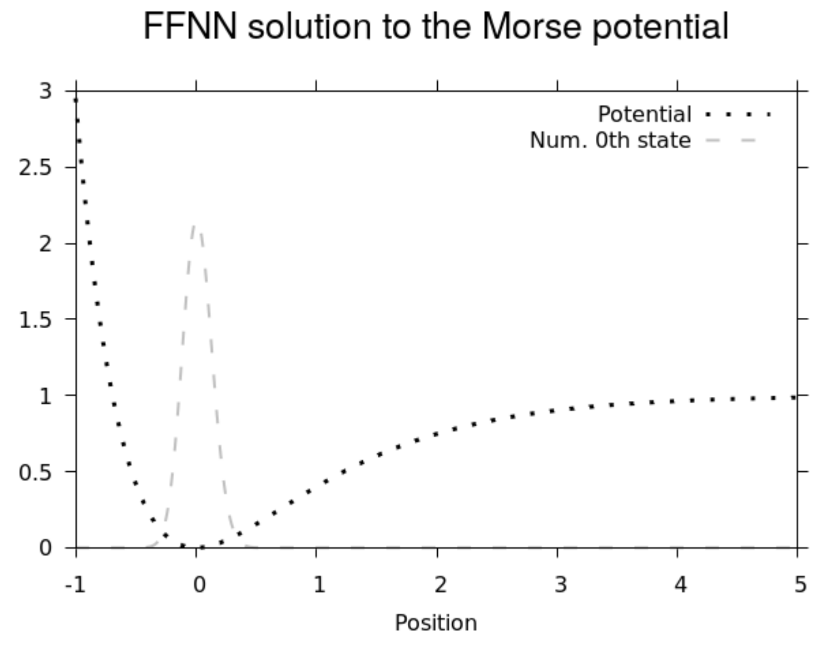


Figure 6.1: The unit-free wave function—as approximated by the implemented feed-forward neural network of this project—in the unit-free Morse potential.

6.2 Orthogonality of Stationary States of the Hermitian Discrete Operator

Let $\mathbb{H}$ be a Hermitian operator, and $|\Psi_i\rangle$ be a set of states such that $\mathbb{H}|\Psi_i\rangle = \lambda_i|\Psi_i\rangle$. Then for $\lambda_i \neq \lambda_j$ it follows that $\langle\Psi_i|\ \Psi_j\rangle = 0$.

Proof: As Ψ_i and Ψ_j are eigenstates of the hermitian operator it follows that $\mathbb{H}|\Psi_j\rangle = \lambda_j|\Psi_j\rangle$ and $\langle\Psi_i|\mathbb{H} = \lambda_i\langle\Psi_i|$. Thus the expectation value of $\mathbb{H}$ is both

$$\langle\Psi_i|\mathbb{H}|\Psi_j\rangle = \lambda_j\langle\Psi_i|\ \Psi_j\rangle \tag{6.3}$$

$$= \lambda_i\langle\Psi_i|\ \Psi_j\rangle \tag{6.4}$$

and as $\lambda_i \neq \lambda_j$, the equality is possible if and only if $\langle\Psi_i|\ \Psi_j\rangle = 0$.

6.3 Fall-off Rates

This section was written, to show the difference of fall-off rates between the used activation functions. A visualisation of the results can be seen on fig. 6.2.

Tail compared of Gaussian and Exponential

To compare the tails of the Gaussian with the exponential function, a ratio between the two values is written.

$$\frac{\exp(-x)}{\exp(-x^2)} = \exp(-x(1-x)) = \exp(x^2), \quad x >> 1,$$

it is readily seen that $\exp(x^2) > 1$, as the exponent is strictly positive and larger than one. Therefore, the ratio is strictly larger than one, so the conclusion is that $\exp(-x) > \exp(-x^2)$ for large x.

Tail comparison of Gaussian and Exponential

Following a similar path as before, it is seen that

$$\frac{\exp(-x)}{x\exp(-x^2)} = \frac{1}{x}\exp(x^2), \quad x >> 1. \tag{6.5}$$

Writing the well-known series for the exponential function, one can see

$$\frac{1}{x}\exp(x^2) = \frac{1}{x}\sum_{i=0}^{\infty}\frac{x^{2k}}{k!} = \sum_{i=0}^{\infty}\frac{x^{2k-1}}{k!} = \left(\frac{1}{x} + x + \frac{x^3}{2!} + \frac{x^5}{3!} + \dots\right), \tag{6.6}$$

where for large x the first term vanishes in contrast to the rest, which will diverge to plus infinity. Therefore, it has been well established that for large x,

$$\frac{\exp(-x)}{x\exp(-x^2)} > 1. \tag{6.7}$$

In conclusion, the tail of the Gaussian wavelet is shorter than the exponential. That is, the Gaussian wavelet converges faster to zero than the exponential.

Tail Comparison of Gaussian and Gaussian Wavelet

The exact same procedure is followed, and only shown here for completion:

$$\frac{x\exp(-x^2)}{\exp(-x^2)} = x >> 1 \tag{6.8}$$

where the last inequality is trivial for large x. Therefore $x\exp(-x^2) >> \exp(-x^2)$ for large x, so the Gaussian wavelet has a longer tail.

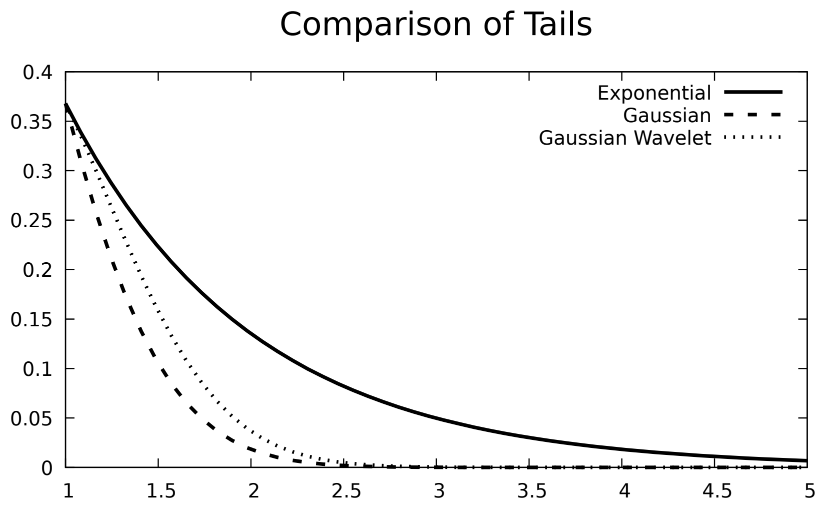


Figure 6.2: Comparison of the three applied activation functions in the implemented feed-forward neural network of this project. It can be seen that the tail of the exponential is the longest, followed by the Gaussian Wavelet and last with the shortest tail, the Gaussian function.

6.4 Code Snippets

In this section, a few snippets of code is documented. This is only for completion. In doing so, I hope to help or inspire someone, who pertains the thought of implementing an artificial neural network. Feel free to contact me for more information or collaboration.

Program for hydrogen anion

Listing 1: Example on a main.c program. This is for the hydrogen anion.

```c
// Preamble
#include <getopt.h>
#include <stdio.h>
#include <assert.h>
#include <math.h>

#include <gsl/gsl_vector.h>

#include "ann.h"
#include "system.h"
#include "hion.h"

int
main (int argc, char **argv)
{
  // Unpack parameters (old method)
  //assert (argc == 3);
  //int n = atoi(argv[1]);
  //int pts = atoi(argv[2]);
  int n = 1, t = 0;
  double pts = 1000;
  double mystep = 1e-2, eps = 1e-2;

  while (1)
  {
    int opt = getopt (argc, argv, "n:p:s:e:t:x:");
    if (opt == -1) break;
    switch (opt)
    {
      case 'n': n = atoi (optarg); break;
      case 'p': pts = atof (optarg); break;
      case 's': mystep = atof (optarg); break;
      case 'e': eps = atof (optarg); break;
      case 't': t = atoi (optarg); break;
      default:
      fprintf (stderr,

          ↪  "Usage: %s [-n neurons] [-p points monte carlo] [-s start step]\
          [-e minimise epsilon] [-t toggle activation function]\n",
          argv[0]);
      exit (EXIT_FAILURE);
    }
  }


  double (*f)(double, double, double, double, double, double)
  ↪  = NULL;
  double (*df2)(double, double, double, double, double,
  ↪  double) = NULL;

  if ((t==0))
  {
    f = &gaussian_f;
    df2 = &gaussian_df2;
  }
  else if ((t==1))
  {
    f = &exponential_f;
    df2 = &exponential_df2;
  }

  ann* network = ann_alloc (n, pts, f, df2);
  init_parameters_network (network);

  // Using previous solved state as guess
  FILE* vector_in;
  FILE* vector_out;

  // Exchanging files (n=1, n=2, n=3, n=4...)
  if (n < 2)
  {
    vector_out = fopen ("stream_out", "w");
    vector_in = fopen ("stream_in", "w");
  }
  else{
```

```c
if (n % 2 == 1)
{
  vector_in  = fopen ("stream_in" , "r");
  vector_out = fopen ("stream_out", "w");
}
if (n % 2 == 0)
{
  vector_in  = fopen ("stream_out", "r");
  vector_out = fopen ("stream_in", "w");
}
}

if (n>1)
{
  gsl_vector* w = gsl_vector_alloc (4*(n-1));
  gsl_vector_fscanf (vector_in, w);
  for (int i=0; i<4*(n-1); i++)
  {
    gsl_vector_set (network->data, i, gsl_vector_get(w, i));
  }
  gsl_vector_free (w);
}

gsl_vector* p = gsl_vector_alloc (4*n);
gsl_vector_memcpy (p, network->data);

// This is for optimization purposes
ann_train_nmsimplex2 (p, network, &cost, mystep, eps);
gsl_vector_memcpy (network->data, p);
gsl_vector_fprintf (vector_out, p, "%lg");
gsl_vector_fprintf (stdout, p, "%lg");

double E_num, E_err;
E_hion (network, &E_num, &E_err);
printf("n\tE_val\n");
printf("%i\t%lg\n", n, E_num);

double E_tab = -0.5277159;
double percent_err = 100.*fabs(E_num - E_tab) / fabs(E_tab);
printf("The energy of this method gives:\t% .8f\n", E_num);
printf("The known energy is:\t% .8f\n", E_tab);
printf("This gives a percentage error of:\t% .8f\n",
↪   percent_err);
```

```
  FILE* gp = NULL;
  if ((t == 0)) { gp = fopen ("neurons0.data", "a"); }
  if ((t == 1)) { gp = fopen ("neurons1.data", "a"); }
  fprintf(gp, "%i\t% .8g\t%lg\t%lg\\n", n, E_num, E_err,
  ↪  E_tab);


  fclose(gp);
  fclose(vector_in);
  fclose(vector_out);
  ann_free (network);
  gsl_vector_free(p);
  return 0;
}
```

Header definition for Neural Network

Listing 2: Example on the header definitions written for the artificial neural network. This is for the hydrogen anion.

```
#include <stdio.h>

#include <gsl/gsl_vector.h>
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_errno.h>
#include <gsl/gsl_multimin.h>

#include "ann.h"
#include "system.c"

ann*
ann_alloc (int n, int pts,
      double(*f)(double, double, double, double, double,
      ↪  double),
      double(*df2)(double, double, double, double, double,
      ↪  double)
)
{
  ann* network = malloc (sizeof(ann));
  network -> n    = n;                      // Neurons
  network -> pts  = pts;                    // Points for
  ↪  monte carlo
  network -> data = gsl_vector_alloc (4*n);//+1); //
  ↪  Parameters (a,b,c,w)
  network -> r    = gsl_vector_alloc (6);   // Coordinates of
  ↪  electrons
  network -> f    = f;
  network -> df2  = df2;
  return network;
}

void
init_parameters_network (ann* network)
{
  gsl_vector_set_all (network->data, 1);
}

void
```

```c
ann_free (ann* network)
{
  gsl_vector_free (network -> data);
  gsl_vector_free (network -> r);
  free(network);
}

double
ann_feed_forward (ann* network)
{
  gsl_vector* p  = network -> data;
  int n = network -> n;

  double r1, r2;
  calc_norms_of_r1_r2 (network, &r1, &r2);

  // Parameters
  double sum = 0, a, b, c, w;
  for (int i=0; i<n; i++)
  {
    a = gsl_vector_get (p, 4*i+0);
    b = gsl_vector_get (p, 4*i+1);
    c = gsl_vector_get (p, 4*i+2);
    w = gsl_vector_get (p, 4*i+3);

    // Definite Positive Problem
    a *= a;
    b *= b;
    c *= c;
    //sum += w*exp(-(a+c)*r1*r1 - (b+c)*r2*r2 + 2*c*r1*r2);
    sum += (network->f)(r1, r2, a, b, c, w);
  }
  return sum;
}

int
ann_train_nmsimplex2 (gsl_vector* p, ann* network, double
↪  (*cost)(const gsl_vector*, void*), double step, double
↪  eps)
{
  // Dimension of parameterspace
  int n = p->size;

  int status, iter = 0;
```

```c
// Multimin function setup
gsl_multimin_function my_func;
my_func.n = n;
my_func.f = cost;
my_func.params = (void*) network;

/* Starting point */
gsl_vector* my_guess = gsl_vector_alloc (n);
gsl_vector_memcpy (my_guess, p);

/* First Step Size */
gsl_vector* step_size = gsl_vector_alloc (n);
gsl_vector_set_all (step_size, step);

const gsl_multimin_fminimizer_type *T;
gsl_multimin_fminimizer * s;

T = gsl_multimin_fminimizer_nmsimplex2;
s = gsl_multimin_fminimizer_alloc (T, n);

gsl_multimin_fminimizer_set (s, &my_func, my_guess,
↪  step_size);
do
{
  iter++;
  status = gsl_multimin_fminimizer_iterate (s);

  if (status) break;
  double size = gsl_multimin_fminimizer_size (s);
  status = gsl_multimin_test_size (size, eps);

  if (status == GSL_SUCCESS)
  {
    printf ("converged to minimum!\n");
    gsl_vector_memcpy(p, s->x);
  }
}
while (status == GSL_CONTINUE && iter < 10000000);
printf("Max Iter:%i\n", iter);

gsl_vector_memcpy(p, s->x);
gsl_multimin_fminimizer_free (s);
gsl_vector_free (my_guess);
```

```
  gsl_vector_free (step_size);

  return GSL_SUCCESS;
}
```

Costfunction

Listing 3: Example on a program to calculate the costfunction. This is for the hydrogen

```
// Preamble
#include <stdio.h>
#include <math.h>
#include <assert.h>

#include "ann.h"
#include "system.h"

#include <gsl/gsl_vector.h>
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_monte.h>
#include <gsl/gsl_monte_vegas.h>
/*
↪  ............................................................
 * Functionals for minimization of energy
 * <psi|H|psi> = integral(psi*H*psi), where we evaluated for
↪  SHO
 * <psi|psi> = integral(psi*psi)
 *
 * the hamiltonian (differential operator):
 * H = (-1/2)*grad^2 + (1/2)*x*x
 *
 * on paper with psi as given:
 * (Ground state:)
 * psi = sum_{i} (w_i * exp( ( (x-a_i)/b_i )^2 ))
 * (Excited states:)
 * psi_n = f(x) - psi_(n-1) <psi|f(x)>
 *
 * where f(x) is the activation function
 *
 * ............................................................
↪  */


// Calculate H|psi>
double
Hpsi (ann* network, double psi)
{
  double grad2r1, grad2r2, r1, r2, r12;
  calc_norms_of_r1_r2_r12 (network, &r1, &r2, &r12);
  smartgrad (network, &grad2r1, &grad2r2);

  assert (r1 != 0 && r2 != 0 && r12 != 0);
  double Hpsi = (-1./2.)*(grad2r1 + grad2r2) - (1./r1 + 1./r2
  ↪  - 1./r12)*psi;

  // If we neglect r12 (compare to solutions of H)
  //double Hpsi = (-1./2.)*(grad2r1 + grad2r2) - (1./r1 +
  ↪  1./r2)*psi;

  return Hpsi;
}

// Integrands
double
psiHpsi (double x[], size_t dim, void* params)
{
  // Unpack parameters
  ann* network = (ann*) params;

  int n = (network -> r) -> size;
  for (int i=0; i<n; i++){gsl_vector_set (network -> r, i,
  ↪  x[i]); }
  double psi = ann_feed_forward (network);
  double Hpsi_val = Hpsi (network, psi);
  double result = psi*Hpsi_val;
  return result;
}

double
psipsi (double x[], size_t dim, void* params)
{
  ann* network = (ann*) params;
  for (int i=0; i<(network->r->size); i++)
  {
    gsl_vector_set (network->r, i, x[i]);
  }
  double psi = ann_feed_forward (network);
  double result = psi*psi;
  return result;
}
```

```c
void
E_hion (ann* network, double* E, double* E_err)
{
  //TRACE ("Entering E_hion");
  size_t dim = 6;
  size_t calls = (size_t) network -> pts;
  double pp, pHp, pp_err, pHp_err;

  const gsl_rng_type* T = gsl_rng_default;
  gsl_rng* r = gsl_rng_alloc (T);

  // Monte considers (xl, xu) open
  double xl[] = {1e-1, 1e-1, 1e-1, 1e-1, 1e-1, 1e-1};
  double xu[] = {3, 3, 3, 3, 3, 3};

  gsl_monte_function F;
  F.f      = &psiHpsi;
  F.dim    = dim;
  F.params = network;

  gsl_monte_function G;
  G.f      = &psipsi;
  G.dim    = dim;
  G.params = network;

////////////////////////////////////////////////////////////////////////////////
  gsl_monte_vegas_state* s = gsl_monte_vegas_alloc (dim);
  gsl_monte_vegas_integrate (&F, xl, xu, dim, calls, r, s,
   ↪  &pHp, &pHp_err);

  do
  {
    gsl_monte_vegas_integrate (&F, xl, xu, dim, calls, r, s,
     ↪  &pHp, &pHp_err);
    //printf ("result = % .6f sigma = % .6f "
    //        "chisq/dof = %.1f\n", pHp, pHp_err,
     ↪  gsl_monte_vegas_chisq (s));
  }
  while (fabs (gsl_monte_vegas_chisq (s) - 1.0) > 0.1);
//  printf ("pHp done\n");
////////////////////////////////////////////////////////////////////////////////
  gsl_monte_vegas_init (s);
  gsl_monte_vegas_integrate (&G, xl, xu, dim, calls, r, s,
   ↪  &pp, &pp_err);
  do
  {
    gsl_monte_vegas_integrate (&G, xl, xu, dim, calls, r, s,
    ↪  &pp, &pp_err);
//    printf ("result = % .6f sigma = % .6f "
//            "chisq/dof = %.1f\n", pp, pp_err,
↪  gsl_monte_vegas_chisq (s));
  }
  while (fabs (gsl_monte_vegas_chisq (s) - 1.0) > 0.1);
//  printf ("pp done\n");
///////////////////////////////////////////////////////////////////////////////

  *E = pHp/pp;
  *E_err = sqrt( pow((pHp/(pp*pp))*pp_err, 2) +
  ↪  pow((1./pp)*pHp_err, 2) );

  network -> norm = pp;

  gsl_monte_vegas_free(s);
  gsl_rng_free (r);
}

double
cost (const gsl_vector* p, void* params)
{
  ann* network = (ann*) params;
  assert (network->data->size == p->size);
  gsl_vector_memcpy (network->data, p);

  //double lambda = gsl_vector_get(network->data,
  ↪  4*(network->n));

  double E, E_err;
  E_hion (network, &E, &E_err);

  return E + (1-network->norm);
}
```

Makefile

Listing 4: Example on a MAKE script. This is for the hydrogen anion.

```
# Compiler
CC = gcc
CFLAGS = -Wall -pedantic -lm $$(gsl-config --cflags)
LDLIBS = $$(gsl-config --libs)

CFLAGS += -Ofast -O2
#CFLAGS += -DNDEBUG
#CFLAGS += -g
#LDLIBS += -pg

# main is called with (n, pts, step, eps)
hion.txt: main
        ./$< -n 3 -p 3000 -t 1  > $@

main: main.o hion.o ann.o

# This is for gaussian trial functions
mybash:
        echo "\
        #!/bin/bash\n\
        for i in 1 2 3 4 5\n\
        do\n\
  ./main -n "'$$i '" -p 2000 -s 1e-2 -e 1e-6 -t 0\n\
        done\n\
        for i in 1 2 3 4 5\n\
        do\n\
  ./main -n "'$$i '" -p 2000 -s 1e-2 -e 1e-6 -t 1\n\
        done\n\
        " > $@
        chmod u+x mybash

neurons.pdf: neurons0.data neurons1.data
        echo '\
        set terminal pdf;\
        set output "$@";\
        set tics out;\
        set xlabel "m [Neurons in hidden layer]";\
        set ylabel "E [Energy]";\
        set xrange [1:9];\
        set title "Convergence test" font "Helvetica, 20";\
        set arrow 10 from 1,-0.5277159032 to 6,-0.5277159032
        ↪ nohead;\
        plot\
        "$<" u 1:2:3 w yerrorbars pt 1 lc "red" t
        ↪ "Approximation (Gaussian)"\
        ,"$(word 2, $^)" u 1:2:3 w yerrorbars pt 2 lc "blue" t
        ↪ "Approximation (Exponential)"\
        ,"$(word 2, $^)" u 1:4 w p pt 6 lc "black" t "Exact";\
        set key default;\
        '|gnuplot
        evince $@


#,"$(word 2, $^)" with lp dt 2 lc "black" t "Exponential";\

profiling:

# Monitor file
gmon.txt: main gmon.out
        gprof $^ > $@

gmon.out: main
#================================================================================#

.PHONEY:clean
clean:                      # this is "clean" target. it does
↪ not depend on anything
        find ./ -type f -executable -delete
        $(RM) *.dat $(SCRIPT) *.svg *.png *.pdf log* *.eps *.o
        ↪ gmon* vector_in vector_out
```