Computational Physics - The Variational Principle

May 2020

Contents

List of Figures

List of Tables

Abstract

The Variational principle of Quantum Mechanics, which states that

$$\langle\psi|\hat{H}|\psi\rangle = E = \frac{\int_{-\infty}^{\infty}\psi^*\hat{H}\psi d\bar{r}}{\int_{-\infty}^{\infty}\psi^*\psi d\bar{r}} \geq E_0$$

was applied to create a numerical solver for bound states of the Time Independent Schrödinger Equation. Python code was developed to systematically select trial distributions with lower and lower energies to converge down to the ground state energy and the ground state distribution. This code was further extended to handle orthonormal states and higher dimensions. Energy eigenstates were computed for analytically solved potential systems in multiple dimensions, and these computed states compared to the known solutions to verify the validity of the numerical results. Once the code was verified to solve for the correct states of known systems, the program was set to solve for states which could not be easily solved for analytically. These investigations showed that the code could be easily implemented to simulate and investigate the behaviour and distributions of the energy eigenstates and energy eigenvalues of particles in bound potential systems.

1 Introduction

The aim of this project was to implement a computational solver for the Time Independent Schrödinger Equation in python, using the methods of the Variational Principle. The Variational Principle in Quantum Mechanics states that for any normalisable distribution ψ [1, p. 417][2]:

$$\langle\psi \mid \hat{H} \mid \psi\rangle = E = \frac{\int_{-\infty}^{\infty}\psi^*\hat{H}\psi d\bar{r}}{\int_{-\infty}^{\infty}|\psi|^2 d\bar{r}} \geq E_0$$

where E_0 is the exact ground state energy. The computational solver was implemented to solve for the energy eigenstates of a known bound potential system, by allowing an initial trial distribution to converge to lower energy distributions, resulting in an upper bound estimate for the actual distribution ψ. The code was implemented to solve for any number of dimensions, and for a desired number of energy eigenstates of the system. The resultant distributions obtained from known systems were compared against analytical solutions from Griffiths' *"Introduction to Quantum Mechanics* 3^{rd} *Edition"*, to verify the accuracy and validity of the program results.

2 Background Theory

The Time Independent Schrödinger Equation (TISE) states:

$$-\frac{\hbar^2}{2m}\nabla^2\psi(\bar{r}) + V(\bar{r})\psi(\bar{r}) = E\psi(\bar{r})$$

where $\psi(\bar{r})$ is the spatial component of the separable wavefunction $\Psi(\bar{r}, t)$, which is given by [1, p. 43]:

$$\Psi(\bar{r}, t) = \psi(\bar{r})e^{-i\frac{E}{\hbar}t}$$

The TISE can be stated as an eigenvalue problem $\hat{H}\psi = E\psi$ in $\hat{H}$, where H is the Hamiltonian given as:

$$H = -\frac{\hbar^2}{2m}\nabla^2 + V(\bar{r})$$

and ∇^2 is the general Laplacian operator. The distributions ψ are energy eigenstates of the Hamiltonian, with eigenvalue E, the energy of the wavefunction $\Psi(\bar{r}, t)$. The Hamiltonian is a hermitian operator, and the assumption is made that all the trial distributions, ψ_i, chosen in the program are normalisable, thus the energies of these trial distributions are always real [1, p. 128], given by:

$$\langle E \rangle = \langle \psi_i \mid \hat{H} \mid \psi_i \rangle \in \mathbb{R}$$

Since the Hamiltonian and associated energy eigenvalue E are real, the ψ distribution can be chosen to be real up to an arbitrary phase, without any loss of generalisation. The implementation of the Variational Principle in this project leverages this assumption to reduce the complexity of the methods implemented.

As the project aim is to implement a computational solver for the TISE, we are restricted by the discretisation limit of the computer in implementing these solutions. As such, the numerical solver is limited to solving only for discrete spectra of the TISE and cannot solve for continuous spectra [1, pp. 128-132]. This results in the condition that all eigenfunctions with distinct eigenvalues are orthogonal [1, p. 128], a condition that is imposed for all computed eigenstates to account for the occurrence of degeneracy. Bound states are states where both the energy eigenvalues and energy eigenstates are discrete and the energy eigenstates are normalisable. The requirement of a discrete spectra and the normalisation assumption in the computation means the program can only solve for bound states. A bound energy state is one where [1, p. 83]

$$E < V(-\infty) \quad and \quad E < V(+\infty)$$

3 Method

All code in the project was written in Python, utilising the SciPy, NumPy and Matplotlib [6, 7, 9] scientific python libraries. The coding of the project was broken up into four main sections, as outlined below:

1. 1D Ground State
2. Higher Order States
3. Higher Dimensions
4. Plotting

The program solves for the energy eigenstates $\psi(\bar{r})$ of a given potential $V(\bar{r})$. In the program, the ψ state is approximated as a column vector, using the NumPy ndarray object [8]. The usage of such an approximation in the numerical method has it's benefits and draw backs. The ndarray object enables the usage of SciPy and NumPy built in matrix methods that are heavily optimised, leading to faster and more readable code. By taking ψ as a column vector, the Dirac Braket notation for Quantum Mechanics can be easily implemented in vector notation, to conceptually simplify the code notation. The drawback of this methodology is that the program is confined to only being able to solve for bound potential states from the discretisation limitation, as outlined in Background Theory above.

3.1 1D Ground State

The Variational Principle states that for any normalisable distribution ψ_i:

$$\langle \psi_i \mid \hat{H} \mid \psi_i \rangle = E_i = \frac{\int_{-\infty}^{\infty} \psi_i^* \hat{H} \psi_i d\bar{r}}{\int_{-\infty}^{\infty} \psi_i^* \psi_i d\bar{r}} \geq E_0$$

where E_0 is the exact ground state energy of the potential system.

The ground state distribution ψ_0 is the only distribution with energy E_0 as it is a non-degenerate eigenstate of the Hamiltonian H with eigenvalue E_0 [1]. Thus a computational method can be created to sequentially change the given distribution ψ_i and select the changes that produce a lower energy distribution, allowing the distribution to converge to the ground state energy E_0, and thus producing the ground state distribution ψ_0.

3.1.1 Initial Method

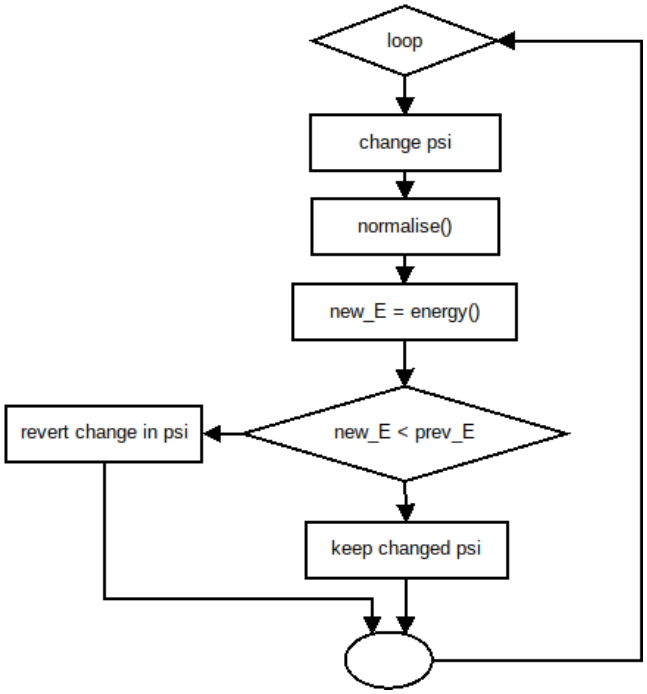


Figure 1: Flow diagram of the numerical method for calculating ψ_0.

The core design of the method of solving for ψ_0 is shown in Figure 1. The program iterated for a given number of iterations, generally 10^5 iterations were chosen for the 1D case with 100 points. Each iteration a random entry in ψ_i would be chosen, and a random number chosen from $[-1, 1]$ to alter that entry by. The random alteration was scaled by a factor of 10, as the normalisation condition imposed that each entry in ψ_i was generally less than 1 for a well spread distribution. This number was also further weighted by a factor $\frac{n-i}{n}$, where n is the total number of iterations, and i is the current iteration number, causing the scale of the change applied to ψ_i to reduce as the total number of iterations was increased. This weighting was applied as it is assumed that for an increasing number of iterations the trial distribution converges to the proper distribution, a smaller change is required each iteration to reach the proper distribution. Each iteration the new changed distribution would be re-normalised, and it's new energy calculated, as described below. The energy of the previous distribution and the new distribution were then compared, and the distribution kept that had the lower energy. In this way the program minimised

the energy of the distribution every iteration, eventually converging upon the approximate ground state distribution, ψ_0 after a sufficient number of iterations.

3.1.2 Normalisation

The normalisation condition for ψ is given as:

$$\int_{-\infty}^{\infty} |\psi|^2 d\bar{r} = 1$$

When the current distribution ψ_i is changed by a random amount as described above, the new distribution obtained needs to be normalised to meet the normalisation condition. For all normalisable distributions, the distribution tends to zero at the boundaries, it can be assumed therefore that the distribution is exactly zero at the boundaries, so the normalisation condition above can be simplified to a finite integral:

$$\int_{a}^{b} |\psi|^2 d\bar{r} = 1$$

where a & b are the bounds of the system. ψ_i is discretised, so that the integral was approximated using a right hand Riemann Sum approximation [3]:

$$S_n = h \sum_{k=1}^{n} f(a + hk)$$

where $h = \frac{b-a}{n}$. For the approximation of the normalisation, h was given as Δr, the spacing between sampled points in the distribution, and the f vector to sum over is simply the $|\psi_i|^2$ distribution. Since ψ_i is real for bounded states, f is given as ψ_i^2, so that the integration is approximated as

$$S_n = \Delta r \sum_{k=1}^{n} \psi_{i_k}^2$$

and the normalised new distribution ψ_i' is given by

$$\psi_i' = \frac{\psi_i}{\sqrt{S_n}}$$

3.1.3 Energy

The expectation value of the Hamiltonian on the distribution ψ_i is the energy of that distribution [1], $\langle H \rangle = \langle E \rangle$, where the expectation value is given as:

$$\langle H \rangle = \int_{-\infty}^{\infty} \psi_i^* \hat{H} \psi_i d\bar{r}$$

as before with the normalisation condition, the integral was approximated using a right hand Riemann Sum approximation [3]:

$$S_n = h \sum_{k=1}^{n} f(a + kh)$$

$h = \Delta r$ as before, since the integration is along the same grid, with the same grid spacing. The f vector to sum over is given as $\psi_i \hat{H} \psi_i$, since ψ_i is strictly real.

In the general case, the Hamiltonian operator is

$$\hat{H} = -\frac{\hbar^2}{2m}\nabla^2 + V(\bar{r})$$

where $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} + ...$ For this initial code, we are confined strictly to the 1D case, where $\psi(\bar{r}) \to \psi(x)$, so the Laplacian is given simply as $\nabla^2 = \frac{d^2}{dx^2}$.

A central finite difference method was used to calculate the second derivative of these ψ_i distributions [5]:

$$\frac{d^2 f}{dx^2} = f''(x) = \frac{f(x-h) - 2f(x) + f(x+h)}{h^2}$$

this finite difference second derivative was calculated in matrix algebra notation:

$$\frac{d^2}{dx^2} f = D_x f = \frac{1}{h^2} \begin{pmatrix} -2 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & -2 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -2 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 \end{pmatrix} f$$

where f represents the ψ_i column vector.

The expectation energy of the ψ_i distribution is now given as:

$$E_i = \Delta r \sum_{k=1}^{n} \psi_{i_k} (D_x \psi_{i_k} + V(x)\psi_{i_k})$$

3.2 Higher Order States

Once the program for generating the 1D ground state distribution of the known potential system was completed, the program was extended to solve for higher energy states of the same potential system. For higher order states, the new states have to be mutually orthogonal to all previous lower order states [1, p.51],

$$\langle \psi_m \mid \psi_n \rangle = 0 \quad (m \neq n)$$

as the previous states are all normalised, once the orthogonality condition is met, the orthonormal condition is met as well.

3.2.1 Composite Psi

In the Dirac Notation, "vectors are represented, with respect to an orthonormal basis $\{|e_n\rangle\}$, by their components," [1, p.149]

$$|\alpha\rangle = \sum_n a_n |e_n\rangle$$

the $\{|e_n\rangle\}$ basis spans the Hilbert Space, whose size is determined by the number of points in the discretisation of the problem in the numerical solver, there are the same number of bases as there are discretised points in the 1D line.

In the previous 1D ground state solver, the program generated a number at a random entry to modify ψ_i by. The subtle assumption made in this method is that the ground state basis $\{|e_n\rangle\}_0$ is given as the columns of the identity matrix,

$$\{|e_n\rangle\}_0 = \left\{ \begin{pmatrix} 1 \\ 0 \\ \vdots \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ \vdots \end{pmatrix}, \dots \right\}$$

the code could then be interpreted as modifying the a_n real coefficients across ψ_i, to minimise it's energy. Using this interpretation, the implementation of the code was changed to represent ψ_i in this vector notation,

$$|\psi_i\rangle = \sum_n a_n |e_n\rangle_i$$

and the program was now changed to modify this a_n by a random value against an explicit basis $|e_n\rangle_i$ across all of $|\psi_i\rangle$.

For ψ_i represented in vector notation, the orthogonality condition now becomes,

$$\langle \psi_i \mid \psi_j \rangle = \sum_m \sum_n a_m a_n \langle e_m \mid_i \mid e_n \rangle_j = 0$$

since $\{|e_n\rangle\}_i$ is defined as an orthonormal basis, so that $\langle e_m \mid_i \mid e_n \rangle_j = 0$.

The variational principle condition

$$\langle \psi \mid H \mid \psi \rangle = E = \frac{\int_{-\infty}^{\infty} \psi \hat{H} \psi dx}{\int_{-\infty}^{\infty} \psi^2 dx} \geq E_0$$

becomes

$$\sum_n a_n^2 \{\langle e_n \mid H \mid e_n \rangle\}_i = E = \frac{\int_{-\infty}^{\infty} \psi \hat{H} \psi dx}{\int_{-\infty}^{\infty} \psi^2 dx} \geq E_i$$

for the ground state, $\{|e_n\rangle\}_i = \{|e_n\rangle\}_0$ the ground state basis, so that the numerical solver converges to the ground state distribution with ground state energy. For $i > 0$, the higher order states, the solver converges to this higher energy state E_i with corresponding distribution $|\psi_i\rangle$, with an appropriate choice of orthonormal basis $\{|e_n\rangle\}_i$. The code therefore was adapted to solve for these higher order states once the new orthonormal bases $\{|e_n\rangle\}_i$ were determined.

3.2.2 Null Space

The concept of a null space was used to determine the higher order orthonormal bases. The *nullspace* of A, an m x n matrix, is defined as [4]:

$$nullspace(A) = \{x \in \mathbb{R}^n : Ax = 0\}$$

From the rules of matrix multiplication, this is the equivalent of saying that the rows of A are orthogonal

to the columns of x.

Thus if we define

$$A = \begin{pmatrix} |\psi_0\rangle \\ |\psi_1\rangle \\ \vdots \end{pmatrix}$$

each row of A being the previous $|\psi_i\rangle$ states to be orthogonal to, then the columns of x, the nullspace of A, will be orthogonal to all previous $|\psi_i\rangle$ states. These column vectors can be used to define the new orthonormal basis $\{|e_n\rangle\}_{i+1}$, needed for the computation of the new higher order state.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \rightarrow \begin{pmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{31} \\ a_{32} \\ a_{33} \end{pmatrix}$$

Figure 2: Reshaping a 3x3 matrix to a 9x1 column vector

3.3 Higher Dimensions

Up to this point, the numerical solver had only been able to solve for states in the 1D case. The task of extending the program to handle higher dimensions was the most challenging part of this project. For the 1D case, the code relied heavily on matrix algebra notation, in the usage of the Dirac Braket notation, and in the calculation of numerical integrals and derivatives. These calculations are easy to perform in the 1D case, as the 1D line can be represented as a column vector, so that linear algebra notation can be adopted readily. In higher dimension cases this is not so easy, as the system is no longer a line, and is instead a grid, volume, or some higher dimension shape.

The core principle used in overcoming this hurdle, was the concept of reshaping these higher dimension arrays back to 1D column vectors. This concept is best illustrated in Figure 2 which shows a 3x3 matrix being reshaped to a 9x1 column vector, producing an equivalent column vector that the methodology employed for the 1D states can be applied to after some minor changes in order to be able to handle these higher dimension states.

For simplicity of adapting the previous code to handle these new higher dimension column vectors, the bounds and discretisation of the system was taken as a symmetric box model, so that the grid bounds a & b and grid spacing Δr employed previously is the same across all dimensions. Using this constraint, the numerical integration approximations using the right hand Riemann sum are unchanged.

$$S_n = \Delta r \sum_{k=1}^{n} f(a + kh)$$

The representation of ψ_i as

$$|\psi_i\rangle = \sum_n a_n |e_n\rangle_i$$

in vector notation is also unchanged, as the code could intrinsically adapt to sample across column vectors of different lengths, depending on the level of discretisation approximation. By the same reasoning, the calculation of the orthonormal bases $\{|e_n\rangle\}_{i+1}$ is unaffected by this change, as each of these new equivalent column vectors can still be taken as the rows in A to generate these basis vectors.

The only section of the code that needed major reworking, was the calculation of the energy expectation value of the $|\psi_i\rangle$ distributions. The general Hamiltonian is given as

$$\hat{H} = -\frac{\hbar^2}{2m}\nabla^2 + V(\bar{r})$$

where ∇^2 is the Laplacian operator, which in the 1D case is given as $\nabla^2 = \frac{d^2}{dx^2}$, the calculation of the Laplacian needed to be extended from the 1D case to higher dimensions.

3.3.1 The Laplacian Operator

The general Laplacian operator is given as

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} + ...$$

In the 1D case, this second derivative was calculated using the second order central finite difference method in matrix form, as outlined in Section 3.1.3. "Numerical Method for Engineers 6^{th} Edition"[5] provided the basis of the implementation of this 1D second order central finite difference method, the principle for higher dimensions can be easily generalised from the methods described in the book.

$$\frac{\partial^2}{\partial x^2} f(x,y,z) = \frac{f(x-h,y,z) - 2f(x,y,z) + f(x+h,y,z)}{h^2}$$
$$\frac{\partial^2}{\partial y^2} f(x,y,z) = \frac{f(x,y-h,z) - 2f(x,y,z) + f(x,y+h,z)}{h^2}$$
$$\frac{\partial^2}{\partial z^2} f(x,y,z) = \frac{f(x,y,z-h) - 2f(x,y,z) + f(x,y,z+h)}{h^2}$$

(in the 3D case)

These operations were represented using matrices, with the method of their generation outlined in the Appendix. Using these matrices, the partial second derivatives could be represented as:

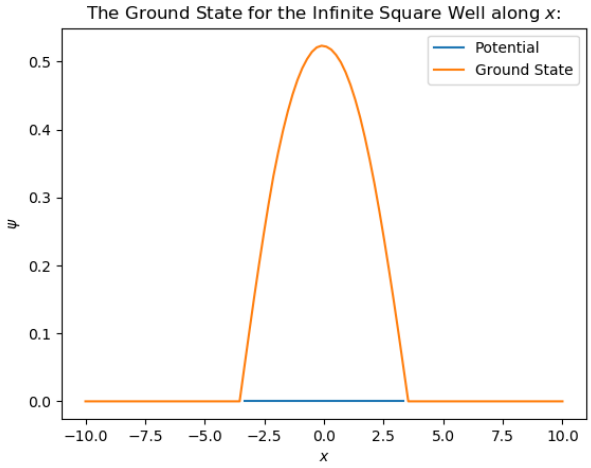
$$\frac{\partial^2}{\partial x^2} = D_x, \quad \frac{\partial^2}{\partial y^2} = D_y, \quad \frac{\partial^2}{\partial z^2} = D_z, \quad etc...$$

So that the Laplacian operator could be simply given as,

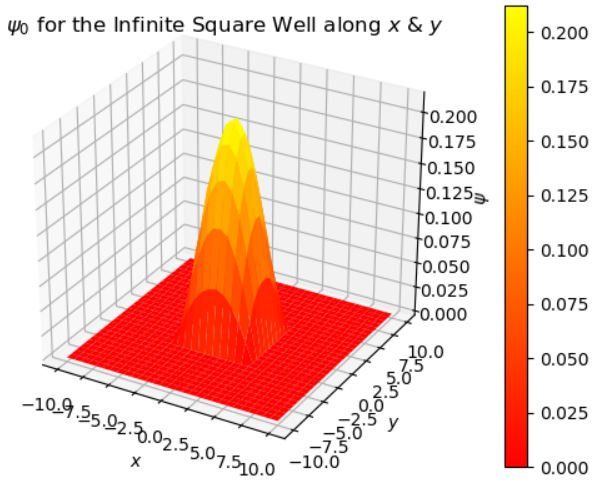
$$\nabla^2 = D = D_x + D_y + D_z$$

by using the distributivity principle of matrix multiplication. These second derivative matrices are diagonalised matrices, as outlined in the Appendix, they were implemented as scipy sparse diagonal matrices[11], these sparse matrix objects are optimised for matrix multiplication as they don't compute the products at the entries with value = 0. The implementation of these SciPy sparse matrix objects had the benefit of giving the 1D case a threefold decrease in computation time for the same potential systems.

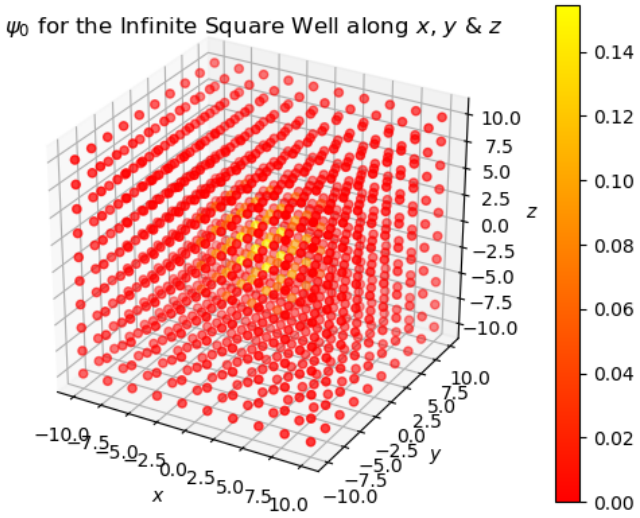
Once these changes for calculating the new higher dimension energy distributions of ψ were implemented, the final approximation for the column vector distribution of $|\psi_i\rangle$ was reshaped back to it's original n x n x ... distribution shape for the purpose of plotting the computed distribution.



(a) 1D line plot



(b) 2D surface plot



(c) 3D scatter plot

Figure 3: The Infinite Square Well ground states plotted in 1, 2, and 3 dimensions.

3.4 Plotting

The logging and display of the distributions and energies calculated for the eigenstates of the system were a core component of this project. While the calculated energies of the distributions could easily be logged to a terminal or output to a file, the Matplotlib library [9] was leveraged to provide the functionality required for plotting the calculated distributions. Plotting methods were implemented in 1D, 2D, and 3D, although the program could in principle solve for higher dimension energy eigenstates, plotting such a distribution in a geometrically feasible way was outside the scope of this project. The *meshgrid()* method from NumPy was used in extending the code to higher dimensions to allow for mapping of the dimension's axes onto each other to generate $\bar{r}$, this implementation of *meshgrid()* also allowed for the easy implementation of plotting for these higher dimensions.

3.4.1 1D

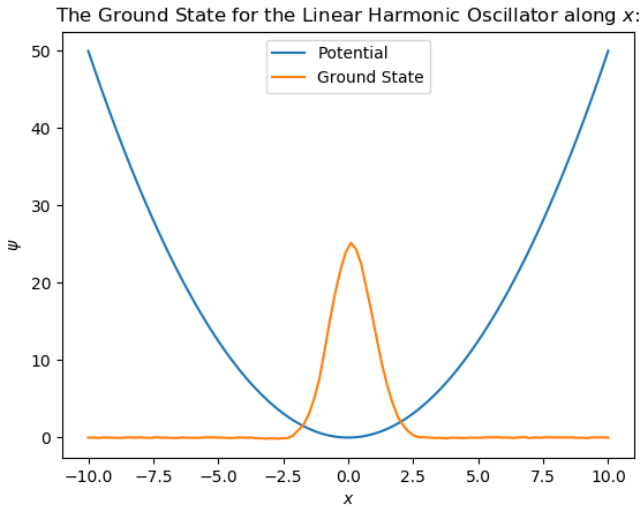
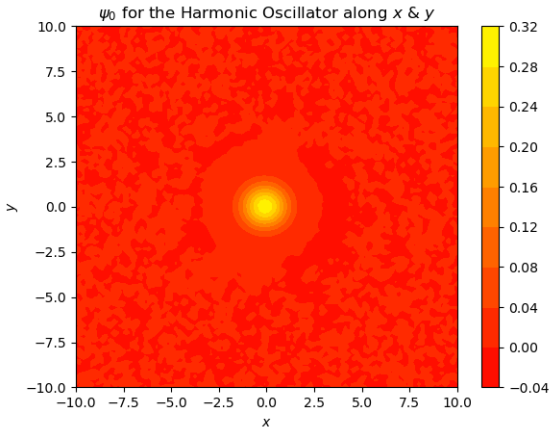
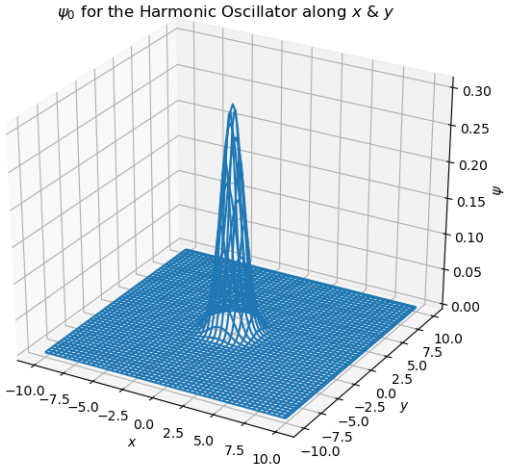


Figure 4: 1D plot of $V(x)$ and $\psi(x)$ scaled by a factor of 30, for the LHO.

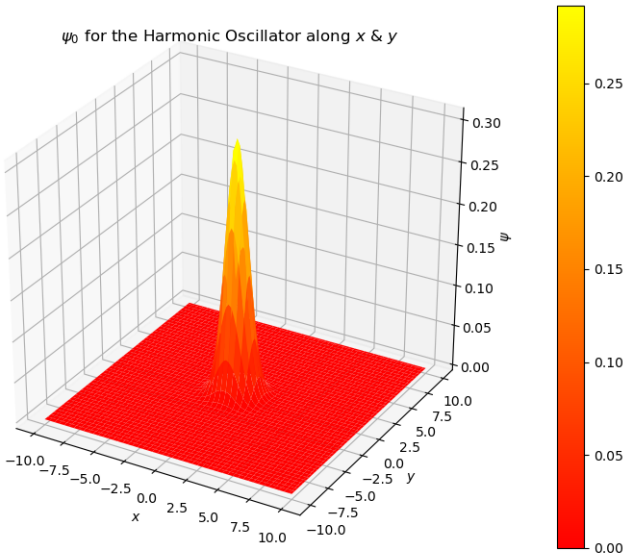
Plotting the ψ distributions in 1D was the easiest form of plotting to implement. The PyPlot sub-package in the Matplotlib library [9] was used extensively for both 1D and higher dimensional plotting. 1D plotting was implemented with the functionality of plotting the potential function and the resultant ψ distribution together on the same plot, along with the ability to scale the ψ distribution in these plots so that it's details could be distinguished against the potential, as can be seen in Figure 4.



(a) Contour plot



(b) Wire frame plot



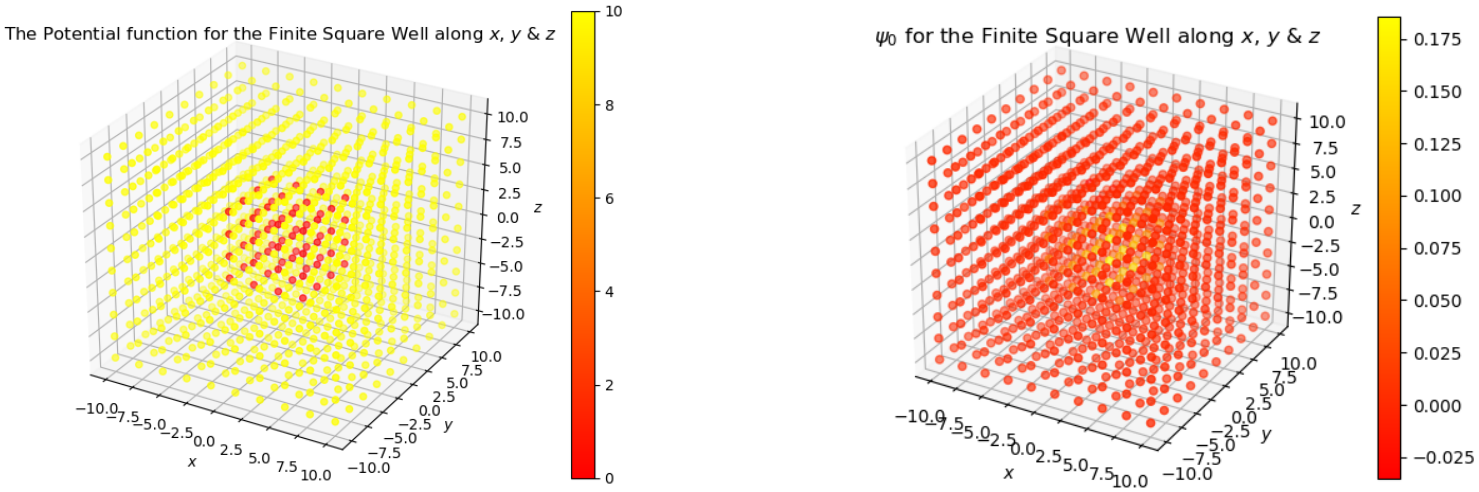
(c) Surface plot

Figure 5: Examples of the three different plot types for the ground state of the 2D Linear Harmonic Oscillator.

3.4.2 2D

Three different types of 2D plots were implemented in order to provide different insights and graphics for the calculated distributions. Examples of these plots are shown in Figure 5 for the ground state of the 2D Linear Harmonic Oscillator. Figure 5a shows an example of a filled contour plot, using the "autumn" colour map provided by Matplotlib. Figure 5b shows an example of a wire frame plot of the same ψ

distribution, while Figure 5c shows a surface plot, which is conceptually similar to a wire frame plot, apart from that it uses a colour map to more readily display contours, features and values of the plot.



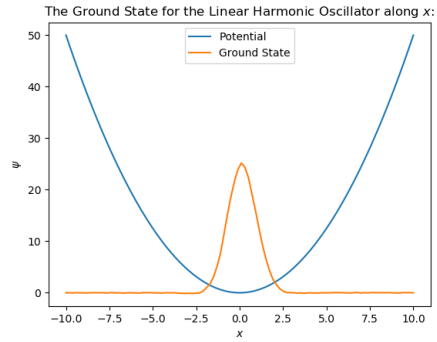
(a) The potential of the system.

(b) The ground state.

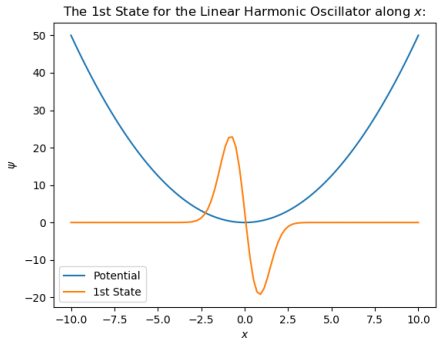
Figure 6: Examples of 3D scatter plots for the Finite Square Well in 3D.

3.4.3 3D

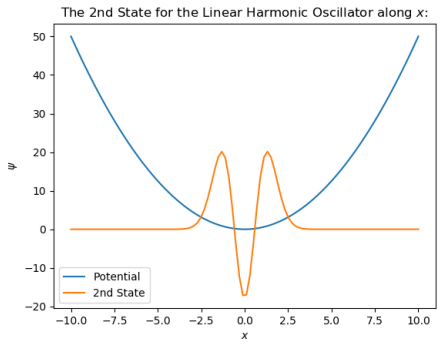
Figure 6 shows an example of the 3D scatter plot for the Finite Square Well potential and ground state in three dimensions. The 3D scatter plot represents the computed distribution as a series of points in the box, colour coded according to the provided colour map to highlight the point's relative values.



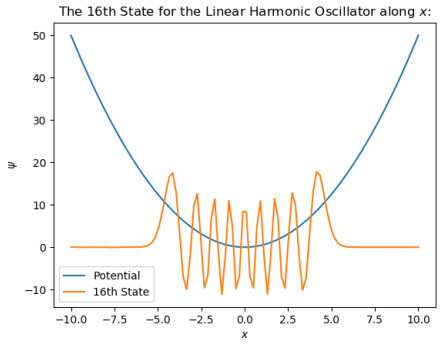
(a) The ground state.



(b) The 1^{st} energy eigenstate



(c) The 2^{nd} energy eigenstate



(d) The 16^{th} energy eigenstate

Figure 7: Computed energy eigenstates for the Linear Harmonic Oscillator in 1D.

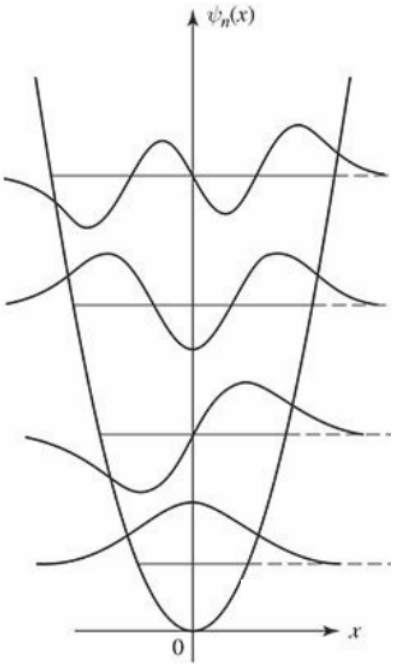


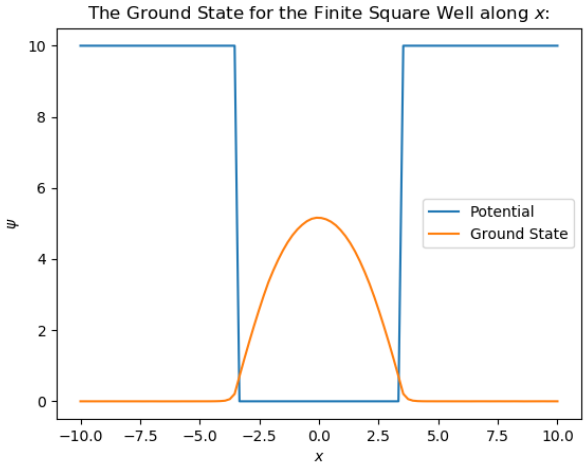
Figure 8: The first four energy eigenstates for the Linear Harmonic Oscillator, adapted from Griffiths Fig 2.7 (a) p. 72 [1].

4 Results & Discussion

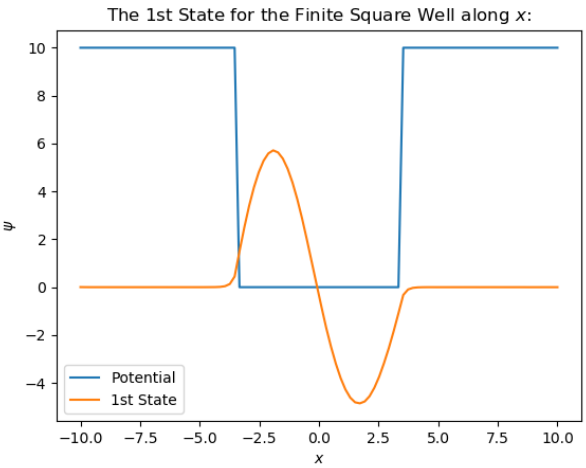
David J. Griffiths' *"Introduction to Quantum Mechanics 3^{rd} Edition"*[1] is referenced extensively throughout this discussion and was used during the development of this project to compare and verify the validity of the computed results for known potential systems. For the calculation of the energy eigenstates in all potential systems in this project, the particle mass was taken to be that of the electron, 9.1093819x$10^{-31}kg$, and $\hbar$, defined as $\frac{h}{2\pi}$ where h is Planck's constant, was taken in electron volts as 6.5821189x$10^{-16}eV$[12].

The Linear Harmonic Oscillator (LHO) system was used throughout the development of this project as a trial system to compare the computed distributions against the known analytic ones. The convention for numbering the states of the LHO is to number the ground state as $n = 0$, as such this convention has been adopted for the general numbering order sequence of states for any distribution, even those that are conventionally sequenced from $n = 1$, as the numerical solver makes no distinction between these types of systems.

The first three energy eigenstates for the one dimensional LHO are shown in Figure 7, these computed distributions match the analytical solutions, shown in Figure 8, up to an arbitrary phase constant, an example of which can be seen for subfigure 7b. The result of the quantum harmonic oscillator distribution tending towards to classical distribution, which is shown by Griffiths Figure 2.7 (b) p. 72 [1] can also be observed for subfigure 7d. Higher order states would show the trend more clearly, but these high order states were not well computed by the simulation run shown, due to the level of discretisation of the grid employed. The discretisation limit can be seen to be saturated in subfigure 7d, as the nodes the plot become sharper and less well defined.



(a) The ground state.



(b) The 1^{st} state.

Figure 9: The first two energy eigenstates for the Finite Square Well in 1D, $V_0 = 10eV$

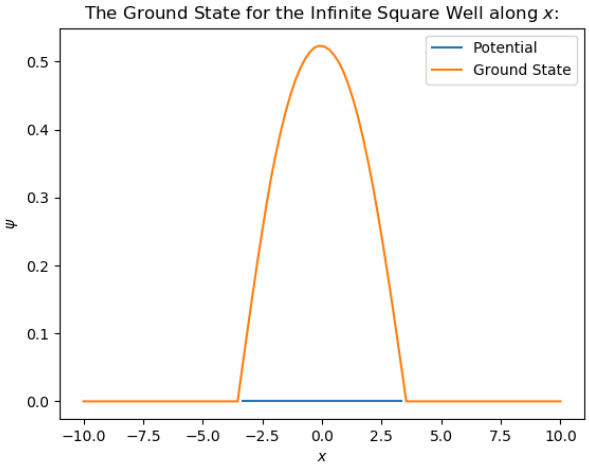


Figure 10: The ground state of the Infinite Square Well in 1D.

The results obtained for both the Finite Square Well and the Infinite Square Well match those predicted analytically. Figure 9 shows the ground state and 1^{st} state for the Finite Square Well, with $V_0 = 10eV$, the plotted distributions have been scaled up by a factor of 10 to allow comparison of the resultant distributions with the potential of the system. Subfigure 9a and subfigure 9b show the quantum tunnelling phenomenon into the potential barrier that is predicted analytically, the observed distributions also match those predicted analytically up to an arbitrary phase constant, as seen previously for the LHO. Figure 10 shows the ground state distribution for the Infinite Square Well system, the regions of infinite potential are not shown in the plot, due to limitations in Matplotlib that prevent the display of infinite values. When comparing the results obtained for the Finite Square Well, to that for the Infinite Square Well, the expected sinusoidal profile can be seen for both distributions in Figure 10 and subfigure 9a, while the ground state of the Infinite Square Well goes to zero exactly at the boundaries, and doesn't display any tunnelling effects unlike the Finite Square Well, as the particle cannot exist in regions of infinite potential.

The computed energies also match the expected analytical trends for the known system. Figure 11 shows the trend in energy eigenvalue for the first 10 energy eigenstates of the Finite Square Well, the

Table 1: Computed energy eigenvalues for the Finite Square Well in 1D.

State Number:	Energy (eV):
0	0.0459
1	0.1841
2	0.4180
3	0.7229
4	1.1298
5	1.5946
6	2.1587
7	2.7861
8	3.4933
9	4.2371

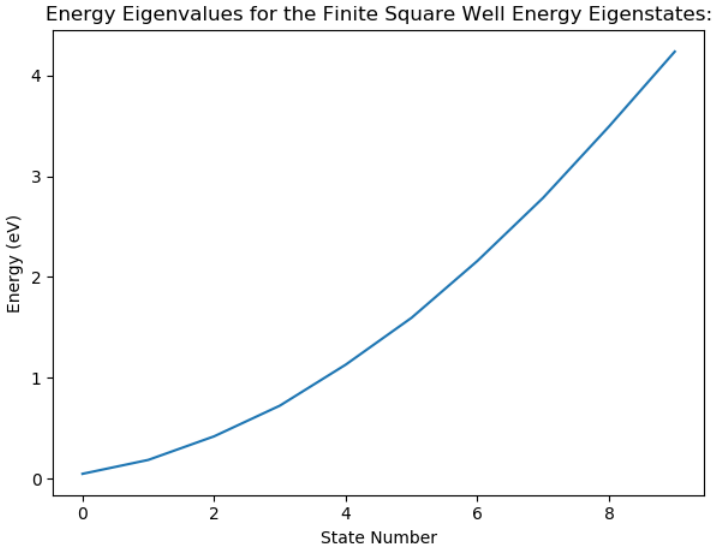


Figure 11: Plot of the computed energy eigenvalues of the Finite Square Well in 1D.

Table 2: Computed energy eigevalues of the Linear Harmonic Oscillator in 1D.

State Number:	Energy (eV):
0	0.3591
1	1.0282
2	1.7248
3	2.4053
4	3.0874
5	3.7729
6	4.3955
7	5.0734
8	5.7235
9	6.3717

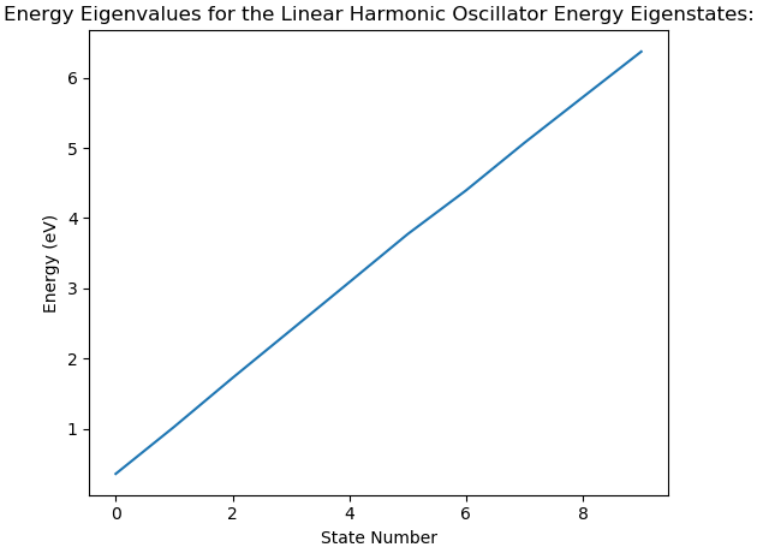
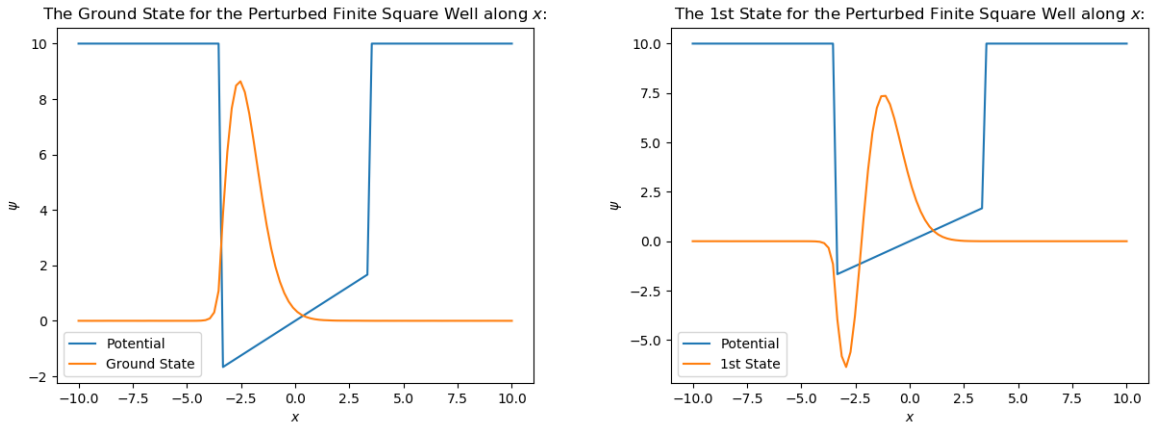


Figure 12: Plot of the computed energy eigenvalues of the Linear Harmonic Oscillator in 1D.

plot shows the general αn^2 shape of the trend, where α is some real constant. Analytically it is derived that the energy levels for the Finite Square Well follow the trend of $E_n \approx \frac{\pi^2\hbar^2}{2mL^2}n^2$[1, p. 95] where L is the width of the well, so that in this case, the constant α is given by $\alpha \approx \frac{\pi^2\hbar^2}{2mL^2}$. The computed energy eigenvalues for the first 10 states of the Finite Square Well are shown in Table 1, rounded to 4 significant figures. A similar plot is shown in Figure 12 for the first 10 energy eigenvalues of the Linear Harmonic Oscillator, the trend computed also matches the analytically predicted one, where the energies are linearly spaced, with a ground state energy of $0.3591eV$.

The energy eigenvalues for the first 10 states are shown in Table 2, the theoretical prediction for the Linear Harmonic Oscillator is that energy levels are separated by a $\hbar\omega$ energy step, where $\omega = \sqrt{\frac{k}{m}}$ [1, p. 57], for the calculation of these states, k was taken as 1, and $\hbar$, m were as stated previously. Using these values the $\hbar\omega = 0.6896eV$, the ground state energy is predicted to be $\frac{1}{2}\hbar\omega = 0.3448eV$, a deviation of $\sim 4.13\%$ from the computed value. The average spacing between computed energy eigenvalues for the first 5 states is $\hbar\omega \approx 0.6821eV$, a deviation of $\sim 1.09\%$ from the theoretically predicted value, if this average is extended to include higher order states, the result becomes less accurate, giving a value of $\hbar\omega \approx 0.6681eV$ for the first 10 states, this is a deviation of $\sim 3.12\%$ from the expected value. This trend of increasing error with higher order states highlights one of the main limitations to the variational principle. The variational principle gives an upper bound estimate of the true energy eigenstate of the system, each subsequent higher order state is computed to be orthogonal to the previous approximations, so each subsequent state is sequentially less accurate due to compounding error. The approximations of the energy eigenstates can be improved by increasing the number of points in the grid, and pairing that with an increase in the number of iterations to calculate over.

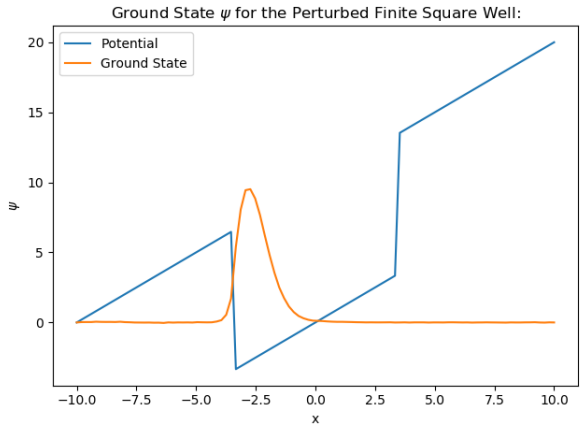


(a) The ground state.

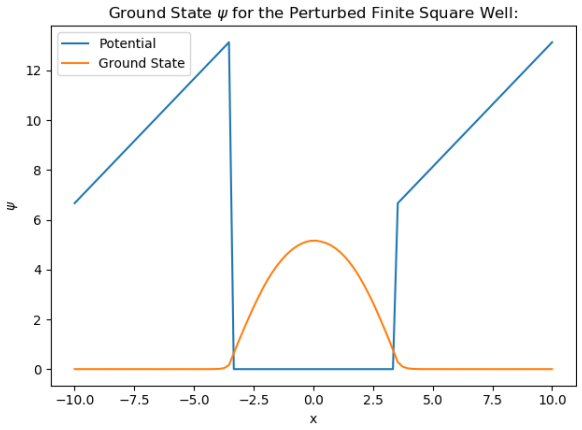
(b) The 1^{st} state.

Figure 13: The first two states of the Perturbed Finite Square Well in 1D.

Griffiths doesn't go into a treatment of these example systems for higher dimensions, so that no direct comparison between the computed results and given analytical systems can be made. However some known properties of the wavefunctions can be generalised to higher dimensions, and can be seen to occur in the higher dimension computed distributions. For example, Figure 3 shows the ground state of the Infinite Square Well in 1, 2, and 3 dimensions, these plots all display the property of the wavefunction going to zero at the boundaries of the well, especially noticeable in the surface plot subfigure 3b. If a comparison is made between subfigure 3c and subfigure 6b, the ground states for the 3-dimensional Infinite Square Well and Finite Square Well respectively, the Finite Square Well ground state is observed to be more diffuse than that of the Infinite Square Well ground state for potential boxes of the same dimension, this indicates the occurrence of quantum tunnelling as seen in the 1D case for the Finite



(a) The ground state for a variation of the Perturbed Finite Square Well in 1D.



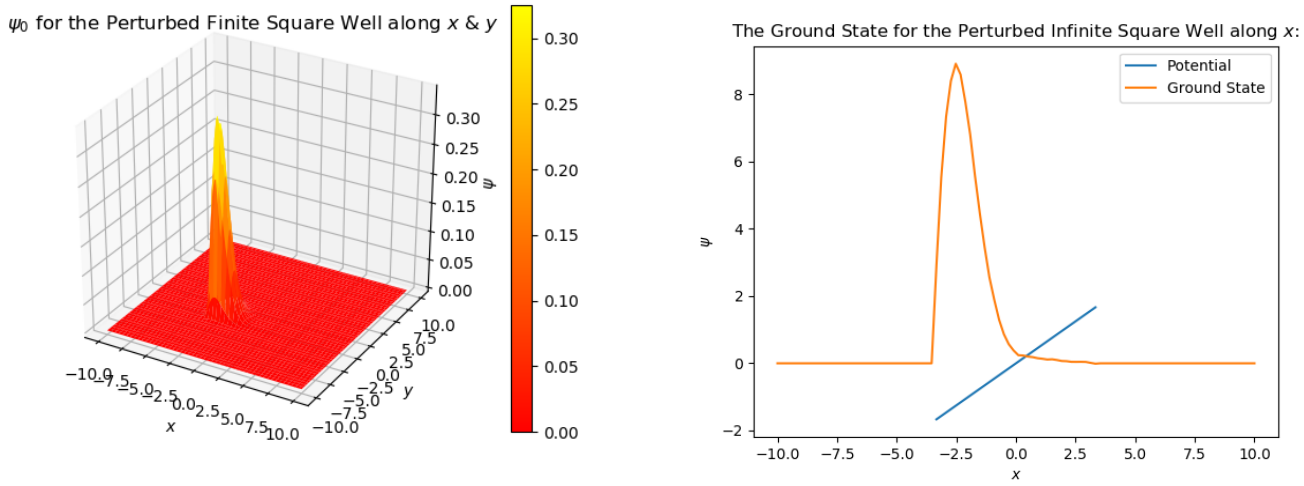
(b) The ground state for a variation of the Perturbed Finite Square Well in 1D.

Figure 14: Computed Ground states for variations of the 1D perturbed Finite Square Well.

Square Well.

The true advantage of the numerical solver can be seen in it's application to systems that have no true analytical solution, like that of the Perturbed Finite Square Well system shown in Figure 13. The system shown is simply the finite square well system with $V_0 = 10eV$ as before, with a perturbation of $\frac{1}{2}x$ applied across the well, the solutions to such a system can only be approximated analytically through the application of perturbation theory [1, p. 356]. The numerical solver handles this system the same as any other, taking approximately 9 seconds to calculate each plot shown in Figure 13 for a grid of 100 points, and 10^5 calculations per state, and producing an upper bound approximate for the true energy eigenstates, not truncated by the order of approximation as the results are through the application of perturbation theory [1, p. 356]. The program was applied to solve for the ground states of an array of perturbed systems in one and two dimensions shown in Figure 14 and Figure 15, these systems include variations of the Perturbed Finite Square Well, which behave similar to the original perturbation in the case of subfigure 14a, or similar to the unperturbed Finite Square Well in the case of subfigure 14b. The Perturbed Finite Square Well was also modelled in two dimensions, producing the graph shown in subfigure 15a, where the peak of the ground state distribution occupies the lowest potential region, as it does in the 1D case. Subfigure 15b shows the Infinite Square Well system modelled with a perturbation across the well, the infinite values of the barrier are not plotted due to a shortcoming in the Matplotlib library as described earlier. The distribution computed shows that the peak probability of finding the particle is in the region of lowest potential, while the particle cannot exist inside the regions of infinite

potential as is the case in the unperturbed Infinite Square Well system.



(a) Ground state of the Perturbed Finite Square Well in 2D.

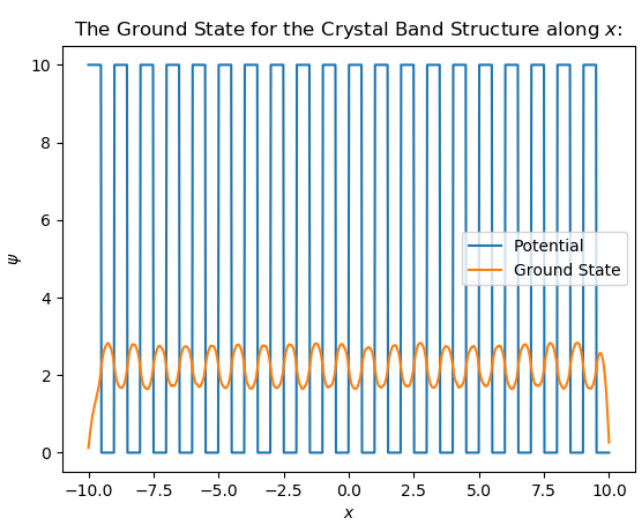
(b) The ground state of the Perturbed Infinite Square Well in 1D.

Figure 15: Computed ground states for the Perturbed Finite Square Well in 2D, subfigure 15a, and the perturbed Infinite Square Well, subfigure 15b.

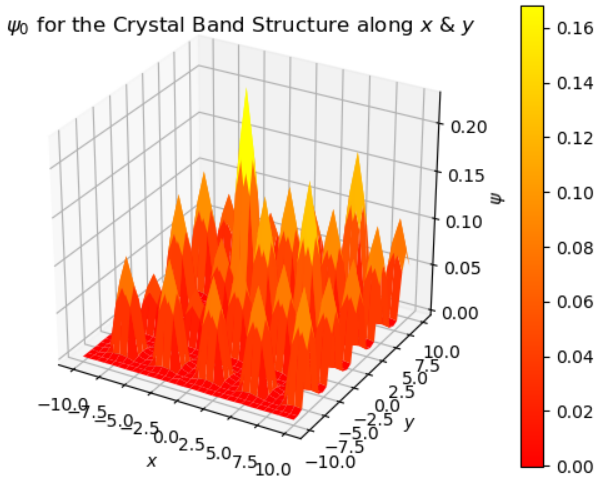
The program was also applied to solve for the ground state for a crystal band structure, the results are shown for 1D and 2D in Figure 16. The crystal band structure was approximated as a series of finite square wells where each atom would be in the lattice, and the particle mass was taken as that of the electron as before. The two computed distributions display a sinusoidal characteristic similar to that of the travelling wave solution, the deviation between these solutions occurs at the boundaries. Whereas the travelling wave solution is non-normalisable without imposing a wave packet solution, resulting in a continuous but normalisable solution, the numerical solver is applied with the assumption that all states to be solved for are normalisable. As such, the computed states drop to zero at the boundaries as a normalisable solution will tend to zero at infinity. Therefore, discounting the boundaries, the numerical solver is able to approximate the travelling wave solution of the crystal lattice under appropriate approximations and assumptions.

Degeneracy of energy levels is expected to occur for higher dimensional states. This result is observed for the 2D Linear Harmonic Oscillator energy eigenstates computed, the first three states are shown in Figure 17. In this case, degeneracy is observed for the 1^{st} and 2^{nd} energy eigenstates, in subfigures 17b & 17c, these states were computed to have an approximately equal energy eigenvalue up to an unknown error value. The values for the energies are shown in Table 3, where it can be seen that the 1^{st} and 2^{nd} state's energies differ by $\sim 1.14\%$. Though a not insignificant deviation, this result it quoted with the caveat of no supporting error estimate, however the result obtained still matches that expected analytically, and still stands to validate the computational results obtained. These results were computed using a 100x100 sized grid, upon which 10^7 iterations were performed, using a moderately performant computer with an Intel i7 processor, these computations took approximately one hour each, giving a three hour total computation time, highlighting future scope for performance improvements of the code, especially for higher dimension computations.

The computational complexity of the code had two main drawbacks. The first limitation was in the usage of the *nullspace()* method provided by the linalg package from scipy [11]. The nullspace method takes as input an mxn matrix as described in section 3.2.2, the method then returns an nxn matrix whose



(a) The ground state for the approximate crystal band structure in 1D.



(b) The ground state for the approximate Crystal Band Structure in 2D.

Figure 16: The ground states for the Crystal Band structure approximated as a series of Finite Square Wells in 1D and 2D.

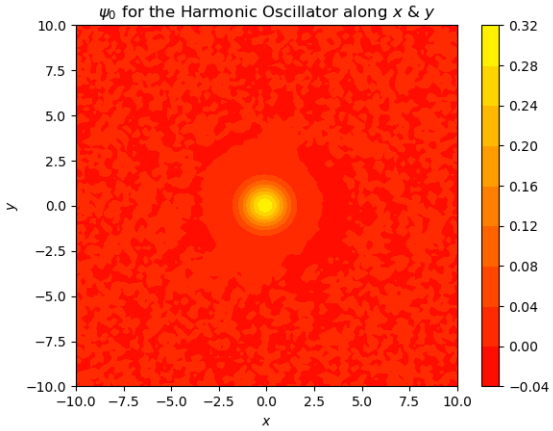
columns are used for the orthonormal basis. The number of entries in the ψ_i distributions are N^D where N is the level of discretisation along the line, and D is the number of dimensions. Thus the input matrix A is given by $m\text{x}N^D$ where m is the number of previous states computed, and the returned basis matrix is $N^D\text{x}N^D$, the size of this matrix grows as N^{2D}, so that the upper limit of memory of the computer used is quickly reached for large grids in higher dimensions. The computational complexity for this operation is $\mathcal{O}(N^{2D})$, as would be expected, so that these larger grids also take longer to compute. Paired with these larger grids, the number of iterations required to converge to the approximate energy eigenstates is also increased, where each entry in the grid requires approximately 100 samplings to reach an upper bound on the acceptable accuracy. Therefore the average number of iterations performed for $N = 100$ was of the order 10^{D+4}, the computational complexity of the sampling was simply $\mathcal{O}(n)$ for n iterations, but the larger grids requiring more iterations quickly slowed down the computation of higher dimension states for large grids.

The code and methods of this project could be further improved by the addition of some more generalisations that were outside the scope of this project. All plots produced from the project are of the $\psi(\bar{r})$ energy eigenstate, these distributions are the ones generally shown in textbook material, though they do not represent the actual probabilistic spatial distribution of the particle. The project could be easily extended to plot the $|\psi(\bar{r})|^2$ distributions if it were to be applied to any real world problems. The code could be further extended to include the generation of approximate time evolution of the computed energy eigenstates. The computed states are the spatial component $\psi(\bar{r})$ of the wavefunction $\Psi(\bar{r}, t)$ which is assumed to be separable during the derivation of the TISE. The actual wavefunction Ψ can be easily computed as

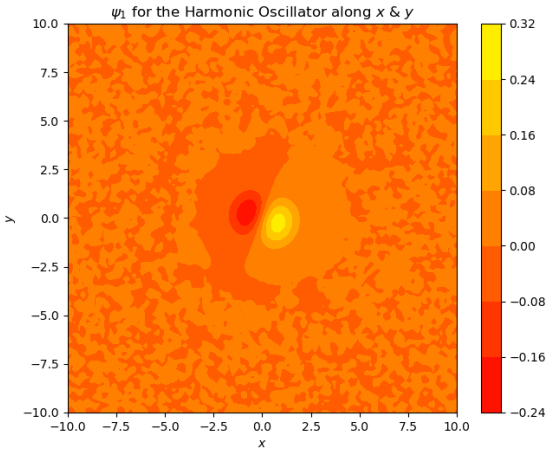
$$\Psi(\bar{r}, t) = \psi(\bar{r})e^{-i\frac{\langle E\rangle}{\hbar}t}$$

using the computed energy eigenvalue for ψ, $\langle E\rangle$. These time-dependent wavefunctions could be computed to further illustrate the behaviour of the particles in the given potential systems.

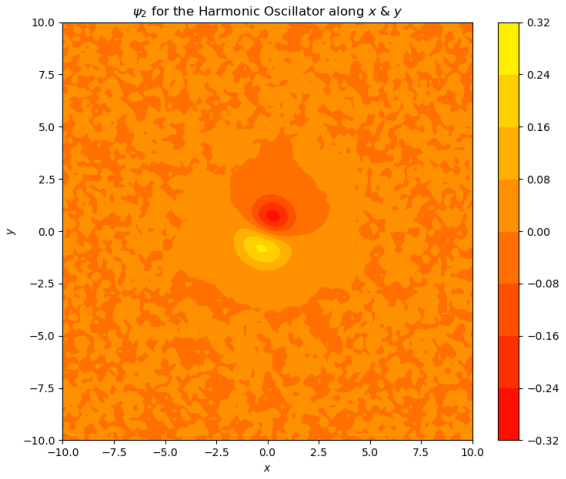
Some other further improvements that could be implemented in the project would be the development and improvement of the numerical methods utilised in the code. Both the normalisation and the



(a) The ground state



(b) The first degenerate state.



(c) The second degenerate state.

Figure 17: Plot of the first three energy eigenstates of the 2D LHO, highlighting degeneracy of states in subfigures 17b & 17c, the energy eigenvalues are shown in Table 3.

Table 3: Energy eigenvalues of the first 3 states of the 2D LHO.

State Number:	Energy Eigenvalue (eV)
0	0.6986
1	1.3854
2	1.4014

expectation value of the energy are approximated using right hand Riemann sum integration approximations. The accuracy of these numerical integration methods could be improved by using higher order methods, for example the Romberg integration method [3, p. 434], or for higher dimension integrations a Monte Carlo method. Numerical differentiation could be improved by using higher order Taylor series approximations to derive the partial derivative matrices described in the Appendix. Error approximations could also be introduced to give a handle of the error of the computed energy eigenvalues and the energy eigenstates. For example, in the 2D LHO, Figure 17, a baseline noise of ± 0.04 is observed for the computed energy eigenstates, though no estimate of this error is given numerically. The error of these states is easily visualised in the contour plots produced for the 2D state, like those in Figure 17 however it would be present to some degree in all plots. This error, that has the appearance of noise in the results, could possibly be filtered out through the application of signal analysis and Fourier analysis techniques.

5 Conclusion

The variational principle in Quantum Mechanics was applied numerically in python to solve for energy eigenstates and energy eigenvalues of given bound potential systems. The method was implemented to be able to solve for the first m number of states of the given bound system, that could span D number of dimensions. Once the code was fully implemented and capable of solving for potential systems in reasonable time, the computed results were compared to known analytical states from Griffiths *"Introduction to Quantum Mechanics* 3^{rd} *Edition"*[1]. The computed energy eigenstates exactly match the known analytical results in 1D, and produce the expected degeneracy of energy states in higher dimensions. Using the developed code, the energy eigenstates of perturbed and complex systems were able to be easily computed, that would not be easy to compute analytically.

6 References

[1] David J. Griffiths, *"Introduction to Quantum Mechanics* 3^{rd} *Edition"*, Cambridge University Press, 2018

[2] Harvey Gould, Jan Tobochnik, and Wolfgang Christian, *"An Introduction to Computer Simulation Methods - Applications to Physical Systems"*, Open Source Physics, p. 687, 2016.

[3] Philip J. Davis and Philip Rabinowitz, *"Methods of Numerical Integration* 2^{nd} *Edition"*, Academic Press Inc., p. 9, 1984.

[4] Gene H. Golub, Charles F. Van Loan, *"Matrix Computations* 4^{th} *Edition"*, John Hopkins University Press, p. 64, 2013.

[5] Steven C. Chapra, *"Numerical Methods for Engineers* 6^{th} *Edition"*, McGraw-Hill, p. 655, 2010.

[6] Pauli Virtanen, et al., *"Fundamental Algorithms for Scientific Computing in Python"*, Nature Methods, 2020.

[7] Travis E. Oliphant, *"A guide to NumPy"*, Trelgol Publishing, 2006.

[8] Stéfan van der Walt, S. Chris Colbert and Gaël Varoquaux, *"The NumPy Array: A Structure for Efficient Numerical Computation"*, Computing in Science & Engineering, 13, 22-30 (2011), DOI:10.1109/MCSE.2011.37.

[9] J. D. Hunter, *"Matplotlib: A 2D Graphics Environment"*, Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007

[10] Carl Nordling and Jonny Österman, *"Physics Handbook for Science and Engineering"*, Studentlitteratur, pp. 12-14, 2006.

[11] scipy.sparse.diags. Retrieved from https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.diags.html.

[12] scipy.linalg.null_space. Retrieved from https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.null_space.html.

7 Appendix

7.1 N-Dimensional Second Order Central Finite Difference Method

In order to calculate the Laplacian Operator on the distribution ψ, the operator had to be extended to be able to handle the second derivatives along the same number of dimensions as the dimensions of ψ. The n-dimensional ψ was reshaped to an equivalent column vector in order to be able to retain the implementation of linear algebra notation, thus the Laplacian operator had to be extended to a generalised second derivative matrix.

For the 3D case, where $f(x, y, z)$ the partial second order central difference methods are given as:

$$\frac{\partial^2}{\partial x^2} f(x,y,z) = \frac{f(x-h,y,z) - 2f(x,y,z) + f(x+h,y,z)}{h^2}$$

$$\frac{\partial^2}{\partial y^2} f(x,y,z) = \frac{f(x,y-h,z) - 2f(x,y,z) + f(x,y+h,z)}{h^2}$$

$$\frac{\partial^2}{\partial z^2} f(x,y,z) = \frac{f(x,y,z-h) - 2f(x,y,z) + f(x,y,z+h)}{h^2}$$

In order to be able to implement the linear algebra methodology, the composition of these higher order second derivative matrices had to be determined. Shown below is one of the simplest higher order examples of the partial second derivatives of a 3x3 matrix f.

$$F = \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix} \rightarrow \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix}$$

$$\frac{\partial^2}{\partial x^2} f = D_x F = \frac{1}{h^2} \begin{pmatrix} -2 & 1 & & & & & & & & \\ 1 & -2 & 1 & & & & & & & \\ & 1 & -2 & 0 & & & & & & \\ & & 0 & -2 & 1 & & & & & \\ & & & 1 & -2 & 1 & & & & \\ & & & & 1 & -2 & 0 & & & \\ & & & & & 0 & -2 & 1 & & \\ & & & & & & 1 & -2 & 1 & \\ & & & & & & & 1 & -2 \end{pmatrix} \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix}$$

$$\frac{\partial^2}{\partial y^2} f = D_y F = \frac{1}{h^2} \begin{pmatrix} -2 & 0 & 0 & 1 & & & & & & \\ 0 & -2 & & & 1 & & & & & \\ 0 & & -2 & & & 1 & & & & \\ 1 & & & -2 & & & 1 & & & \\ & 1 & & & -2 & & & 1 & & \\ & & 1 & & & -2 & & & 1 & \\ & & & 1 & & & -2 & & 0 & \\ & & & & 1 & & & -2 & 0 & \\ & & & & & 1 & 0 & 0 & -2 \end{pmatrix} \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix}$$

All blank entries in the matrices correspond to 0. It can be clearly seen that these matrices are sparse diagonal matrices. The sparsity gets worse as the number of points in the model is increased, along with the number of dimensions, the size of these matrices grows as N^DxN^D where N is the number of samples along a line in the problem, and D is the number of dimensions. The usage of the sparse matrix object from SciPy [11], allows these large high dimension matrices to be generated without storing needless information in the form of 0s.

The general diagonal stencil was determined for higher dimension states, where N & D are as defined earlier, and *axis_number* is an index from 0 to D - 1, which corresponds to x, y, z, etc...:

```
import scipy.sparse as spr

# axis_number is the axis index to use:
# x = 0, y = 1, z = 2, etc...

num_cells = D - (axis_number + 1)
# The general pattern for a derivative matrix along the axis:
# axis_number, for a num_axes number of
# dimensions, each of length N
diagonals = [
[-2] * N**D,
(([1] * N**axis_number)*(N - 1) + [0]*N**axis_number) * N**num_cells,
(([1] * N**axis_number)*(N - 1) + [0]*N**axis_number) * N**num_cells
]

#Create a sparse matrix for the given diagonals, of the desired size.
D_n = spr.diags(diagonals, [0, -N**axis_number, N**axis_number],
shape=(N**D, N**D))
```

Using these generalised partial second derivative matrices, once scaled by a factor $\frac{1}{h^2}$, the total

Laplacian operator ∇^2 was simply the sum of these individual matrices.

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} + ... \rightarrow D = D_x + D_y + D_z + ...$$