Solving the Time Independent Schrödinger Equation on a Lattice

Max Heimann 580560, Lyding Brumm 604522, Burak Varol 623941

November 2023

Contents

1 Introduction

Solving the time independent Schroedinger equation (TISE) is only possible in an analytical fashion for a subclass of trivial problems. This explains the outstanding importance of analytical solvable potentials like e.g. the quantum harmonic oscillator, but wherever there is no suitable model potential a solution can be still obtained by numerical calculations. We present an algorithm to obtain the ground-state and its energy by approximating the continuous case by a discrete lattice. We prove some useful properties of the discretised Hamiltonian and the use algorithm to solve the 'Mexican hat' potential:

$$V(x) = \frac{m\omega^2}{8r^2}(x^2 - r^2)^2$$

which will be completely described in terms of natural units. The code consists of following modules:

Module Name	Summary
grid.py	Implements grid related methods
hamilton.py	Implements the hamiltonian
eigen.py	Implements the power-methode and conjugate-gradient algorithm
observables.py	Implements a set of operators
main.py	The main code and cli-based user interface
qho.py	A toy example to test nondimensionalisation
1d_calcs_and_vis.py	Visualise 1d potential and wave-function
2d_calcs_and_vis.py	Visualise 2d potential and wave-function

Table 1: Summary of python modules

2 Modules

2.1 Grid Module

This module creates:

- The grid that we want to work with. In this code every dimension is discretised in $2N+1$ points, where the points span from $-N, .., 0, ...N$, with total D dimensions. To create the grid we use the numpy function "np.meshgrid" and then stack the coordinates together with "np.stack" to create a human readable grid.
- The grid which contains the distance squared values for every value. This part is the main part that we need to calculate the potential.

2.2 Hamilton Module

This Module contains:

- The potential energy operator. The function calls "calculate grid distance square" function from grid module and applies the formula $V = \frac{\mu}{8}(\epsilon^2|n|^2 - 1)^2$. Result is an np.ndarray.
- The kinetic energy operator. The function applies the formula $-\frac{1}{2\mu\epsilon^2}\Sigma_k(\psi(n + e_k) + \psi(n - e_k) - 2\psi(n))$. Here we use the function "np.roll" which considers the values that are translated in the specified direction when called. This imposes periodic boundary condition.
- The create_hamiltonian function, which returns the Hamiltonian. We created this so that we can create the Hamiltonian operator and save it to a variable for further use.

2.3 Eigenvectors/values Module

This Module consists of:

- "get highest eigenvalue" and "apply inverted" functions from warm-up exercises, details can be found in the related project. We slightly modified them by creating "gen invert operator", where now the parameters "tolerance" and "res recal iterations" are directly passed to apply inverted; and "get smallest eigenvalue" function, where the smallest eigenvalue are calculated by calculating the highest eigenvalue of operator^{-1}

- Three test functions "test get highest eigenvalue", "test get smallest eigenvalue" and "test apply inverted." The first two creates random matrices with positive determinants and calculates the highest/lowest eigenvalue with corresponding eigenvector. If the calculated values are true, then $\langle v^T A v \rangle$ is the eigenvalue and $1 - \langle v^T A v \rangle$ is the error. Actually, this just answers the question if the calculated values are eigenvalues or not, but due to the nature of the algorithm it must be the highest/lowest eigenvalue. The last test compares the result with the multiplication of the proper inverse matrix multiplied to a test vector.

2.4 Observables

This Module consists of the functions:

- "mu 0", calculating the expectation value of an observable O
- "var 0", calculating the variance of an observable O
- "stat psi" where it calculates the upper two at the same time. If individual values are needed then it is more efficient to call one of the first functions. If both values are needed , the third function would be faster.
- "apply x 1d" Applies the location operator x along an axis to psi.
- "sphere r prob" finds the probability to find a the particle within a radius of r for a given wave-function.
- " single slice " which slices an array at a specific axis.(not used, but kept for convenience)
- "mult along axis" which multiplies a 1d array (B) along an axis of an Nd-array(A). This is helpful for the location operator

Tests for Gaussian

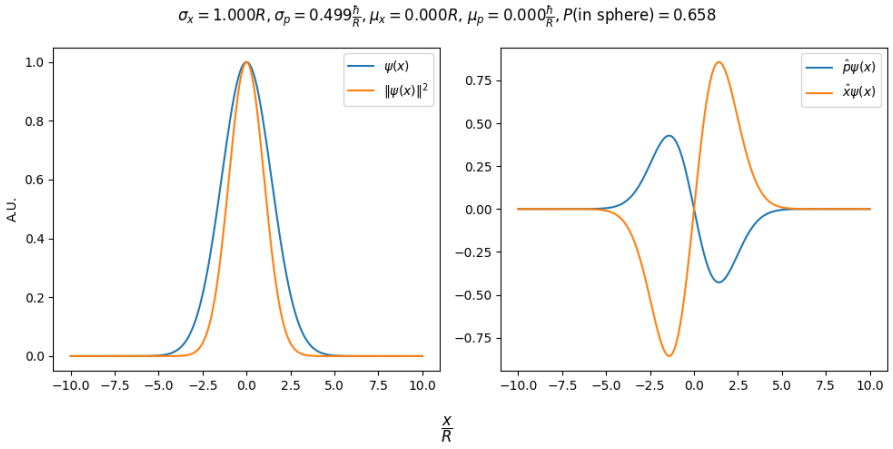

Figure 1: Observables of a Gaussian test function

To run a quick test on the observables module we run every function on a wave-function with a Gaussian PDF which has $\sigma = R$ in 1D. This way we can do a easy sanity check and see if everything is as expected, i.e. the expectation values for $\hat{x}$ and $\hat{p}$ are zero as well as $\sigma_x = R$ and $\sigma_p = \frac{\hbar}{R}$. Additionally the probability of the particle being inside the sphere of radius R (the interval $[-R, R]$) is about 65.8%. Everything works as expected.

2.5 Usage of the main code

The module "main.py" implements a cli-based user-interface for the our TISE-solver. All following simulation parameters have to be specified in an input file:

Parameter Name	Summary
eps	First simulation parameter
mu	Second simulation parameter
N	The number of discretisation points
D	The dimension of the grid
tolerance	The acceptable error for power-method and conjugate gradient
res_recal	How frequently should the residuum recalculated precisely
groundstate_output_file	Where to save the wave-functions

Table 2: Description of input parameters

The program will calculate the ground-state energy, the expectation value of the location, the moment and their standard deviations and write the back to the input file. The wave-function will be stored in a numpy binary format, so it can be processed easily in the future. Multiple input files can be specified and will be processed sequentially, implementing multiprocessing support should be fairly easy and might be added in the future.

3 System of Natural Units

Over the centuries of physical research different branches of physics prefer different systems of units. This is known to be a source of errors. Even if there are modern attempts to incorporate unit systems to programming languages it is still considered to be best practice to describe each physical system in its own units. Is the system described by a Hamiltonian i.e. a kinetic energy and a potential energy term the potential dictates a system of units. For the 'Mexican hat' potential the parameters are:

$$\mu = \frac{m\omega r^2}{\hbar}$$
$$\epsilon = \frac{a}{r}$$

Where a is the lattice constant.

4 The discretised Hamiltonian and its properties

In continuous quantum mechanics hermiticity of all operators corresponding to observables is an axiom. This is therefor especially true for the Hamiltonian of a system. After discretisation hermiticity and other properties of the Hamiltonian operator has to be proven. The scalar product on a finite dimension Hilbert space is defined as

$$(\psi, \phi) = \sum_n \psi(n)^* \phi(n)$$

which is sesquilinear in the first element by definition.
The Hamiltonian is defined as:

$$H_a\psi(n) = -\frac{1}{2\mu\epsilon^2}\sum_k [\psi(n+e_k) + \psi(n-e_k) - 2\psi(n)] + \frac{\mu}{8}(\epsilon^2 n^2 - 1)^2\psi(n)$$

and therefor can be split up into a kinetic and potential term. This allows use to prove linearity and hermiticity of both terms separately which makes the proves shorter and allows us to neglect the prefactors in some cases.

Lemma 4.1. *Let ψ, ϕ be discrete wave-functions, than $(\psi, \phi)^* = (\phi, \psi)$*

Proof.

$$\begin{aligned}(\psi,\phi) &= \sum_n \psi(n)^*\phi(n)\\ &= \left(\sum_n \psi(n)\phi(n)^*\right)^*\\ &= \left(\sum_n \phi(n)^*\psi(n)\right)^*\\ &= (\phi,\psi)^*\end{aligned}$$

□

This allows us to prove linearity in only one element.

Theorem 4.2. *Let V be the (real) potential operator. Then $(\alpha\psi_a + \beta\psi_b|V|\phi) = \alpha(\psi_a|V|\phi) + \beta(\psi_b|V|\phi)$*

Proof. The sum is linear. □

Theorem 4.3. *Let ϕ, ψ be wave-functions and V the real potential operator, then $(\psi|V\phi) = (V\psi|\phi)$*

Proof.

$$(\psi|V\phi) = \sum \psi(n)^*V(n)\phi(n) = \sum \psi(n)^*V(n)^*\phi(n) = \sum V(n)^*\psi(n)^*\phi(n) = (V\psi|\phi)$$

□

Theorem 4.4. *Let ϕ, ψ be wave-functions and T the discretised kinetic operator,then $(\alpha\psi_a + \beta\psi_b|V|\phi) = \alpha(\psi_a|V|\phi) + \beta(\psi_b|V|\phi)$*

Proof. The sum is linear. □

Theorem 4.5. *The kinetic energy operator is positive i.e. $(\phi_j|T|\phi_j) > 0$*

Proof. Let $\mathcal{B}$ be the discretised location basis made up by discretised Dirac functions, then the diagonal elements are:

$$\begin{aligned}T_{ii} &= e_i^\dagger T e_i\\ &= e_i^\dagger \frac{\hbar^2}{2m}(2e_i - e_{i-1} - e_{i+1}) = \frac{\hbar}{m}\end{aligned}$$

Where e_i are the basis functions, which are by definitions orthonormal. □

Theorem 4.6. *The kinetic energy operator is hermitian.*

Proof. Note that $\delta_{i,j+k} = \delta_{j,i-k}$ Again we use the orthonormal location basis $\mathcal{B}$. We can write:

$$\begin{aligned}T_{ij} &= e_i^\dagger T e_j = \frac{\hbar^2}{2m}(2\delta_{ij} - \delta_{i,j-1} - \delta_{i,j+1})\\ &= \frac{\hbar^2}{2m}(2\delta_{ji} - \delta_{j,i+1} - \delta_{j,i-1}) = e_j^\dagger T e_i = T_{ji}\end{aligned}$$

So the kinetic operator is symmetric. As it is purely real as well, the statement is shown. □

From $H = T + V$ and T, V being hermitian it follows, that H is hermitian as well. The eigenfunctions of the continuous kinetic energy operator are plane waves. Restricting the overall space to a box with periodic boundary conditions imposes a restriction to the k-vector of the planar wave functions as known from e.g. Solid State Physics. The same is true for our discretisation:

Theorem 4.7. *The discretised plane wave-functions $\phi_k(n) = \exp\left(\frac{2\pi i n k}{N}\right)$ are eigenfunctions of the kinetic energy operator. Where $k \in \mathbb{Z}^D$.*

Proof. First one notices that the (discretised) plane wave-functions are product functions of 1d plane wave-functions.

$$\begin{aligned}\phi_k(n) &= \Pi \exp\left(\frac{2\pi i n_l k_l}{N}\right) \\ &= \exp\left(\frac{2\pi i n_l k_l}{N}\right)\xi_l\end{aligned}$$

where ξ_l is the product over all $1d$ plane wave functions with indices not equal to l. Obviously $\left[T_l^{1D}, \xi_l\right] = 0$, where T_l^{1D} denotes the 1d kinetic energy operator in direction of l. With this one can write:

$$\begin{aligned}T\phi_k(n) &= \left(\sum_l \exp(2\pi i(n_l - e_l)k_l/N) + \sum_k \exp(2\pi i(n_l + e_l)k_l/N) - \sum_k \exp(2\pi i n_l k_l/N)\right)\xi_l \\ &= \left(\sum_l \exp(2\pi i n_l k_l/N)\exp(-2\pi i e_l k_l) + \sum_l \exp(2\pi i n_l k_l/N)\exp(+2\pi i e_l k_l) - \sum_l \exp(2\pi i n_l k_l/N)\right)\xi_l \\ &= \sum_l \exp(2\pi i n_l k_l/N)\left(\exp(-2\pi i e_l k_l) + \exp(+2\pi i e_l k_l) - 2\right)\xi_l \\ &= 2\left(\sum_l \cos(2\pi e_l k_l) - 1\right)\phi_k(n)\end{aligned}$$

□

5 Results

We ran the program for 1D and 2D for the given parameters in the script. The results are visualized in Figures 2 to 4. The observalbles for the 1D case are

$$\begin{aligned}\mu_{\vec{x}} &= -0.0780r & \mu_{\vec{p}} &= 0.0r^{-1}\hbar \\ \sigma_{\vec{x}} &= 0.970r & \sigma_{\vec{p}} &= 3.06r^{-1}\hbar \\ P(\text{in sphere}) &= 0.578\end{aligned}$$

For the 2D case the observables are

$$\begin{aligned}\mu_{\vec{x}} &= (3.64 \cdot 10^{-07}, -4.89 \cdot 10^{-04})^T r & \mu_{\vec{p}} &= (0.0, 0.0)^T r^{-1}\hbar \\ \sigma_{\vec{x}} &= (0.687, 0.687)^T r^{-1}\hbar & \sigma_{\vec{p}} &= (2.13, 2.13)^T r^{-1}\hbar \\ P(\text{in sphere}) &= 0.593\end{aligned}$$

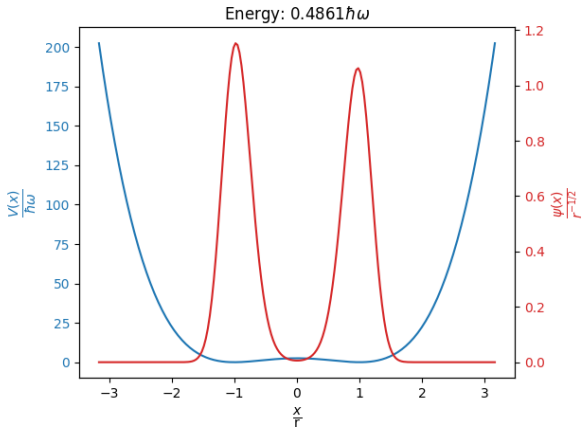

Figure 2: 1D Potential: $D = 1$, $N = 201$, $\mu = 20$, $\epsilon^2 = 0.001$

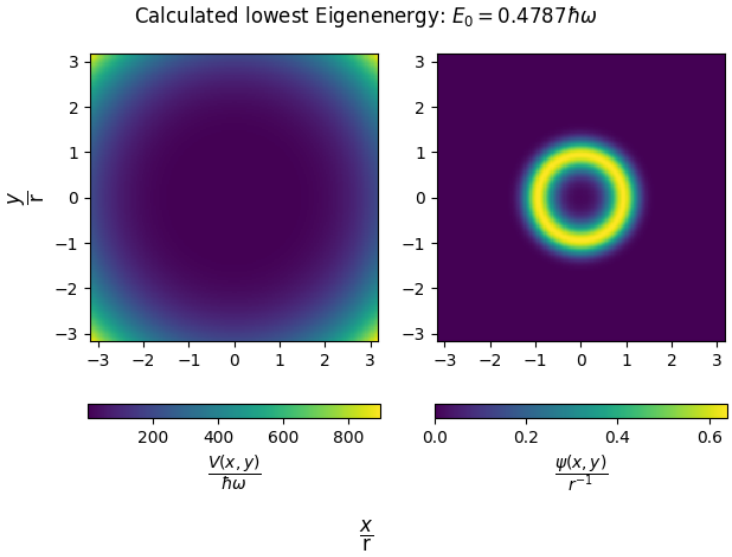

Figure 3: On the left the 2d potential, on the right the ground-state: $D = 2$, $N = 201$, $\mu = 20$ $\epsilon^2 = 0.001$

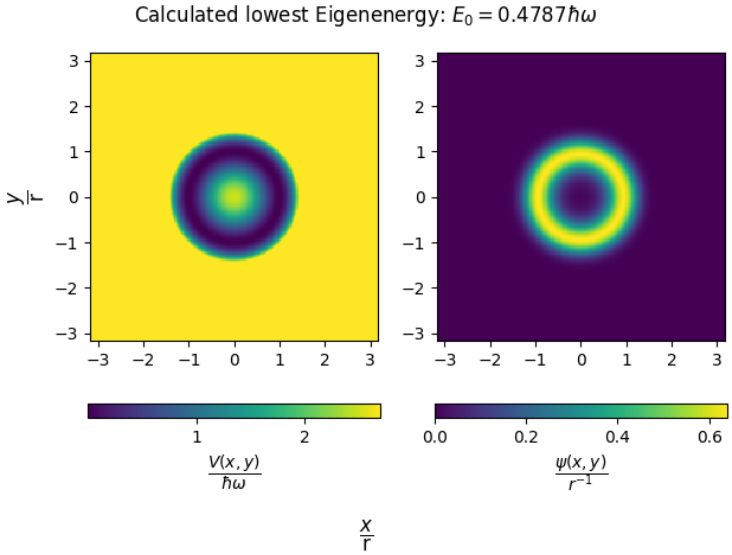

Figure 4: On the left the 2d potential, on the right the ground-state (zoomed)

If we want to have a finer grid for the same regime, we must send $\epsilon \to 0$ and $N \to \infty$ while keeping ϵN fixed. We did this for 1D case.

The continuum limit $a \to 0$, $L \to \infty$ is simulated by sending ϵ to zero while sending N to infinity. Since $L = aN$, N must go faster to infinity than a goes to 0. The results can be found in figure 6:

5.1 Remarks

5.1.1 Wave-function

- We observe two peaks of the wave function, which are localized in the local minima of the potential, as expected.
- There is an asymmetry of the wave-function which is unexpected for a symmetric Hamiltonian. We observed different heights of the peaks for different executions of the same program, which made us conclude that the asymmetry is caused by a computational error. A possible way out to minimize the error is allowing lower tolerance values in the functions that we used. (eg. for getting the highest/lowest eigenvalues). This is also visible in the expectation value in $\vec{\mathbf{x}}$. For the 2D case this error is smaller, which is not necessarily trivial to expect.
- **Continuum limit**: The wave function behaves similar to the other cases, which is a sign that everything

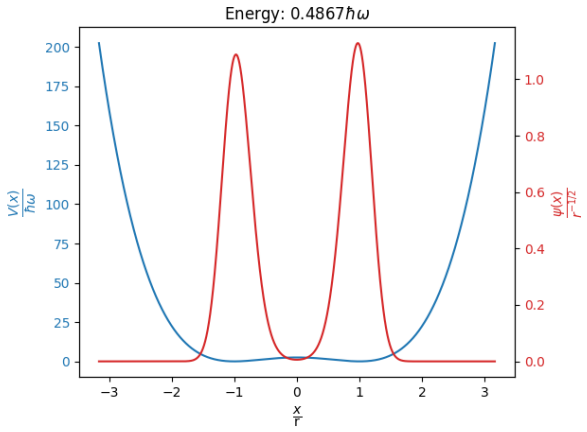

Figure 5: Finer grid for 1D: $N = 20001$, $\mu = 20$ and $\epsilon^2 = 10^{-7}$

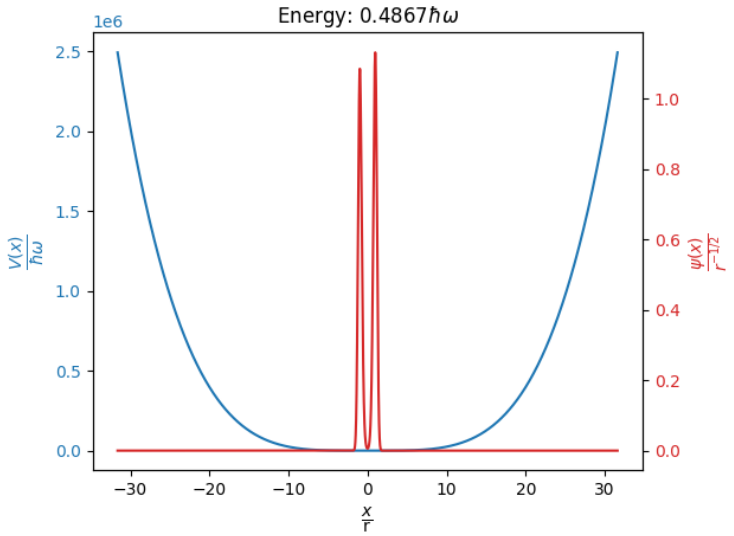

Figure 6: Continuum Limit: With $N = 20001$, $\mu = 20$ and $\epsilon^2 = 10^{-5}$

is working as expected.

- **The Heisenberg uncertainty principle** is true for both 1D& 2D, however $\sigma_x \sigma_y$ is not approximately $\frac{\hbar}{2}$, like in the QHO case.

5.1.2 Energy

- For the energy, we make the following Taylor approximation around minima (x=a) for our potential (first derivative does not appear because we are at minima):

$$V(x) = V(a) + \frac{1}{2}V\text{“}(a)x^2 + ...$$

With

$$\partial_x V = \frac{mw^2}{2r^2}(x^2 - r^2)x$$
$$\partial_x^2 V = \frac{mw^2}{2r^2}(x^2 - r^2) + \frac{mw^2 x^2}{r^2}$$

We see (with the help of the plots too) $x = \pm r$ are the minima. If we plug $x = \pm r$ to the Taylor approximation:

$$V = \frac{mw^2 x^2}{2} + ...$$

So in the quadratic order, the problem is an harmonic oscillator! This means that we must get energy values close to $0.5\hbar w$, which is the ground state of harmonic oscillator. This is what we also observed in our results!

- **Continuum limit**: The energy value does not drastically change for the finite case, which is a good sign for convergence of the code. Moreover, the calculated energies for "finer grid" and continuum limit cases are the same, meaning just taking the $L \to \infty$ limit does not affect the result significantly.