1D Quantum Harmonic Oscillator

Abstract

In this assignment, the 1D Schrödinger equation with a harmonic potential is solved numerically. The code is then evaluated in terms of correctness, numerical stability, discretization, flexibility and efficiency.

Theory

The hamiltonian that was used is the following:

$$\hat{H} = \frac{\hat{p}^2}{2m} + \frac{1}{2}m\omega^2\hat{x}^2 \tag{1}$$

In the x-space representation, which will be used from now on, the momentum operator $\hat{p}$ can be written as $\hat{p} = -i\hbar\frac{\partial}{\partial x}$. In a discretized setting, with the space of interest being divided in N points, this hamiltonian can be written by using an $N \times N$ matrix. There are two contributions to this hamiltonian, one from the kinetic part (which, in this discretized context, is a discrete laplacian) and one from the potential. The discrete laplacian is, in general, a tridiagonal matrix with -2s in the diagonal and 1s on either side, while the potential, being dependent only on x, can be written as a diagonal matrix, with the i-th element being the value that the potential takes on the i-th point in the discretization grid.

The numerical method that was employed is the finite difference algorithm, which is based on the above description, and consists in diagonalizing the hamiltonian to find the eigenvalues and eigenvectors of the discrete system. While this method produces approximated results, this particular system has an analytical solution to which results can be compared. Given the equation

$$\hat{H}\,|\psi\rangle = E\,|\psi\rangle \tag{2}$$

with $\hat{H}$ as above, the analytical solution reads

$$\psi_n(x) = \frac{1}{\sqrt{2^n\,n!}} \cdot \left(\frac{m\omega}{\pi\hbar}\right)^{1/4} \cdot e^{-\frac{m\omega x^2}{2\hbar}} \cdot H_n\left(\sqrt{\frac{m\omega}{\hbar}}x\right), \qquad n = 0, 1, 2, \ldots \tag{3}$$

corresponding to eigenvalues

$$E_n = \hbar\omega\left(n + \frac{1}{2}\right) \tag{4}$$

where H_n are the Hermite polynomials of order n. The linear behavior of the eigenvalues will prove to be an effective way to inquire about the quality of the solution provided by the numerical approach. Since the space is being modeled as a finite grid of points, a set of boundary conditions has to be included; depending on the choice, a 'shadow potential' is introduced, which limits the solutions to only have non-null values on a finite interval. This will be discussed more in detail in the Results section.

Code development

Two separate Fortran programs were developed, one for computing the approximated solution using the finite difference method and one for computing the analytical solution provided above. This makes it possible to compare the predictions with the numerical results. All utilities for computing the necessary quantities are collected in various modules, so to improve readability and modularity.

`schrod.f90` : this module contains the functions and subroutines that may be used to solve the 1D quantum harmonic oscillator; contains definitions for the laplacian, the harmonic potential, diagonalization and saving results to file.

`Hermite.f90` : this module contains utilities for the computation of Hermite polynomials. This module has been adapted from this reference.

`debugger_module.f90` : this module contains debugging routines that were used during the development process.

`finite_diffs.f90` : this code contains the actual computations. Of note is the possibility for the user to provide the parameters from the command line, as to make it as flexible as possible. The `EQUIVALENCE` statement was used to create a dictionary of all parameters, so to make their insertion more readable. The normalized eigenvectors and the probability densities are computed.

`analytic.f90` : this code contains the computation of the analytical solutions; as in `finite_diffs.f90`, the user can provide input parameters from the command line. The normalized eigenfunctions and the probability densities are computed.

A Python script is then used to take all input parameters from the user, to then provide them to the executables. A gnuplot script then produces relevant plots, and all files are then stored in an appropriate folder, as to not crowd the directory.

The code for the finite difference method is as follows. A lower and upper extremes are used, together with the number of points N, to create a grid of points. A discrete laplacian is then created, with step size equal to $h = (x_{up} - x_{low})/N$. The harmonic potential is also computed on these points, and multiplied by the appropriate constants. The diagonalization then produces both eigenvectors and eigenvalues. The eigenvectors are then normalized with respect to the grid: as the `ZHEEV` subroutine returns eigenvectors of unitary norm, they need to be divided by $\sqrt{h}$, so that the sum of the square moduli of each entry equals 1. This makes the integral of the probability distribution equal to 1.

Since the hamiltonian in this assignment is a real, tridiagonal matrix, one could argue that a better choice performance-wise is to use tridiagonal-specific subroutines, like `DPTEQR`, in place of something as generic as `ZHEEV`. However, in order to make the program more flexible, a less specific subroutine was preferred. The `INFO` value was also used to communicate any eventual error in the diagonalization.

Results

An example of a solution, together with the respective analytic analogous, is shown. With $\hbar = 1$, the parameters are $x_{low} = -2$, $x_{up} = 2$, $\omega = 15$, $N = 500$.

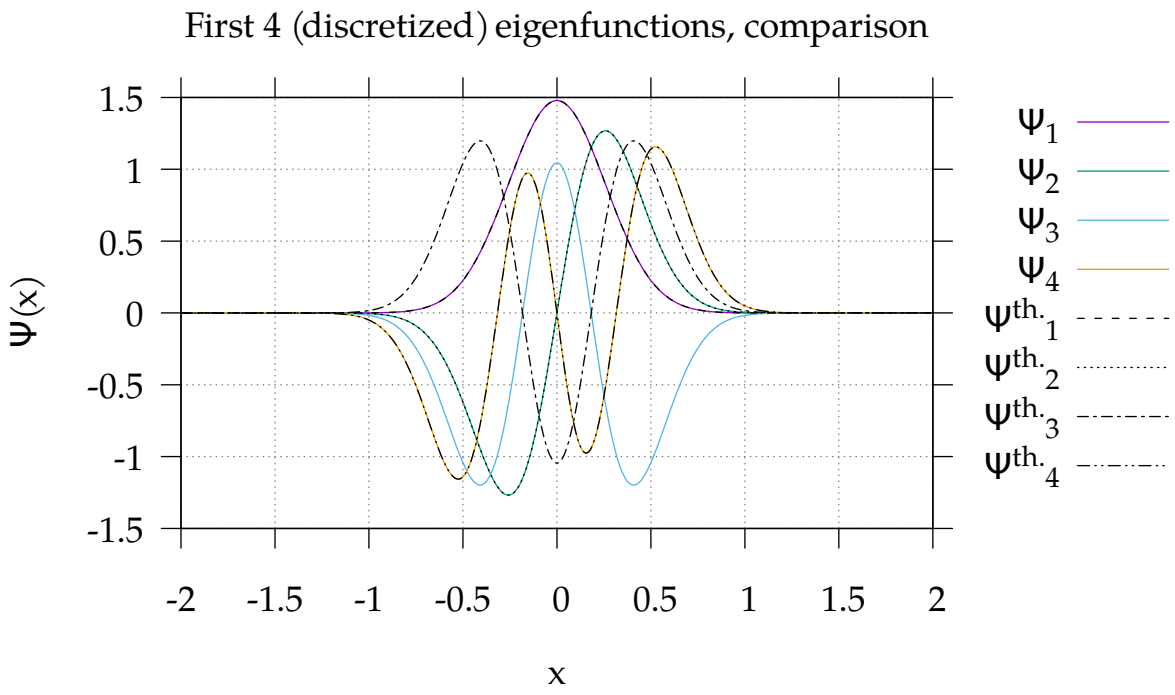

Figure 1: Comparison between the numerical (colored) and analytical (black, dashed) solutions. $x_{low} = -2$, $x_{up} = 2$, $\omega = 15$, $N = 500$

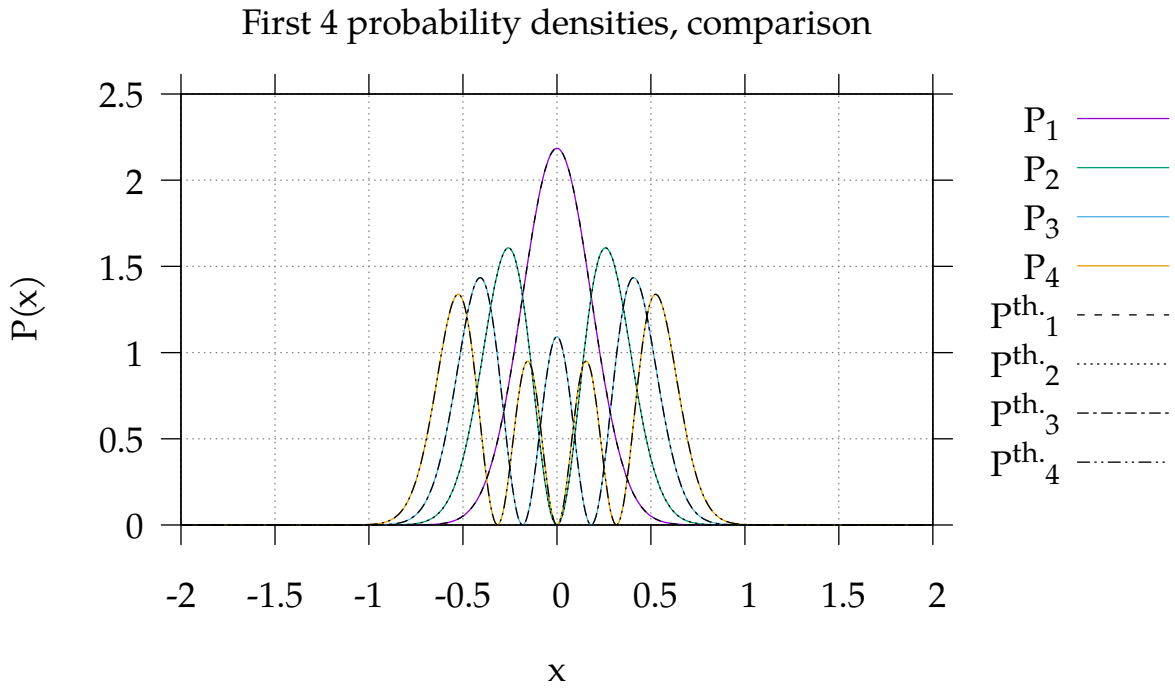

Figure 2: Comparison between the numerical (colored) and analytical (black, dashed) probability densities. $x_{low} = -2$, $x_{up} = 2$, $\omega = 15$, $N = 500$

The overlap between the analytical and numerical solutions is very good. In Fig. 1, only ψ_3 and its theoretical counterpart do not seem to overlap; however, since the normalized eigenvectors are determined up to a -1 constant, one should not be alarmed. Indeed, the probability densities in Fig. 2 show that the overlap is good in each of the 4 considered solutions. In Tab. 1, the values of the first 6 eigenvalues are reported, both the numerical estimates and the theoretical values.

Index	**Numerical**	**Analytical**
1	1	1
2	2.9998	3
3	4.9995	5
4	6.9989	7
5	8.9981	9
6	10.9969	11

Table 1: The values of the first 6 eigenvalues in E_0 units.

Even if the results, up to this point, seem to agree with the theory, if further values are computed, some differences arise.

Index	**Numerical**	**Analytical**
25	50.9710	51
26	53.0176	53
27	55.1052	55
28	57.2505	57
29	59.4683	59
30	61.7698	61
40	89.8344	81
50	126.5037	101

Table 2: The values of higher order eigenvalues in E_0 units.

As Tab. 2 shows, from the 26-th eigenvalue onward, the numerical estimates no longer undershoot, and instead overestimate; this effect becomes more and more pronounced as the index increases.

This effect becomes much clearer if the parameters are changed; with $N = 500$, $x_{low} = -1$, $x_{up} = 1$, $\omega = 1$, the following plots are produced. Clearly, in this case the results do not agree. The same can be seen if the eigenvalues are reported (Tab. 3). The agreement does not hold for any eigenvalue.

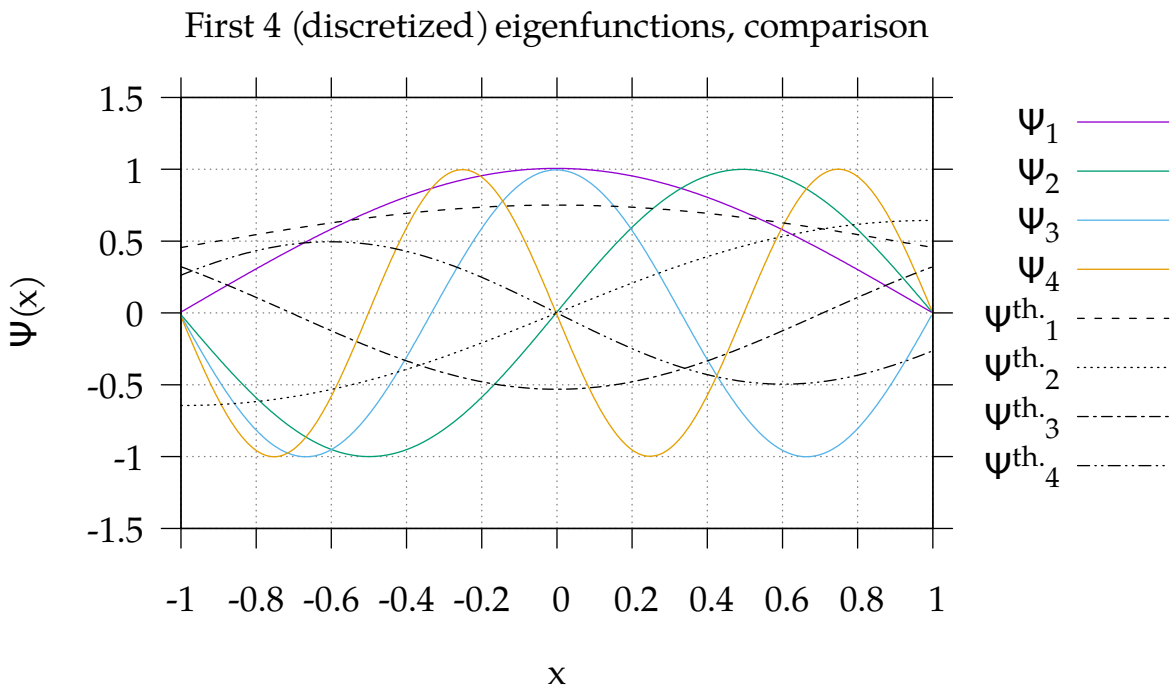

Figure 3: Comparison between the numerical (colored) and analytical (black, dashed) solutions. $N = 500$, $x_{low} = -1$, $x_{up} = 1$, $\omega = 1$

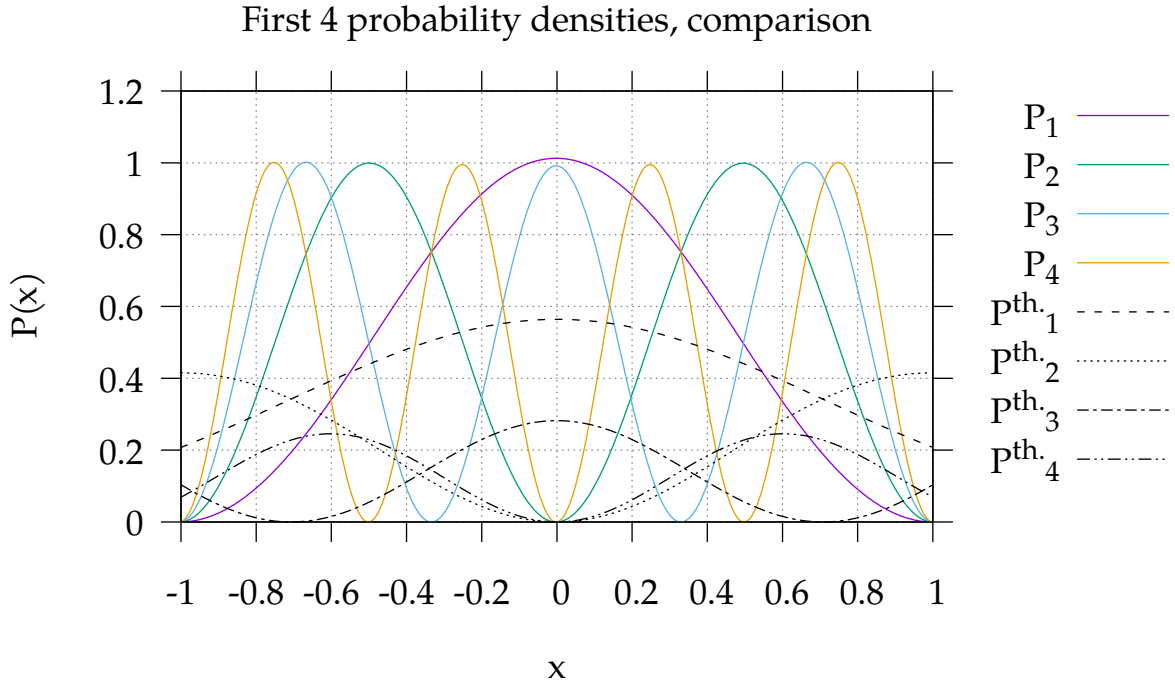

Figure 4: Comparison between the numerical (colored) and analytical (black, dashed) probability densities. $N = 500$, $x_{low} = -1$, $x_{up} = 1$, $\omega = 1$

Index	Numerical	Analytical
1	1	1
2	3.9082	3
3	8.6682	5
4	15.3198	7
5	23.8681	9
6	34.3142	11

Table 3: The values of the first 6 eigenvalues in E_0 units.

The reason for this seemingly inconsistent behavior can be traced to the finiteness of the interval: this imposes to choose a set of boundary conditions. In this case, such choice was to simply set the values of the function at the extremes to 0. This equates to taking a tridiagonal laplacian. If, for instance, one were instead to take periodic boundary conditions, the first and last values in the secondary diagonal of the laplacian should also be set to 1. Taking the former as boundary conditions, the eigenvectors are bound to be null at the extremes of the considered interval; in this sense, the boundary conditions act as a 'shadow potential', just like an infinite well would. This also has repercussions on the eigenvalues, as they diverge from a linear behavior; indeed, as Tab. 3 shows, they exhibit a roughly square dependence on the index, which is consistent with the behavior of the eigenvalues of an infinite well.

Further trials showed that:

- keeping the range fixed, a higher ω results in more and more agreement between theoretical and numerical results; conversely, a wider range for a fixed ω has the same effect;
- due to the aforementioned finiteness, for any combination of ranges and values of ω, at some value of the index n of eigenvalues and eigenvectors the numerical results will diverge from the theoretical ones.

As a way of summarizing this, one may argue that, in this context, the potential is actually a sum of a harmonic potential and an infinite well, which is null in the range that has been chosen and infinite elsewhere.

Employing periodic boundary conditions resulted in energy eigenvalues getting doubled (as the topology of the problem changed, so do the properties of the solutions). As a consequence, no results about that choice are reported here; however, changing a simple flag in `finite_diffs.f90` will produce those solutions.

Code evaluation

As far as correctness goes, this program performs as expected, replicating theoretical results, provided appropriate conditions are met.

Numerical stability is also achieved, as all the involved quantities are double precision (or 4 bytes, if integer). The diagonalization subroutine, which is the most troublesome part of the program, is done by a `LAPACK` subroutine, the result of which is controlled through the `INFO` variable. This is also related to the discretization of the algorithm:

of course, one should employ a high number of points in the grid, so to create smooth approximations. This is even more important if one were to require high order solutions: a Hermite polynomial of degree n has $n-2$ changes of concavity, and so may require many points to be appropriately represented.

Flexibility is a main characteristic of these programs, as they allow the user to input most of the physical parameters. Furthermore, other different potentials may be added in the `schrod.f90` module, and the main program would only need minor adjustments.

Efficiency has not been a major focus: as was already mentioned, all subroutines were kept as general as possible. The optimization flags of the compilers, however, seem to be enough to produce a fast program.

Comments and self evaluation

I found this assignment to be very interesting; this has effectively been my first contact with numerical solution of differential equations, and it has also proven useful to strengthen my understanding of the machinery of quantum mechanics.

I would evaluate this assignment positively, especially when compared to the previous one: both the results and the time management seem to me to having improved.