Continuous Time-independent Schrodinger Equation

Alberto Chimenti

Exercise n.6
Information Theory and Computation - 2019

Abstract

In the following diagonalize and compute eigenvectors and eigenfunctions of the hamiltonian of the quantum harmonic oscillator.

1 Theory

The hamiltonian of the quantum harmonic oscillator can be written as:

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

One can find analitically the solutions of the Schrodinger equation which are:

$$\psi_n = \frac{1}{\sqrt{2^n n!}} \Big(\frac{m\omega}{\pi\hbar}\Big)^{\frac{1}{4}} e^{-\frac{m\omega x^2}{2\hbar}} H_n\Big(\sqrt{\frac{m\omega}{\hbar}}x\Big) \qquad\qquad E_n = (2n+1)\hbar\omega$$

with the Hermite polynomials defined like:

$$H_n(z) = (-1)^n e^{z^2} \frac{d^n}{dz^n}\big(e^{-z^2}\big)$$

Therefore we proceded in computing the first 4 eigenvectors and eigenfunctions with the method explained in the following section in order to compare them with the theoretical results.

2 Code Development

Now, for simplicity we set the constants equal to 1 for the description of the computational approach. We can write the Schrodinger equation as:

$$\Big(-\frac{\partial^2}{\partial x^2} + x\Big)\psi = E\psi$$

Finding the solution for such equation is equivalent to solving the eigenproblem for the hamiltonian. In order to be able to compute it, one has to discretize the space by choosing a cut-off and an internal spacing for the mesh. The choice was to take a symmetric interval with respect to zero setting $x_{min} = -5$ and $x_{max} = +5$. The values were chosen by looking at the results of the wavefunctions during previous tries with the computation and the sapcing was calculated as:

$$h = \frac{x_{max} - x_{min}}{N}$$

with N equal to the number of points in the mesh. The estimation of the second derivative operator can be performed using finite difference method as follows:

$$\frac{\partial^2}{\partial x^2}\psi_n = \frac{\psi_{n-1} - 2\psi_n + \psi_{n+1}}{h^2}$$

Therefore one can write the full equation in discretized form:

$$\frac{1}{h^2}\Big[-\psi_{n-1}-\psi_{n+1}+\big[2+h^2(x_{min}+(n-1)h)^2\big]\psi_n\Big]=E_n\psi_n$$

In matrix form it can be explicitly written as:

$$\frac{1}{h^2}\begin{pmatrix} 2+h^2(x_{min})^2 & -1 & & & \\ -1 & 2+h^2(x_{min}+h)^2 & -1 & & \\ & & \dots & & -1 \\ & & & -1 & 2+h^2(x_{min}+N\cdot h)^2 \end{pmatrix}\psi=E\psi \tag{1}$$

Here we note that the second order derivative computed with three points yields a $o(h^4)$ precision in the computation. Note that the cutoffs choice is important since here we are supposing that the function is zero out of the mesh as one can see from the matrix form previously written.

The code was implemented in fortran with separate subroutines to create the matrix and compute the solution of the eigenproblem using the LAPACK subroutine *zheev*. In order to avoid problems with finite precision, the factor $\frac{1}{h^2}$ was omitted in the computation of the matrix and just used to rescale the eigenvalues. Another this one has to consider is that the discretization of the space introduces some changings in the normalization of our eigenfunctions. This was crucial in comparing the results obtained with the theoretical curves. Indeed all the eigenfunctions were rescaled by a factor equal to $\frac{1}{\sqrt{h}}$.

3 Results

The code was run by choosing 1000 points for the mesh. It was chosen to be the best trade-off between computation times and goodness of the results for the eigenvalues.

E_{comp}	E_{th}
0.999964	1
2.999986	3
4.999932	5
6.999885	7

In the following we plot the comparison between the calculated eigenvalues and the theoretical ones:

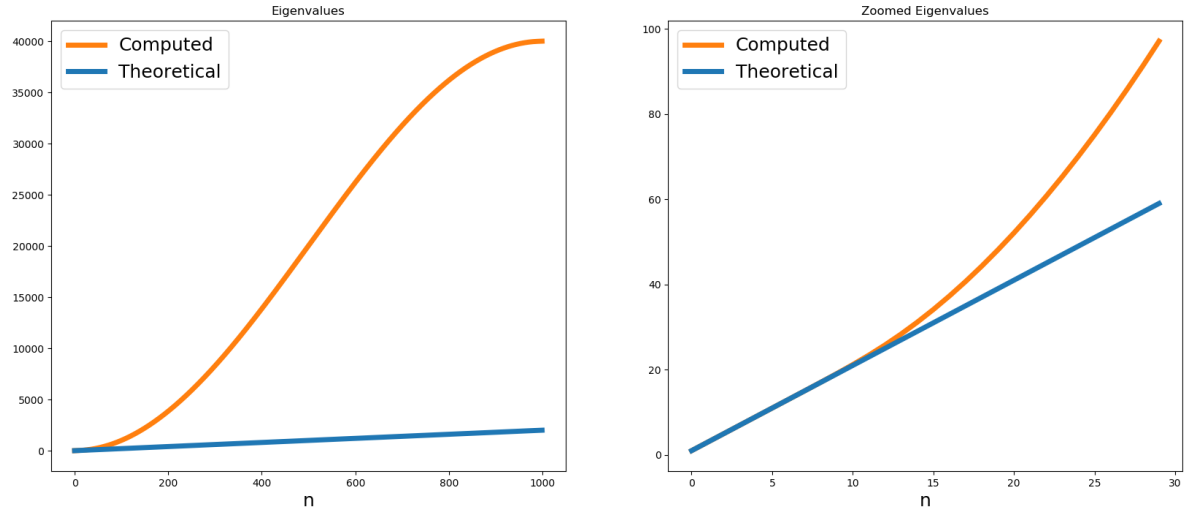

Figure 1: Eigenvalues (zoomed version on the right)

As one can see, they diverge quite significantly after $n \simeq 20$. It is worth noting that the distribution saturates which might be connected with the discretization choices. As an example, We plot the results for the eigenfunctions and their comparison with the theoretical ones.

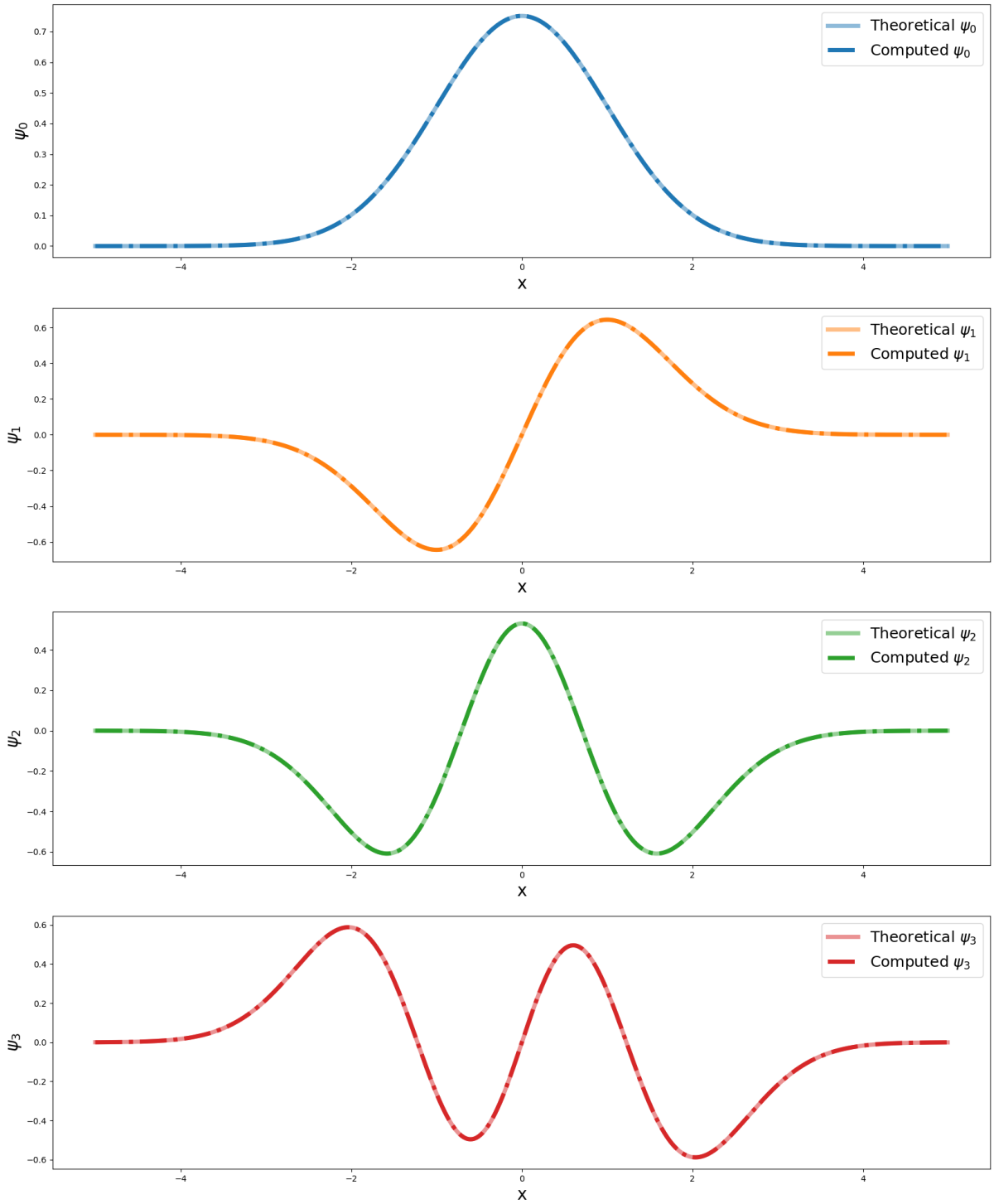

Figure 2: Eigenfunctions

As we can see, the choice for the cut-off seems quite reasonable judging by the behaviour of the eigenfunctions but still the distribution of the eigenvalues doesn't follow the theoretical one in any way. The discrepancy seems to be too big to be due to finite precision issues. Therefore one can address, as stated before, the further research to the choice of the mesh parameters. However, the distribution doesn't seem to change with the spacing/number of points in the mesh, indeed, the shape of the distribution seems to be invariant to the density of the grid (after overcoming a precision threshold $\simeq 100$ points). The saturation is always present, no matter the value of N.

4 Self-evaluation

- **Correctness** The agreement with the theoretical values is very good for the first eigenvalues and eigenfunctions and as we can see the eigenvalues tend to diverge from the actual distribution over a certain value. Here one has to think that we are making a very strong approximation in pretending that our cut-off would suit the actual whole real axis space. In general, the bigger the interval taken the more the eigenvalues curve should "lay" on the theoretical one. This still means that in order to obtain a good result one has to increment the number of points in the same manner. Therefore, the correctness of the algorithm seems to be satisfactory in case we are interested into the low energy levels of the system.

- **Stability** The numerical stability was tested using some debug code previously written, in order to check the hermitianity of the matrix or the actual square size. The run of LAPACK performance variable *lwork* was automatized by runnning the subroutine one first time setting it equal to -1 and then extrapolating the actual best value from the *work* vector and running it again. A good implementation would be to add a discrete integral subroutine to check the normalization of the calculated wavefunctions.

- **Accurate discretization** The chosen finite different method which involves the use of 3 points gives a very good degree of accuracy. Indeed, by considerng the 5 point case, which performs better in first derivative calculations, doesn't provide any better result in the second derivative case, yielding accuracy of the order $o(h^4)$. On the other side the algorithm results are very sensitive to the choice of the interval.

- **Flexibility** The methods used were all generalized in case of complex matrices, therefore one should be able to use, for example, the diagonalizing function on a generic complex matrix. What would be changed is the check over the normalization of the wavefunctions which could be a good generalized subroutine to implement. The program prompts the choice of the number of points desired in the mesh and therefore calculates the spacing in the symmetric inteval. It would be an easy upgrade to be able to change the extremes of the interval as well, since this is such an important arbitrary choice in the evaluation of the eigenvalues.

- **Efficiency** On the efficiency side, some tradeoffs where put in place. The use of double complex matrices is an overkill for the problem since we are dealing with a real matrix. Saving double the memory certainly is relevant when dealing with high intervals and therefore high dimensional matrices. Some generic calculations of related objects were kept standardized for every defined matrix, like the computation of the adjoint, in order to maintain some flexibility. All of these unnecessary steps slow down the computation time. On the contrary, by leaving out the evaluations and checks one could run into problems with unexpected behaviour. So one straightforward improvement to reduce the memory cost would be to specify the solving method for real double precision matrices instead of complex ones.