Continuous Time-indipendent Schrödinger Equation

Davide Maniscalco

November 19, 2019

Abstract

In this paper a computational solution of the quantum harmonic oscillator is presented and discussed.

COMPILATION INSTRUCTIONS:

- The suggested compilation for the code (that handles properly lapack, output filename) is the following: `gfortran Ex6-Maniscalco-CODE.f90 -llapack -o Ex6-Maniscalco.out`
- The compilation suggested for the python script is: `python3 Ex6-Maniscalco-SCRIPT.py`
- The `figure` folder is empty: there is were the figures produced by the python script are stored into

1 Theory

The problem consists in solving the one dimensional time-independent Schrödinger equation $\hat{H}\psi = E_n\psi$ with the hamiltonian $H = \hat{p}^2 + \omega^2\hat{q}^2$, namely, in the position representation:

$$\left(-\frac{\hbar}{2m}\frac{\partial}{\partial x^2} + \omega^2 x^2\right)\psi = E\psi \, .$$

From the theory, it is known that the eigenvalues are

$$E_n = 2n - 1$$

with n natural. For the eigenvectors we have instead that:

$$\psi_n(x) = \left(\frac{2m\omega}{\pi\hbar}\right)^{1/4} \frac{1}{2^n n!} H_n\left[\sqrt{\frac{2m\omega}{\hbar}}x\right] \exp\left(-\frac{m\omega}{\hbar}x^2\right)$$

where H_n are the Hermit polynomials. In particular, the first three eigenfuncitons are:

$$\psi_0(x) = \pm\left(\frac{2m\omega}{\pi\hbar}\right)^{1/4} \exp\left(-\frac{m\omega}{\hbar}x^2\right)$$

$$\psi_1(x) = \pm\left(\frac{2m\omega}{\pi\hbar}\right)^{1/4} x\sqrt{\frac{4m\omega}{\hbar}} \exp\left(-\frac{m\omega}{\hbar}x^2\right)$$

$$\psi_2(x) = \pm\left(\frac{2m\omega}{\pi\hbar}\right)^{1/4} x\left[\frac{4m\omega}{\hbar} - 1\right] \exp\left(-\frac{m\omega}{\hbar}x^2\right)$$

where the $\pm$ is to recall that both sign are possible, because the meaningful physical quantity is the square modulus of the wave function.

From now on we will take natural units for sake of simplicity, i.e. setting: $\hbar = 1, \omega = 1, m = 1/2$. The problem is computationally attached with the finite difference method, that consists in discretizing the space in a closed interval (and therefore obtaining a discretized solution), and in computing the second derivative with its discretized form. The interval is chosen symmetrical respect to the origin; calling $x_{\min}$ and $x_{\max}$ the two extremes, we have $[x_{\min}, x_{\max}]$ and $x_{\min} = -x_{\max}$. The number of spacings the interval is divided into is called N, and therefore the step of the discretizaton is simply $d = 2x_{\max}/N$. Therefore, with this method $N + 1$ eigenvalues are calculated and the eigenfunctions are calculated in $N + 1$ points, where the n-th point is placed in $x_{\min} + (n - 1)d$. Summing up, we have that the equation to be solved by the computer is:

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

that in the matrix form becomes:

$$\frac{1}{d^2}\begin{pmatrix} 2 + d^2(x_{\min})^2 & -1 & & & \\ -1 & 2 + d^2(x_{\min} + d)^2 & -1 & & \\ & -1 & ... & -1 & \\ & & & ... & \\ & & & -1 & 2 + d^2(x_{\min} + Nh)^2 \end{pmatrix} \psi = E\psi\ ,$$

and now the problem is solved diagonalizing the matrix.

2 Code development

A fortran code for the diagonalization has been written, and a python script to handle inputs and outputs and for graphics. The fortran takes as input from a file named `input.dat` the $x_{\max}$ value and the number of spacings N. Then it initializes the matrix as explained in the previous section and performs the diagonalization through the `dsyev` routine. The whole vector containing the eigenvalues is printed in the second column of an output file named `eigenvectors.dat` while in the first column the natural numbers up to $N + 1$ are printed. Concerning the eigenvectors, only the first three are printed as columns in a file named `eigenvectors.dat`; the first column of this file contains instead the respective position along the x axis. The code contains also a checkpoint on the eigenvalues that can be activated setting the `debug` variable to true, and is ready to run for different values of the constants $\hbar, \omega, m$ that by default are set to $1, 1, 1/2$.
All the process can be handled efficently running the python script. It creates the `inputdim.dat` file once received as inputs the values for N and $x_{\max}$, and runs the fortran code. Then it collects the results and make four plots (that are stored in the `figures` folder): the first to compare the computed eigenvalues with the theoretical ones and the last three to compare the first three eigenfunctions with the theoretical ones, defined in the script itself. Finally, to give an estimation of the goodness of the computed eigenvalues with respect to the theory, it computes the mean absolute error as follows:

$$MAE = \frac{1}{N+1}\sum_{i=1}^{N+1}|\hat{y}_i - y_i|$$

where $\hat{y}_i$ are the theoretical eigenvalues for the quantum harmonic oscillator and y_i are the ones computed by the program.

3 Results

A first trial was made setting $x_{\max} = 5$ and $N = 1000$; the relative results are shown in Fig.1, Tab.1.

λ_{th}	λ_{comput}
1	0.99999
3	2.99997
5	4.99992
1997	40008
1999	40015
2001	400015

Table 1: First and last three eigenvalues for $N = 1000, x_{\max} = 5$

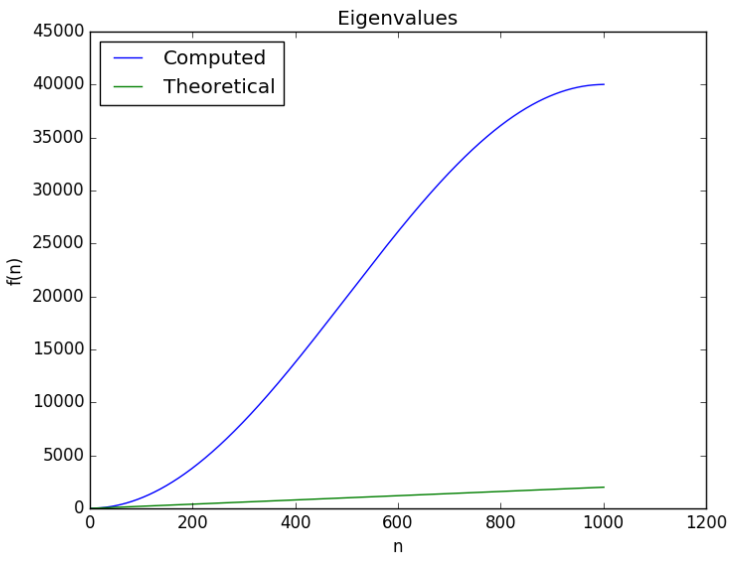
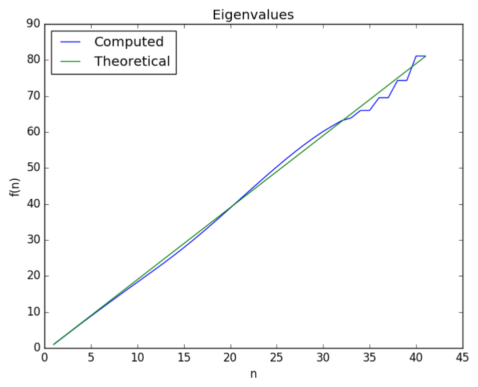
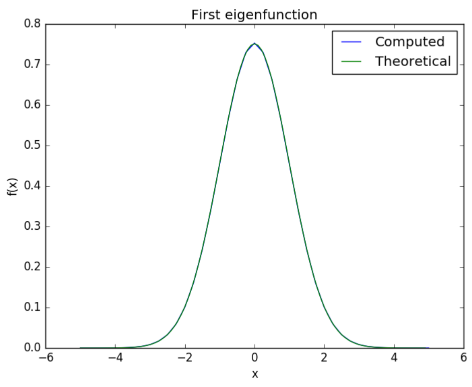


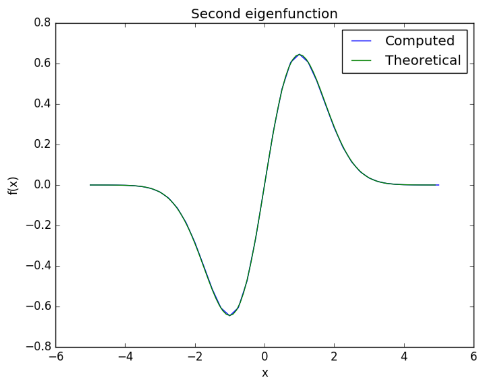
Figure 1: Eigenvalues for $N = 1000, x_{\max} = 5$



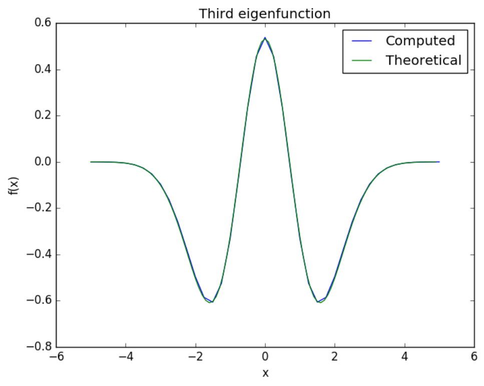
(a) Eigenvalues for $N = 40, x_{\max} = 5$



(b) ψ_0 in $[-5; 5]$ for $N = 40$



(c) ψ_1 in $[-5; 5]$ for $N = 40$



(d) ψ_2 in $[-5; 5]$ for $N = 40$

Figure 2: Eigenvalues and first eigenvectors in $[-5; 5]$ for $N = 40$

While the computation is optimal for the first eigenvalues (the worst error is $\simeq 0.0016\%$), it is completely wrong for the high ones ($MAE_{5;1000} = 19007$).
Unfortunately, the program seems to be unable to compute correctly all the eigenvalues for "wrong" values of the parameters. Keeping $x_{\max}$ fixed, the best parameter was found as the one minimizing the MSE. This way it was found:

$$x_{\max} = 5 \qquad N = 40 \qquad \rightarrow \qquad d = 0.25$$

that lead to the results shown in Fig.2, Tab.2 with

$$MAE_{5;40} = 0.943\,.$$

λ_{th}	λ_{comput}
1	0.996
3	2.980
5	4.949
77	74.317
79	81.103
81	81.103

Table 2: First and last three eigenvalues for $N = 40, x_{\max} = 5$

As it can be seen, there is not an evident difference visible by eye between the computed and the theoretical eigenfunctions. Concerning eigenvalues, this is the best estimation that

the program can do. As it can be seen, there has been a loss of precision for the lowest eigenvalues, and the last eigenvalues start degenerating for unknown reasons.
At this point, it was tried to change the $x_{\max}$ value also; for each tried value, the best (i.e. the one minimizing the MAE) value of N was found, leading to the results reported in Tab.3.

$x_{\max}$	Best N	d	$2x_{\max}^2/N$
3	15	0.4	1.2
5	40	0.25	1.25
7	127	0.142	1.278
9	156	0.128	1.282
10	500	0.072	1.296
18	619	0.065	1.292

Table 3: Best MAE parameters for various $x_{\max}$ values.

It seems that doubling the maximum abscissa, the new best space width is about half of the previous, such that

$$x_{\max} \cdot d = \frac{2x_{\max}^2}{N} = c$$

where c is a constant. In order to give an estimation of c, a fit of the points with the function $N = 2x_{\max}^2/N$ was made, leading to

$$c_{\text{best}} = (1.292848 \pm 4 \cdot 10^{-6}) \, .$$

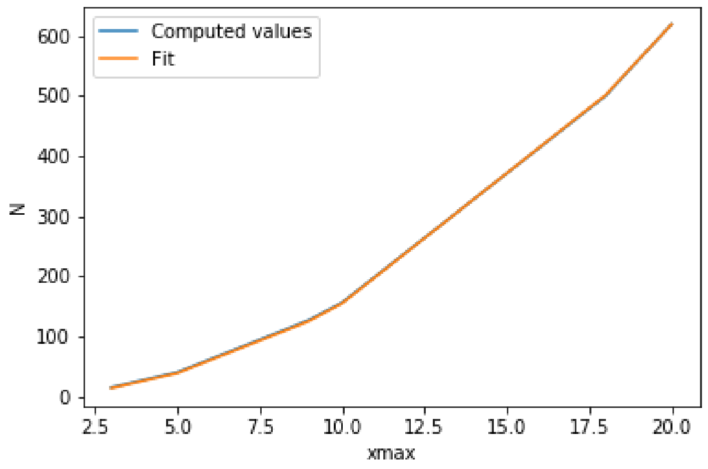


Figure 3: Best number of spacings as a function of the maxiumum abscissa

In spite smaller matrices make the program more efficient, it is preferable to choose bigger matrices, for which the precision, even for small eigenvalues, is higher.

4 Self Evaluation

In this section it is discussed the written program in terms of correctness, stability, accurate discretization, flexibility.

The program leads to correct results for low eigenvalues and eigenfunctions, while moving to higher values, there is not sure anymore of good accordance with the theory. Thank to the fits, the MAE, and the checkpoint on eigenvalues one can check his results. The program is unfortunately not stable: only certain values of the parameter lead to resonable results. For this purpose the best possible discretization rule has been found, with the available time and the information. The program has a small flexibility thank to the possibility of changing the value of the parameters $\hbar, \omega, m$. More could have been done, for example giving the possibility to choose an interval not centred in 0, or giving the possibility to choose different boundary conditions for the discrete computation.