Time-Dependent Schrödinger Equation

Davide Maniscalco

December 3, 2019

Abstract

In this paper, the evolution under a time-dependent hamiltonian of the ground state wave function of the harmonic oscillator is computationally studied using the split operator method.

NOTE ABOUT THE COMPILATION

- The Fortran code, in order to make it use lapack and fftw3 packages, must be compiled the following way: `gfortran -I/usr/local/include Ex7-Maniscalco-CODE.f90 -llapack -lfftw3 -o Ex7-Maniscalco.out`, where the folder `/usr/local/include` is the one that should contain the `fftw3.h` file.
- The python script (that also performs compilation and execution of the fortran code) should be run the following way: `python3 Ex7-Maniscalco-SCRIPT.py`

1 Theory

Let's consider the following Schrödinger equation:

$$\hat{H}(t) = \frac{\hat{p}^2}{2} + \frac{1}{2}(\hat{q} - q_0(t))^2$$
$$q_0(t) = t/T$$

This equation presents a time-dependent potential, that basically leads to the translation of the function the hamiltoninan is applied to, and we should have:

$$|\psi(t)\rangle = e^{-i\hat{H}t}|\psi_0\rangle$$

In particular, we apply this hamiltonian to the ground state eigenfunction ψ_0 of the harmonic oscillator. We expect to see the function translating without changing its shape till the last point: in the end we should in fact have:

$$\psi(T) = \psi_0(x-1)$$

The ground state eigenfunction of the harmonic oscillator was computed in the previous exercise with natural units $\hbar = 1$, $\omega = 1$, $m = 1/2$.

The idea for solving computationally the equation is the split operator method. First of all we operate a discretization in the time interval $[0, T]$, where the time step is called Δt. Then, if Δt is small enough, it worth that $\psi(t+\Delta t) \simeq e^{-i\hat{H}\Delta t}\psi(t)$, where the hamiltonian is the one defined above, composed by a kinetic and a potential term. Assuming Δt being small we can commute terms committing an error of $O(\Delta t)^3$ and write

$$e^{-i\hat{H}t} = e^{-\frac{i}{2}\hat{V}\Delta t}e^{-\frac{i}{2}\hat{p}^2\Delta t}e^{-\frac{i}{2}\hat{V}\Delta t}\ .$$

Finally, we want to use the $\hat{p}$ operator in the momentum representation, and this can be done with a Fourier transform. The equation that we are going to solve computationally is then:

$$\psi(t+\Delta t) \simeq e^{-\frac{i}{2}\hat{V}\Delta t}\mathcal{F}^{-1}e^{-\frac{i}{2}\hat{p}^2\Delta t}\mathcal{F}e^{-\frac{i}{2}\hat{V}\Delta t} \, . \tag{1}$$

2 Code development

The ground state eigenfunction of the harmonic oscillator, that here is the function ψ_0 at time 0, is computed using the previous code. In the fortran code, the user can select the parameters for the spatial and for the temporal discretizations. These are simply:

$$x_j = x_{\text{min}} + (j-1)\Delta x$$

$$t_i = i \cdot \Delta t$$

with $j \in \{1, ..., N+1\}$ and $i \in \{0, ..., \frac{T}{\Delta t}\}$. Once chosen this, the program makes the calculations for the temporal evolution just according Eq.1. The most tricky part is about the Fourier transforms. Given a vector containing the elements, its discretized Fourier transform is calculated with the package "Fast Fourier Transform in the West", and the same for the antitrasform. The package must be included in each subroutine that use it with the commands:

```
use, intrinsic ::  iso_c_binding
include "fftw3.f03"
```

and properly set in the compilation options. Finally, in order to multiply the vector resulting from the transformation for the exponential containing the p term, a grid in the momentum space must be defined:

$$p_k = \frac{2\pi k}{x_{\text{max}} - x_{\text{min}}} \qquad \text{with} \qquad k \in \{0, ..., N\} \, .$$

These are the moments that fill the moments vector that appears in the exponential between the antitransform and the transform. The unique output is a file named `'eigenstates.txt'`, whose first column contains the tics on the x space, while all the others contain the norm of the wave function from time 0 for all the subsequent time steps.

The python script, instead, is just meant for automation and for graphics. The script gives the possibility to compile and re-run the fortran code or just to make the plots based on the `'eigenstates.txt'` file. The script produces two plots: one that shows the evolution of the wave function at five subsequent evenly spaced instants, and the other that shows just the first and last ones.

3 Results

For what concerns the spatial discretization, all the simulations were made with the values $x_{\text{max}} = 5$ and $N = 1000$.

The unique results similar to the expected ones have been found for $\Delta t = 1$, no matter of the total time. As it can be seen in Fig.1, in this case the shape of the wave function remains the same and the function reaches the coordinate 1 at the last step. However, this is the only case. If the number of time steps is increased (thing that should lead to better results, by the moment that we supposed Δt to be small), completely wrong results are found.

As it can be seen in Fig.2 the wave function neither remains constant nor reaches 1. A lot of attempts have been made (in term of code development, in particular about the momenta discretization) in order to find the bug, but without success.

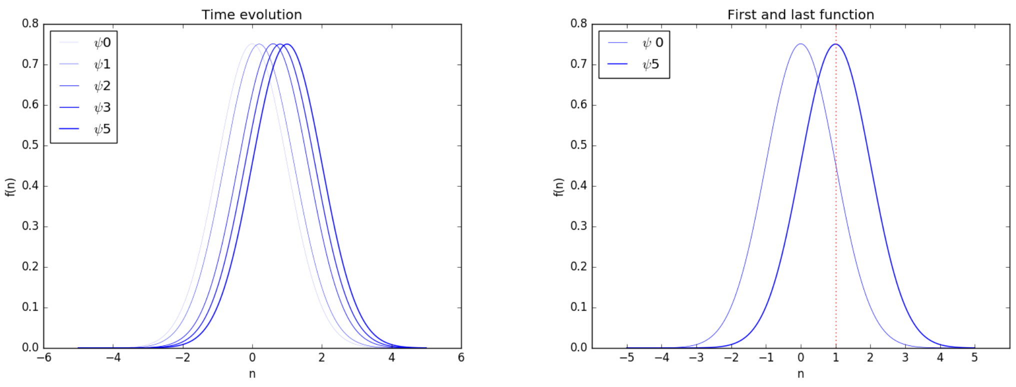

Figure 1: Time evolution for $T = 5$ with 5 time steps

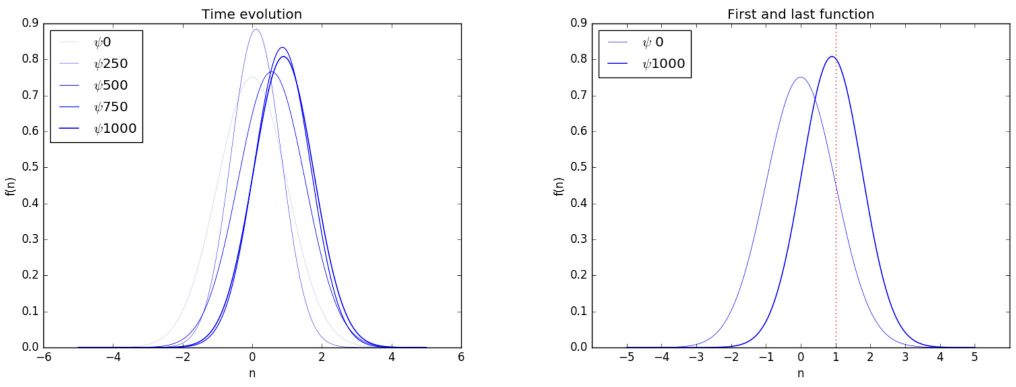

Figure 2: Time evolution for $T = 5$ with 1000 time steps

4 Self evaluation

Mainly, the installation and usage of the fftw3 package has been learned, together with the procedures to write a fortran program for the time evolution. The target has not been completely achieved: the function evolves and reaches 1, but it doesn't remain constant during the evolution. The big amount of time spent in trying to fix this prevented to make further analysis (e.g. about errors, error scaling, time scaling).