UNIVERSITY OF PADUA
PHYSICS AND ASTRONOMY DEPARTMENT

Homework #7
Time-Dependent Schrödinger equation

Student name: *Beatrice Segalini* – 1234430

Course: *Quantum Information and Computing 2020* – Professor: *S. Montangero*
Due date: *Novembre 24th, 2020*

Abstract

In this report, the study of a one-dimensional time-dependent Schrödinger equation is displayed. The Split Operator Method is applied to perform such task, thanks to the library `fftw` provided for `Fortran`.

Theory

The Hamiltonian of the system analysed is:

$$\hat{H} = \frac{1}{2}\hat{p}^2 + \frac{1}{2}\left(\hat{q} - q_0(t)\right)^2 \tag{1}$$

where $\hat{p}$ is the momentum operator, $\hat{q}$ is the position one and $q_0(t) = \frac{t}{T}$ is a time dependent shift, with $t \in (0, T]$. In this work, the constants are set to simplify the problem equal to $m = 1$, $\hbar = 1$, $\omega = 1$ and the time evolution of the main physical values will be analysed for different values of T, which influences both the time range and the time-dependent part of the potential in the Hamiltonian.

It is important to remark that the Hamiltonian can be written as sum of a *kinetic* part and a *potential* one, with the latter depending on time, according to the following relationship:

$$\hat{H} = \hat{T}(p) + \hat{V}(x, t) \tag{2}$$

As a consequence, one can approximate the time evolution in the infinitesimal step with the following expression, applying the **Split Operator Method**:

$$|\psi(t + \Delta t)\rangle = e^{-i\Delta t\hat{H}}|\psi(t)\rangle \approx e^{-\frac{1}{2}i\Delta t\hat{V}} e^{-i\Delta t\hat{T}} e^{-\frac{1}{2}i\Delta t\hat{V}}|\psi(t)\rangle \tag{3}$$

One could notice that $\hat{T}$ is diagonal in the momentum basis and $\hat{V}$ is diagonal in the position basis: hence, the time evolution and hence the solution of the time-dependent Schrödinger equation can be so evaluated by performing two changes of basis through a Fourier transform, combined with the expression in Equation 3.

Code development

Discretisation of position, momentum and time spaces. In order to perform the simulation, it is required to generate discretised intervals for both the position basis, the momentum basis, and the time.

To do so, the following code is implemented (Listing 1): the function generates a $N_{int} + 1$ dimensional grid, ranging from `down` to `up`, with N_{int} intermediate equal steps of width `dx`. The results are stored in a user-defined type `Grid1dim`, which allows to include all the relevant informations about a discretised interval within one variable.

In particular, the space range is $[-L, L] = [-50, 50]$ with 10000 intervals, the momentum range is $[-\frac{\pi}{L}, \frac{\pi}{L}]$ with 10000 intervals, and the time one is $[0, T] = [0, 20]$ with 1000 intervals.

The momentum grid, however, needs to be rearranged due to the shift that is performed by the Fast Fourier transform when the change of basis is performed. The first half of the $p-$grid will start from 0 and include the positive values, then the second half will start from the minimum value (which is equal to $-\frac{\pi}{L}$) and sort in increasing order all the negative values.

Listing 1: Function to calculate the lattice.

```
function GenLattice(down, up, N_int)result(grid)

double precision, intent(IN) :: down, up
integer, intent(IN)          :: N_int
type(Grid1dim)               :: grid

grid%minimum = down
grid%maximum = up
grid%N       = N_int+1
grid%dx      = (up-down)/N_int

allocate(grid%lattice(grid%N))

do aa = 1, grid%N
        grid%lattice(aa) = down + (aa-1)*grid%dx
end do

end function
```

Computation of the Fourier transforms and time evolution. The code displayed in Listing 2 is able to reproduce the step-by-step time evolution thanks to the `FFTW` library, which computes easily Fast Fourier Transforms. The program, starting from the initial state, applies the Split operator method, performs the required changes of basis, and compute at each time step the most significant physical values, i.e. the expected value of the position $E(x)$, of the momentum $E(p)$ and their standard deviations $\sigma_x = \sqrt{E(x^2) - E^2(x)}$, $\sigma_p = \sqrt{E(p^2) - E^2(p)}$ for the computed time evoluted wavefunction ψ.

Listing 2: Code that simulates time evolution.

```
do ii = 1, tgrid%N
do jj = 1, xgrid%N
        ! kinetic part
        vect_U_T(jj) = EXP(dcmplx(0d0,-0.5d0) * tgrid%dx * ps(jj)**2d0)
        ! potential part
        vect_U_V(jj) = EXP(dcmplx(0d0,-0.5d0) * tgrid%dx *
               Potential(xgrid%lattice(jj), tgrid%lattice(ii), TT(pp)))
end do

do jj = 1, xgrid%N
        A_q(jj) = vect_U_V(jj) * psi_now(jj)
end do

call dfftw_execute_dft(dfft_plan, A_q, A_p)

do jj = 1, xgrid%N
        B_p(jj) = vect_U_T(jj) * A_p(jj)/sqrt( dble(xgrid%N) )
```

```
end do

call dfftw_execute_dft(idfft_plan, B_p, B_q)

do jj = 1, xgrid%N
        psi_next(jj) = vect_U_V(jj) * B_q(jj)/sqrt( dble(xgrid%N) )
end do

psi_next = psi_next/norm(psi_next, xgrid%dx)

call dfftw_execute_dft(dfft_plan, psi_next, psi_tmp)

psi_tmp = psi_tmp/norm(psi_tmp, pgrid%dx)

q2 = xgrid%lattice**2
p2 = ps**2

ex  = ExpectedValue(psi_next, xgrid%lattice, xgrid%dx)
ex2 = ExpectedValue(psi_next, q2, xgrid%dx)
ep  = ExpectedValue(psi_tmp, ps, pgrid%dx)
ep2 = ExpectedValue(psi_tmp, p2, pgrid%dx)

! write outputs
! position expected value
write(11, *) tgrid%lattice(ii), ex
! variance of position
write(12, *) tgrid%lattice(ii), sqrt(ex2 - ex**2)
! momentum expected value
write(13, *) tgrid%lattice(ii), ep
! variance of momentum
write(14, *) tgrid%lattice(ii), sqrt(ep2 - ep**2)

psi_now = psi_next

end do
```

Results

The time evolution is simulated with the already described code for three different values of the parameter T, namely $T = 1$, $T = 20$, and $T = 100$.

To show the results, Figures 1, 2 and 3 are plotted. In each of them, the expected values of the position and of the momentum are plotted versus time. The coloured bands represent the standard deviation, and are plotted only for the position for visualisation purposes.

Observing the results, one immediately notices that they are consistent with what it was predicted. In fact, the average position is expected to move with the following law:

$$E(x) = \frac{t}{T} - \sin\left(\frac{t}{T}\right)$$

because the system considered is ruled by a harmonic potential with bottom moving at constant velocity $\frac{1}{T}$.

To further prove this, the value $E(x) - \frac{t}{T}$ is plotted against time for $T = 20$ and $T = 100$ (Figure 4).

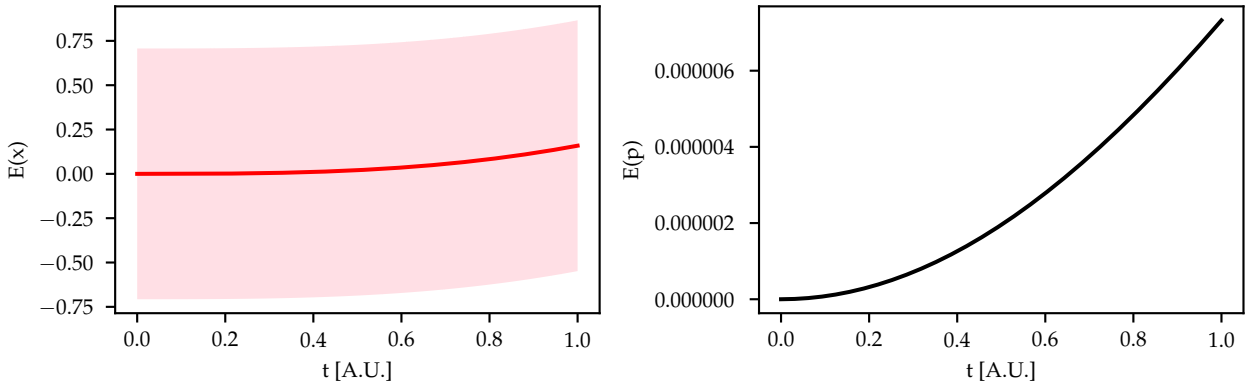

Figure 1: Evolution in time of the expected value of positions $E(x)$ and of momenta $E(p)$ for $T = 1$.

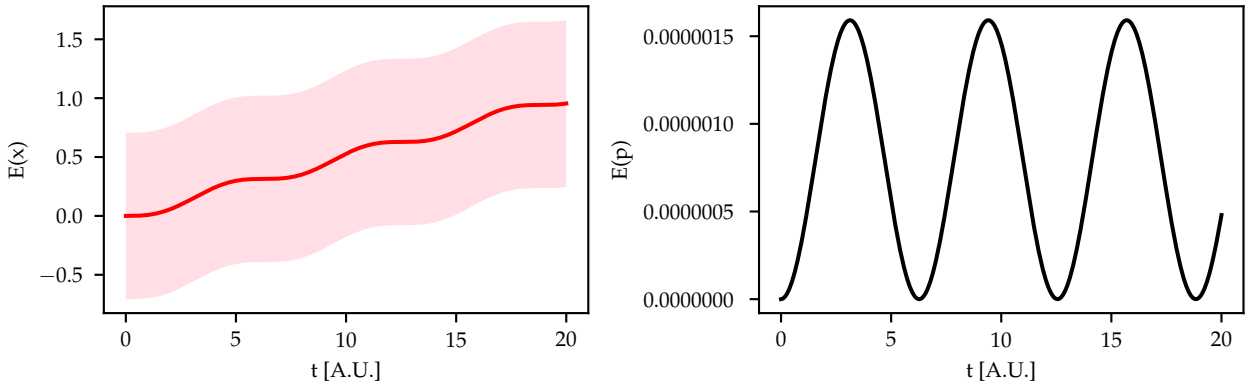

Figure 2: Evolution in time of the expected value of positions $E(x)$ and of momenta $E(p)$ for $T = 20$.

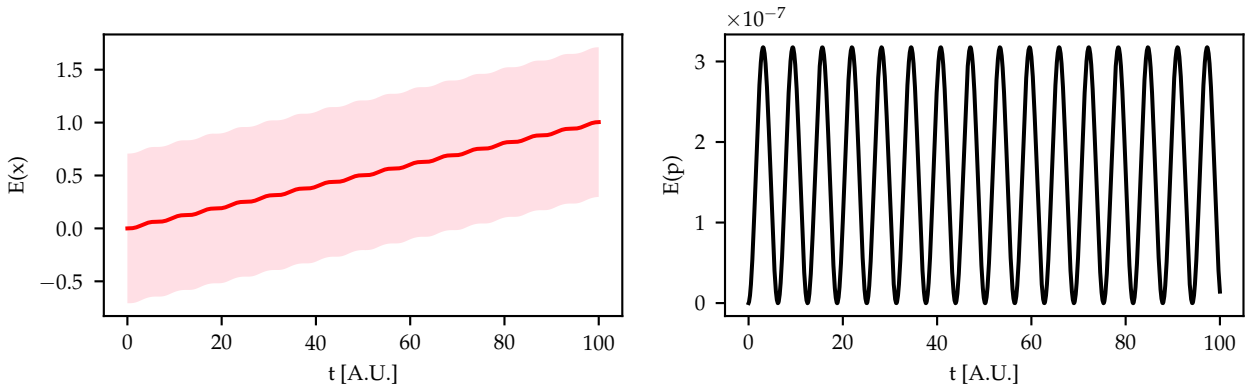

Figure 3: Evolution in time of the expected value of positions $E(x)$ and of momenta $E(p)$ for $T = 100$.

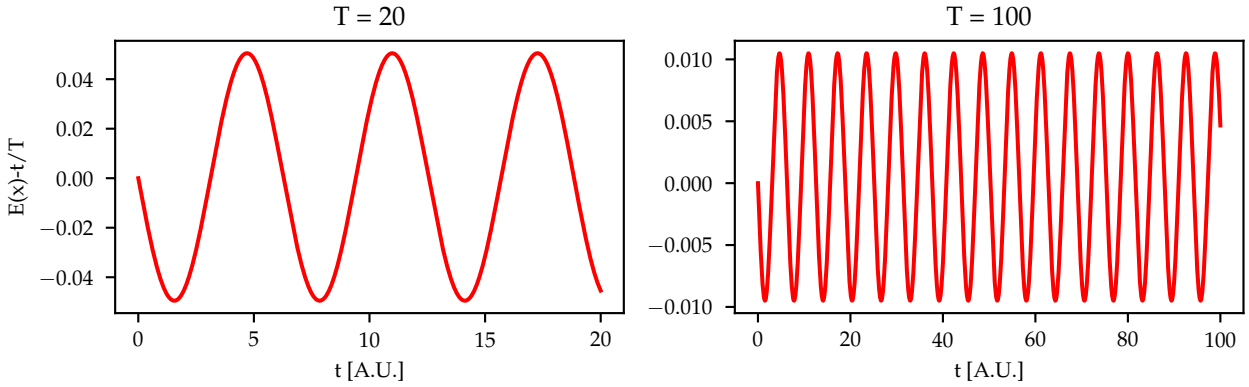

Figure 4: Oscillations of the average position, shifted by $-\frac{t}{T}$ compared for time evolutions with $T = 20$ and $T = 100$.

With this representation, it is immediately clear that the fluctuations are proportional to the opposite of a sinusoidal function, they are periodic with period 2π and have amplitude equal to $\frac{1}{T}$.

This tendency is not observed in the case of Figure 1: as a matter of facts, this is perfectly consistent with what it is expected, because $T = 1 < 2\pi$, hence it is impossible to observe even one complete oscillation.

It is also worth noticing that the periodical fluctuations of the momentum are, in order of magnitude, very small. However, oscillations with period of 2π are observed coherently also for $E(p)$.

Self evaluation

All the implementation analysed produced results consistent with the theory, so one can conclude that the program worked properly. However, some improvements to this study can be made.

For example, one could more quantitatively estimate the parameters that characterise the time evolution of the expected values and compare them with theory, for example by fitting the obtained curves.

Also, finding a more intriguing way of graphically representing time evolution of the wavefunction could be interesting.

Furthermore, to have more insight from a numerical point of view, one could also deepen their study of this problem by changing the space and momentum discretisation. In fact, as it is explained in Report for Homework 6, the discretisation can influence heavily the performances and results of the algorithm.