Time-dependent Schrödinger equation

1 Abstract

This time our aim is to evolve ground-state's harmonic oscillator considering time dependent potential $V(x,t)$.

2 Theory

Suppose that our system is described by time-dependent hamiltonian matrix H(t) and we want to know how a wavefunction $\varphi(x,t)$ evolves starting from instant t_0 .
In order to achieve this aim we can use time evolution operator $\hat{U}(t,t_0)$ from instant t_0 to generic instant t:

$$\hat{U}(t,t_0) = e^{-\frac{i}{\hbar}\int_{t_0}^{t}\hat{H(t)}dt}$$

or we can evolve the wavefunction from t_0 to $t_0+\tau$ using spilt operator method :

$$\hat{U}(t+\tau,t) \approx e^{-\frac{i}{\hbar}\hat{H}(t)\tau} = e^{-\frac{i}{\hbar}\frac{\hat{V}(x,t)}{2}\tau} e^{-\frac{i}{\hbar}\frac{\hat{P}^2}{2m}\tau} e^{-\frac{i}{\hbar}\frac{\hat{V}(x,t)}{2}\tau} + o(\tau^2)$$

and iterating this procedure ,time-step by time-step, until instant t is reached.

In the following we apply this last procedure using as hamiltonian

$$H(t) = \frac{p^2}{2m} + \frac{1}{2}m\omega^2\left(x - \frac{t}{T}\right)^2$$

In order to simplify calculus after first application $e^{-\frac{i}{\hbar}\frac{\hat{V}(x,t)}{2}\tau}$ we need *Fourier transform* to pass from position space to momentum one : in that base momentum operator $\hat{P}$ is diagonal with values p on diagonal!
So after $e^{-\frac{i}{\hbar}\frac{\hat{P}^2}{2m}\tau}$ we need to come back to position space using *inverse Fourier transform* and finally apply again $e^{-\frac{i}{\hbar}\frac{\hat{V}(x,t)}{2}\tau}$

3 Code development

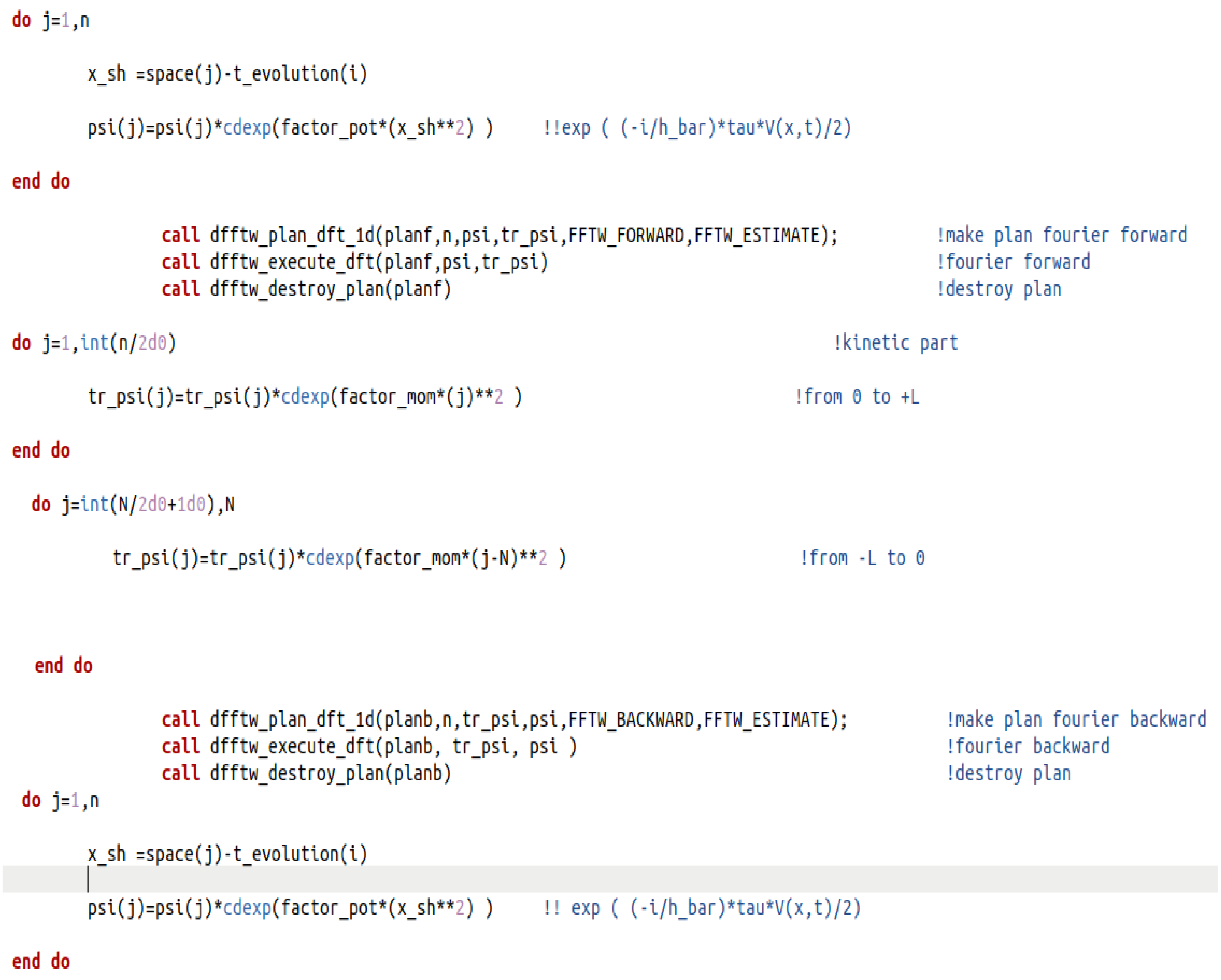
My code is composed by two units program : one *module* and one *main program.*

- MODULE ***harmonic oscillator time indipendent***:

  here we define harmonic oscillator time indipendent hamiltonian matrix and useful parameters like *integration interval* [-L : L] ,ω,*space resolution* ϵ , *time step* τ and T_{max}, we apply also DSYEV lapack function in order to compute ground state.

- PROGRAM ***main***:

  here i recall module defined above in order to start with groundstate eigenfunction and then compute ,for each time-step, wavefunction evolution until T_{max} is reached.



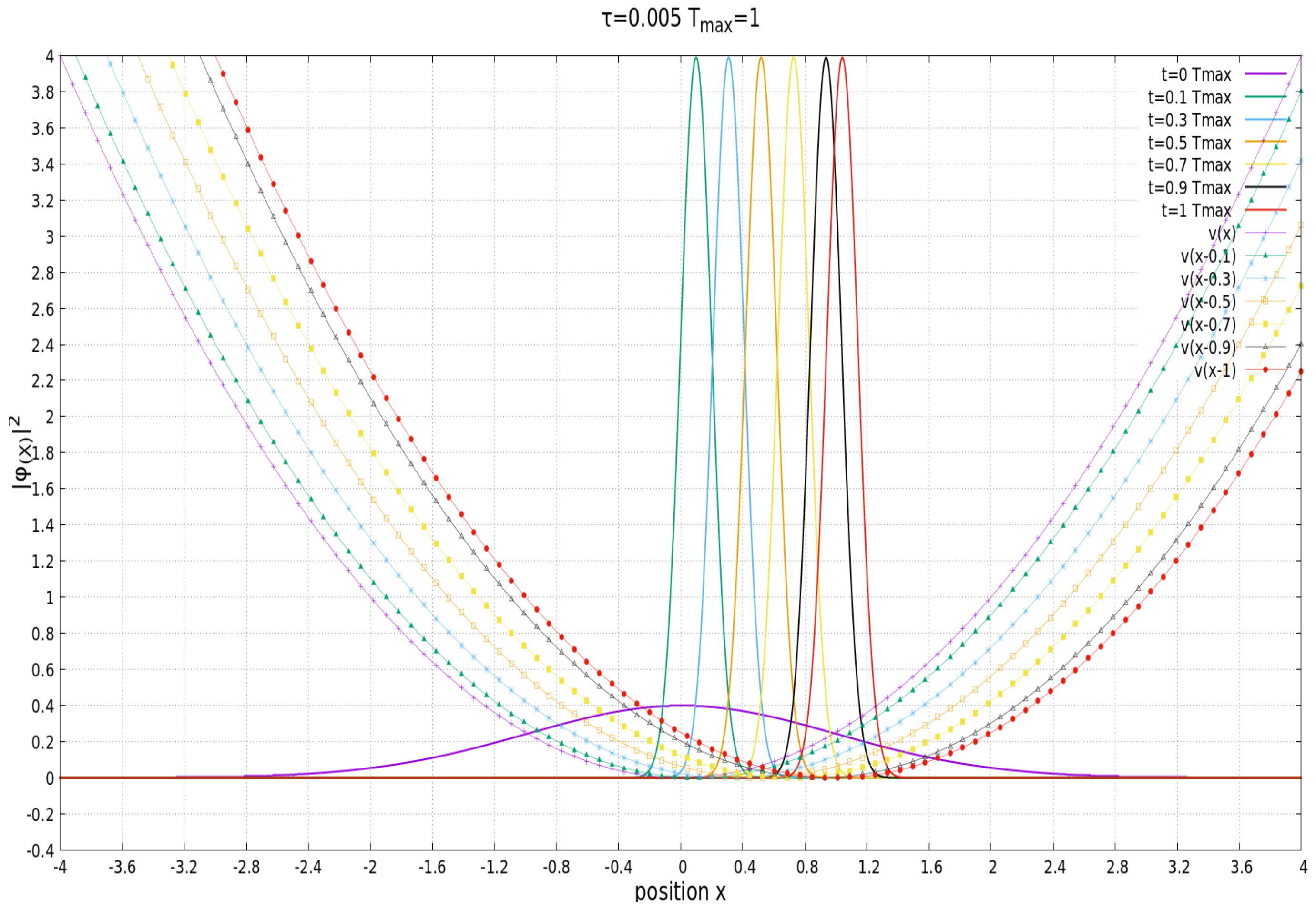
```
do j=1,n

       x_sh =space(j)-t_evolution(i)

       psi(j)=psi(j)*cdexp(factor_pot*(x_sh**2) )     !!exp ( (-i/h_bar)*tau*V(x,t)/2)

end do

              call dfftw_plan_dft_1d(planf,n,psi,tr_psi,FFTW_FORWARD,FFTW_ESTIMATE);         !make plan fourier forward
              call dfftw_execute_dft(planf,psi,tr_psi)                                        !fourier forward
              call dfftw_destroy_plan(planf)                                                  !destroy plan

do j=1,int(n/2d0)                                                             !kinetic part

       tr_psi(j)=tr_psi(j)*cdexp(factor_mom*(j)**2 )                        !from 0 to +L

end do

  do j=int(N/2d0+1d0),N

         tr_psi(j)=tr_psi(j)*cdexp(factor_mom*(j-N)**2 )                    !from -L to 0



  end do

              call dfftw_plan_dft_1d(planb,n,tr_psi,psi,FFTW_BACKWARD,FFTW_ESTIMATE);         !make plan fourier backward
              call dfftw_execute_dft(planb, tr_psi, psi )                                     !fourier backward
              call dfftw_destroy_plan(planb)                                                  !destroy plan
 do j=1,n

       x_sh =space(j)-t_evolution(i)

       psi(j)=psi(j)*cdexp(factor_pot*(x_sh**2) )     !! exp ( (-i/h_bar)*tau*V(x,t)/2)

end do
```

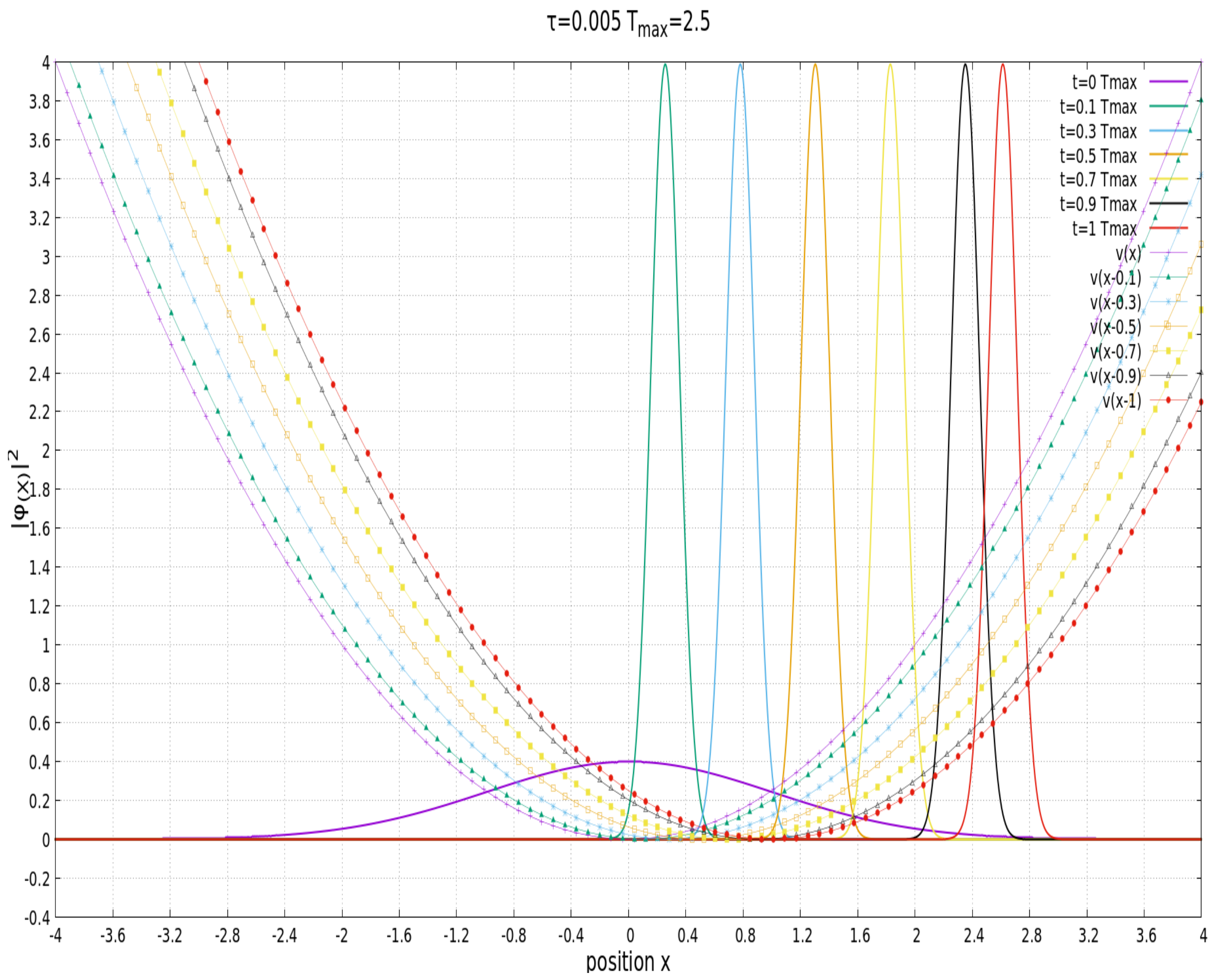
Moreover because of we apply both *Fourier transform* and *inverse* one, each input is multiplied by N and so after each time-step τ we need to normalize wavefunction (this appear in the code but not in figure). Finally we collect real part,imaginary part and the squared norm of wavefunction (for each time-step) and write them into three different files.

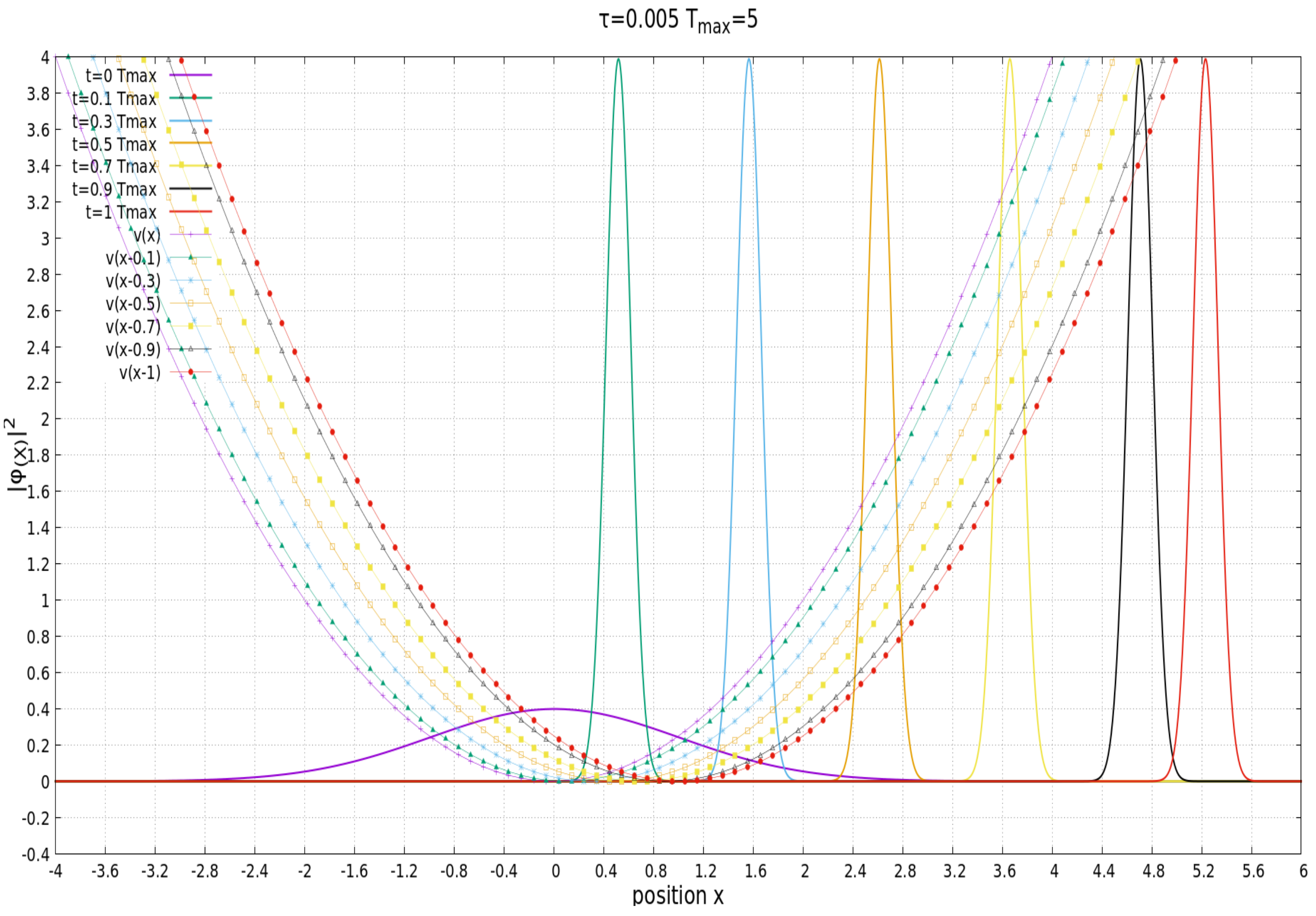
In each unit program we apply some checkpoints also related to time consuming for each operation done.

4 Results

In the following we'll show the wavefunction's squared norm and potential that evolve during intervals $[0 : T_{max}]$ for $T_{max} = 1, 2.5, 5$. Simulations are done using respectively $L = 3, 6, 7$, and always $\omega = 1, \epsilon = 0.002, \tau = 0.005$




τ=0.005 Tmax=2.5
|φ(x)|²
position x
t=0 Tmax
t=0.1 Tmax
t=0.3 Tmax
t=0.5 Tmax
t=0.7 Tmax
t=0.9 Tmax
t=1 Tmax
v(x)
v(x-0.1)
v(x-0.3)
v(x-0.5)
v(x-0.7)
v(x-0.9)
v(x-1)



Except for T_{max} =1 ,wavefunctions tend to anticipate its "respective potential" and this behaviours is explicitly clear looking graph above.
It is due to instability of the algorithm. Moreover, time evolution wavefunctions are less widespread and more picked respect to initial state.

5 Self-evaluation

This exercise is been challenging due to correct application of Fourier transform : it works in counter-intuitive way because store positive momenta until N/2 and then negative ones in *backwards* order.
Plotting real and imaginary parts for each wavefunction make confusion in the graphs and so i preferred to plot only the squared norms. Printing time-consuming we can conclude that code part most time consuming is linked to application DSYEV function.