WEEK No. 9 - Lab Session No. 2: Numerical Solution of 1D Schrödinger equation for a Gaussian wave packet.

The target of the simulation is explained first. Then, the numerical method is described. Its implementation (using any language you desire - FORTRAN, C, Python, Matlab....) is described in detail. Some programming work is required from you to complete it. Once answers have been found for all questions, send it back to me at: **raul.sanchez@uc3m.es**

Target of this Numerical Lab Session.-

Your objective will be to **simulate the temporal evolution of an electron whose wave function is described by a Gaussian wave packet as it collides with a finite square barrier**. The electron is coming from the left and interacts with a finite square barrier of height V_0. You will analyze the resulting interaction by integrating numerically the 1D time-dependent Schrödinger equation.

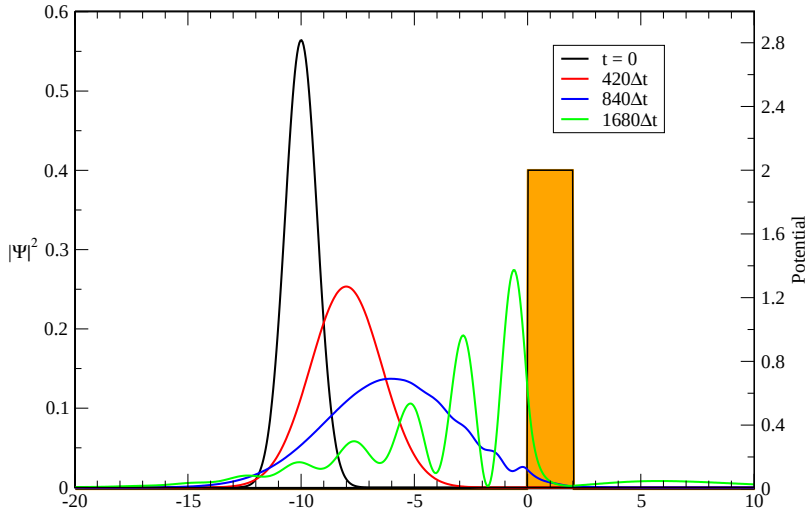

Figure 1: A Gaussian packet with $\sigma_0 = 1$ advances towards a finite square barrier. Both reflection and quantum tunnelling are apparent from the shape of the wavefunction ahead and beyond of the barrier at later times.

Description of the problem.-

The evolution of the Gaussian packet of interest is dictated by Schrödinger's equation,

$$i\hbar\frac{\partial\psi}{\partial t}(x,t) = -\frac{\hbar^2}{2m_e}\frac{\partial^2\psi}{\partial x^2}(x,t) + V(x)\psi(x,t), \tag{1}$$

starting from the initial condition,

$$\psi(x,0) = \left(\frac{1}{\pi\sigma_0^2}\right)^{1/4} e^{ik_0 x} e^{-\frac{(x-x_0)^2}{2\sigma_0^2}}. \tag{2}$$

This function corresponds to a Gaussian packet that, at time $t = 0$ is centered at x_0, has[1] a spatial width of the order of σ_0, an initial expected velocity $v_0 = \hbar k_0/m_e > 0$ and an expected energy of the packet is $E_0 = (\hbar^2/2m_e)\,(k_0^2 + 1/(2\sigma_0^2))$. The electron is, therefore, moving towards the right. In its path, it is going to encounter a finite square barrier, given by the potential,

$$V(x) = \begin{cases} 0, & x < 0 \\ V_0, & 0 < x < L \\ 0, & x > L \end{cases} \tag{3}$$

Objective: Integrate the problem numerically and show what happens when the particle encounters the barrier depending on the energy of the packet.

Normalization to atomic units.-

Before introducing Schrödinger's equation in any computer, one needs to make sure that we will not have to deal with either very large or very small numbers, that can exceed the computer precision and mess up the results. This is an issue here because there are quite a number of small numbers in Eq. 1. Indeed, quantities so small as $\hbar = 1.054 \times 10^{-34}$ J$\cdot$s, $m_e = 9.109 \times 10^{-31}$ kgr will cause large inaccuracies if introduced like that in a computer. For that reason, the first thing one needs to do is to move to a more adequate set of units. For Schrödinger equation, the so-called "**atomic units**" are particularly useful. These units are defined by the following transformations,

$$x \to \bar{x} \equiv \frac{x}{a_0}, \quad E \to \bar{E} \equiv \frac{E}{E_h}, \quad t \to \bar{t} \equiv \frac{t}{\hbar/E_h}, \tag{4}$$

with $a_0 = 4\pi\epsilon_0\hbar^2/(m_e e^2)$ (that corresponds to Bohr's radius) and $E_h = \hbar^2/(m_e a_0^2)$.

Basically, what this normalization means in practice is that **physical quantities are measured in multiples of these basic units**, more suited for phenomena that take place at the atomic scale. For example, time is measured in multiples of $\hbar/E_h = 2.42 \times 10^{-17}$sec, and lengths in multiples of the Bohr radius, that is of 5.29×10^{-11}m. Velocity, being the ratio of a length and a time, is measured in increments of $a_0(\hbar/E_h) \simeq 2.19\times10^6$m/s. Linear momentum, being the product of mass and velocity, is measured in increments of $m_e a_0(\hbar/E_h) \simeq 2 \times 10^{-24}$ kgr $\cdot$ m/s. For instance, when expressed in atomic units, the expected energy of our wave packet becomes $E_0 = (k_0^2 + 1/(2\sigma_0^2))/2$, with σ_0 measured in increments of a_0 and k_0 in increments of a_0^{-1}.

The resulting "normalized Schrödinger equation" is then,

$$i\frac{\partial\psi}{\partial\bar{t}}(\bar{x},\bar{t}) = -\frac{1}{2\bar{m}}\frac{\partial^2\psi}{\partial\bar{x}^2}(\bar{x},\bar{t}) + \bar{V}(\bar{x})\psi(\bar{x},\bar{t}), \tag{5}$$

[1] See the solution of the exercise proposed in class for Topic 7.

with $\bar{V} = V/E_h$ and $\bar{m} = m/m_e$. That is, with the potential and mass in terms of multiples of E_h and the electron mass. For instance, the mass of an electron would be $\bar{m}_e = 1$ in atomic units, while the mass of a proton is $\bar{m}_p = m_p/m_e = 1836.15$. As can be seen, there are no "small quantities" that can be seen explicitly any more, which should avoid any accuracy issues in a computer. The numerically solution of this equation is explained in what follows. We will however drop the tildes in what follows for the sake of simplicity. We will also assume that we are dealing with an electron, so that we can set $\bar{m} = 1$ and make things even simpler.

Discretization of the problem.-

To solve any differential equation numerically in a computer, one must consider both time and space as discrete variables that form a **mesh**. The process basically goes like this:

Time: it will be considered as a discrete variable running as,

$$t = 0,\ \Delta t,\ 2\Delta t, 3\Delta t, \cdots, (N-1)\Delta t \quad \text{or} \quad t_k = (k-1)\cdot\Delta t, \quad k = 1, 2, \cdots, N \tag{6}$$

Time starts at zero and increases in steps of Δt. The value of Δt will be chosen later on.

Space: Similarly, the one-dimensional real line is replaced by another discrete variable that runs between a minimum ($-\mathrm{x_{min}}$) and maximum value ($\mathrm{x_{max}}$) in Δx increments,

$$x = -x_{\mathrm{min}}, -x_{\mathrm{xmin}} + \Delta x, -x_{\mathrm{xmin}} + 2\Delta x, \cdots, x_{\mathrm{xmax}} - \Delta x, x_{\mathrm{xmax}}. \tag{7}$$

The number of points (M) in set by the value of Δx. The points in the mesh are:

$$x_j = -x_{\mathrm{min}} + (j-1)\cdot\Delta x, \quad j = 1, 2, \cdots, M, \quad \text{with} \quad \Delta x = \frac{x_{\mathrm{max}} + x_{\mathrm{xmin}}}{M-1}, \tag{8}$$

Remark: x_{min} and x_{max} must be chosen large enough so that the wave function is basically zero there for all the period of time of interest. Otherwise, boundary effects can distort the solution, since the packet maybe reflected back at the boundaries!

Wave function: The wave function that provides our solution to Schrödinger's equation is represented by a **complex matrix**. It is defined on the space-time mesh just introduced as:

$$\psi_j^k \equiv \psi(t_k, x_j), \quad k = 1, 2, \cdots, N, \quad j = 1, 2, \cdots, M. \tag{9}$$

Discretization of the derivatives.-

There are two derivatives in Schrödinger equation, a temporal one and a (second order) spatial one. Here, these derivatives are approximated following an approach known as **finite differences**[2].

[2] As a general rule, time derivatives are distinguished by placing a dot on top of the variable, and spatial derivatives by using a prime!

Time derivative of wave function: it is approximated as,

$$\frac{\partial \psi}{\partial t} \simeq \frac{\psi(t+\Delta t, x) - \psi(t,x)}{\Delta t} \rightarrow \quad \dot{\psi}_j^k = \frac{\psi_j^{k+1} - \psi_j^k}{\Delta t} \tag{10}$$

Therefore, $\dot{\psi}_j^k$ is the complex matrix that contains the time derivative of the wave function at position x_j and time t_k.

Space derivative of wave function: It is approximated as,

$$\frac{\partial^2 \psi}{\partial x^2} \simeq \frac{\psi(t, x+\Delta x) - 2\psi(t,x) + \psi(t, x-\Delta x)}{(\Delta x)^2} \rightarrow \quad \psi_j^{''k} = \frac{\psi_{j+1}^k - 2\psi_j^k + \psi_{j-1}^k}{(\Delta x)^2} \tag{11}$$

Therefore, $\psi_j^{''k}$ is the complex matrix that contains the second spatial derivative of the wave function at position x_j and time t_k.

Integration method (first "failed" attempt):
If the previous discretizations for the temporal and spatial derivatives of the wave function are used, the time-dependent 1D Schrödinger's equation becomes discretized as [3]:

$$\dot{\psi}_j^k = \frac{i}{2}\psi_j^{''k} - iV_j\psi_j^k, \tag{12}$$

where $V_j = V(x_j)$ is the discretized representation of the potential. By making all dependencies explicit, keeping on the left only those things depending on the future time t_{k+1} and moving to the right hand side everything else, it is found that,

$$\psi_j^{k+1} = \psi_j^k + \frac{i\Delta t}{(\Delta x)^2}\left(\psi_{j+1}^k - 2\psi_j^k + \psi_{j-1}^k\right) - 2i\Delta t V_j \psi_j^k. \tag{13}$$

This type of equation is what is needed. It permits to calculate the wave function at time t_{k+1} (at any desired location x_j) using as input the values of the wave function at the previous time, t_k. This type of method is called **explicit**. This particular implementation, known as **forward Euler method**, has a big problem, though. It is highly unstable! That is, any perturbation of the solution (introduced by the unavoidable truncation errors, for instance) ends up growing without bound which makes the solution obtained in this way worthless pretty quickly.

Integration method (second "good" attempt):
One way to go around this problem is by using the so-called **Crank-Nicholson** method. The change required is seemingly innocent! One just replaces the right hand side of Eq. 12 by its temporal average between t_k and t_{k+1}:

$$\dot{\psi}_j^k = \frac{i}{4}\left(\psi_j^{''k+1} + \psi_j^{''k}\right) - \frac{i}{2}V_j\left(\psi_j^{k+1} + \psi_j^k\right). \tag{14}$$

The spatial derivatives now contain information about the past **and the future** wave function. This type of methods are known as **implicit**, and are usually much more

[3] We will assume $m = 1$ in what follows!

stable. They are however a little bit more involved to advance. Indeed, by making the dependencies explicit and separating past and future information, you would get to,

$$\psi_j^{k+1} - \frac{i\Delta t}{4(\Delta x)^2}\left(\psi_{j+1}^{k+1} - 2\psi_j^{k+1} + \psi_{j-1}^{k+1}\right) + \frac{i}{2}\Delta t V_j \psi_j^{k+1} = \tag{15}$$

$$= \psi_j^k + \frac{i\Delta t}{4(\Delta x)^2}\left(\psi_{j+1}^k - 2\psi_j^k + \psi_{j-1}^k\right) - \frac{i}{2}\Delta t V_j \psi_j^k \tag{16}$$

This expression certainly looks more complicated than before. It becomes much clearer after introducing the **wavefunction vector**,

$$\Psi^k \equiv \begin{pmatrix} \psi_1^k \\ \psi_2^k \\ \vdots \\ \psi_M^k \end{pmatrix} \tag{17}$$

that contains the values of the wave function at all points at time t_k. Assuming now that $\Psi_0^k = \Psi_{M+1}^k = 0, \quad \forall k$ (these are the boundary conditions!), it is then possible to rewrite Eq. 15 as:

$$\mathbf{L} \cdot \Psi^{k+1} = \mathbf{R} \cdot \Psi^k \tag{18}$$

with the two matrices defined by the expressions:

$$\mathbf{L} \equiv \begin{pmatrix} \beta + i\Delta t V_1/2 & -i\alpha & 0 & 0 & 0 & 0 \\ -i\alpha & \beta + i\Delta t V_2/2 & -i\alpha & 0 & & \vdots \\ 0 & -i\alpha & \beta + i\Delta t V_3/2 & -i\alpha & 0 & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & -i\alpha & \beta + i\Delta t V_{M-1}/2 & -i\alpha \\ 0 & \cdots & \cdots & 0 & -i\alpha & \beta + i\Delta t V_M/2 \end{pmatrix}$$

and,

$$\mathbf{R} \equiv \begin{pmatrix} \beta^* - i\Delta t V_1/2 & i\alpha & 0 & 0 & 0 & 0 \\ i\alpha & \beta^* - i\Delta t V_2/2 & i\alpha & 0 & & \vdots \\ 0 & i\alpha & \beta^* - i\Delta t V_3/2 & i\alpha & 0 & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & i\alpha & \beta^* - i\Delta t V_{M-1}/2 & i\alpha \\ 0 & \cdots & \cdots & 0 & i\alpha & \beta^* - i\Delta t V_M/2 \end{pmatrix}$$

with $\alpha = \Delta t/4(\Delta x)^2$ and $\beta = 1 + 2i\alpha$. Clearly, $\mathbf{R} = \mathbf{L}^*$ in this case, due to the fact that the potential is real and does not change with time. They are also tridiagonal and symmetric complex matrices.

To find the new wavefunction vector Ψ^{k+1} one just needs to multiply first the current wavefunction vector Ψ^k by $\mathbf{R}$ from the left, and then solve Eq. 18, inverting matrix $\mathbf{L}$ in the process. Matrix inversion is typical of implicit problems. How can you do that numerically?

Solving tridiagonal matrix problems: So the question is how to solve numerically a problem of the type,

$$\begin{pmatrix} d_1 & a & 0 & 0 & 0 & 0 \\ a & d_2 & a & 0 & & \vdots \\ 0 & a & d_3 & a & 0 & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & a & d_{M-1} & a \\ 0 & \cdots & \cdots & 0 & a & d_M \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ \vdots \\ x_M \end{pmatrix} = \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ \vdots \\ \vdots \\ s_M \end{pmatrix} \tag{19}$$

Since the matrix is tridiagonal, it is not as complicated as it might appear at first glance. There is one method that is quite elegant and is very easy to implement numerically. It is given here without proof. It goes as follows. First, one needs to compute the **recursive** coefficients,

$$a'_i = \begin{cases} \dfrac{a}{d_1}, & i = 1 \\ \dfrac{a}{d_i - a \cdot a'_{i-1}}, & i = 2, 3, \cdots M-1 \end{cases} \tag{20}$$

and,

$$s'_i = \begin{cases} \dfrac{s_1}{d_1}, & i = 1 \\ \dfrac{s_i - a \cdot s'_{i-1}}{d_i - a \cdot a'_{i-1}}, & i = 2, 3, \cdots, M-1, M \end{cases} \tag{21}$$

As can be seen, the i-th coefficients for a and s' **DO depend** on the $(i-1)$-th ones, which means that one needs to iterate this in **increasing order** in i.
Once these coefficients have been calculated, one obtains the final solution easily by the following backsubstitution recursive rule,

$$x_i = \begin{cases} s'_M, & i = M \\ s'_i - a'_i \cdot x_{i+1}, & i = M-1, M-2, \cdots, 2, 1 \end{cases} \tag{22}$$

In contrast to the previous step, here one needs the $(i+1)$-th entry of the solution to calculate the i-th one. Therefore, you must iterate this rule in **decreasing order** in i.

Parameter values to be used in your computer code:

For the first run you should use $\sigma_0 = 3$ and $k_0 = 1$. As a result, the expected speed of the package will be about one in atomic units. The choice $\sigma_0 = 3$ is sufficiently large to give a rather spread-out package in space, which leads to a narrow shape function in wavenumber, as Heisenberg's principle states. In addition, you will use $\Delta t = 0.005$ and $\Delta x = 0.05$. With these choices[4], the wave packet will initially be covered by about sixty mesh points, and the time for the packet to advance the distance between two adjacent mesh points, given by $\Delta x/v_0 = \Delta x/k_0 = 0.05$ is ten times larger than Δt. You will also choose $-x_{\min} = -60$ and $x_{\max} = 60$, which means that the total number of points is $M = 1 + (x_{\max} + x_{\min})/\Delta x = 2401$. Then, you will set the width of the potential barrier to be $L = 2$, starting at $x = 0$. The initial location of the center of the wave packet will be chosen at $x_0 = -10$. In this way, the packet is sufficiently far from both boundaries as well as from the barrier [5], and there is ample space at both ends of the barrier to observe the interaction. You will also set the height of the barrier to $V_0 = 2$, so that the energy of the packet in atomic units, $E = (k_0^2 + 1/(2\sigma_0^2))/2 = 19/36$, is well below the top of the barrier. Finally, with a speed (in atomic units) of $v_0 = k_0 = 1$, the time needed for the center of the Gaussian packet to reach the barrier will be $t_{\rm int} \simeq 10/k_0 = 10$ in atomic time units. Since $\Delta t = 0.005$, that is about $N \sim t_{\rm int}/\Delta t = 2000$ iterations. The effects of the interaction will however apparent much earlier, since this is the time for the peak to reach the barrier, but the tail of the packet will be there much earlier. They will also extend for a much longer time. For all runs, you should set $t_{\rm final} = 50$, so that we have ample time to observe what is going on.

Parameter	value
Δt	0.005
$t_{\rm final}$	50
Δx	0.05
$x_{\min}$	-60
$x_{\max}$	60
k_0	1
σ_0	3
x_0	-10
L	2
V_0	2
M	2401

Table 1: Parameters values to be used for the first simulation.

[4]The next two conditions are important for numerical stability and accuracy reasons!

[5]We need that the initial condition provided by the Gaussian wave packet becomes essentially zero well before it reaches either the barrier and the boundaries. Had we picked, say, $\sigma_0 = 6$, then locating the center of the packet at -10 might be too close to the barrier. If we then move the packet at $x = -30$, it might be too close to the boundary. Can you figure out why?

Algorithm YOU MUST implement:

Using any software of your preference (i.e., FORTRAN90, C, Python, Matlab,....) write a code by yourself to simulate the process just described. You should follow these steps:

1. Define the parameters of the problem. Use all parameter values collected in Table 1.

2. Define vectors for the wave function.- You will need to define and keep in memory at least two vectors of size M for the state: the **current one** at time t_k (say, PSI_CURRENT) and the **next one** at time $t_k + \Delta t$ (say, PSI_NEXT). You do not need to keep information about any other previous state in memory. Therefore, these vectors can be overwritten as you advance the wavefunction in time.

3. Construct initial state vector.- At the beginning of the run you will have to start by filling with values the vector PSI_CURRENT. To do it, you just need to evaluate the provided initial condition (Eq. 2) on the points of the spatial mesh.

4. Build **L** and **R** matrices.- Since the matrices do not change with time (the potential is constant in time), you need to build them just once at the beginning of the run. Both matrices are tri-diagonal and symmetric. Furthermore, all the entries in the first supra- and sub-diagonals are the same in each matrix. As a result, you need to store in memory only the main diagonal (as a vector of size M) and the value of the first subdiagonal (as a scalar) for each matrix. You could name them, for instance, LMAT_DIAG, LMAT_SUBDIAG, RMAT_DIAG and RMAT_SUBDIAG.

5. Iterate the wave state for as many iterations as required.- you will have to advance the solution using Eq. 18 for as many iterations as needed. You will need to define first two vectors of size M, S_TRID and A_TRID. S_TRID contains the result of acting with **R** on the current state PSI_CURRENT. A_TRID is then calculated using Eq. 20, starting with the first entry. Then, you overwrite S_TRID using Eq. 21, also starting from the first entry, since you do not need the previous values any more. Finally, you obtain the new state and store it in PSI_NEXT using Eq. 22 starting with the M-th entry and then working our way downwards to 1!

6. Produce required output.- It is convenient to store in a file (or in several files) the values of the probability density vector (remember, it is $|\Psi(x,t)|^2$) on the spatial mesh at each (or, better, every so many) iteration(s). This file will be key in order to be able to graph the evolution of the packet and to perform the various calculations required. You should also include appropriate additional lines to calculate the quantities needed to complete the "**Results and Data Analysis**" section.

7. Monitoring the quality of the solution.- in order to check the correctness of your implementation, you should implement a quality diagnostic. You should use the value of the integral of $|\psi(x,t)|^2$, that should always be equal (or very close) to one. The integral of the probability density can be estimated on the discrete mesh by performing the following sum, at every time t_k,

$$I^k = \Delta x \cdot \left(\sum_{i=1}^{M} a_j \cdot |\Psi_j^k|^2 \right), \tag{23}$$

with $a_1 = a_M = 0.5$ and $a_j = 1$ for all other values of j. The value of this integral **at every iteration** should be written out to file to use for monitoring purposes!

Results and Data analysis.-

Once your code is ready and running, carry out the following tasks:

1. Produce a sequence of plots (or make a movie!) showing the probability density for the electron at least for the following times:

$$t = 0,\ 500\Delta t,\ 1000\Delta t,\ 1500\Delta t \text{ and } 2000\Delta t.$$

   Overlay these plots with another showing the potential so that you can see the relative position of barrier and packet. Rescale all plots so that you can easily see the effect of the barrier on the wave function. Compare your results with Fig. 2. If the agreement is good, you can then proceed to solve the next items!

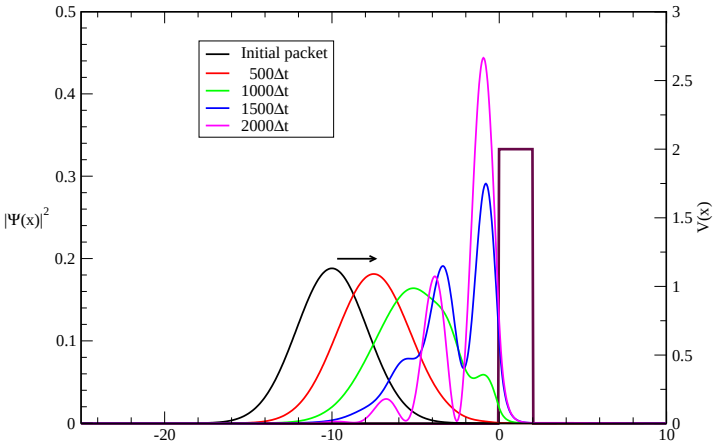

Figure 2: Ilustration of the target simulation. Snapshots of the Gaussian packet with $\sigma_0 = 3$ as it advances towards a finite square barrier at selected times.

2. Calculate the probability T of finding the electron beyond the barrier as a function of time for the previous run. *You can estimate it via the same sum as in Eq. 23, but restricting the sum to the mesh positions to the right of the end of the barrier.* Produce a plot showing the time dependence of the estimated transmission probability T and find its asymptotic value as shown in Fig. 3.

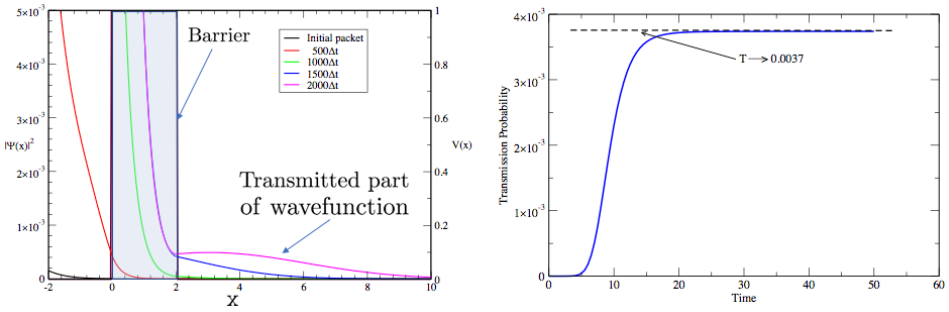

Figure 3: **Left:** zoom-in of the transmission region beyond the barrier. **Right:** resulting transmission probability $T(t)$ with asymptotic value shown with a dashed line.

3. **Study the dependence of the (asymptotic) transmission probability on k_0.-** For this study, you need to carry out a series of simulations for various values of k_0 varying from 0.5 to 8. All other parameters remain with the same values collected in Table 1. Use at least fifteen different values for k_0. An example of what you should get is shown in Fig. 4.

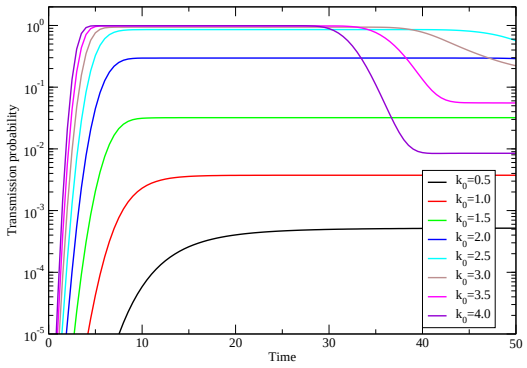

Figure 4: Transmission probabilities as a function of time for various values of k_0.

By the way, can you explain why, at the largest values of k_0, a drop in the transmission probability is observed? Help: Remember that we are estimating it as the integral of the wavefunction beyond the barrier! Why is the drop not observed for the lower k_0? Would it be also observed if we wait for a longer time?

4. **Study the dependence of the (asymptotic) transmission probability on the energy E_0.-** Plot the values of the transmission probabilities obtained in the previous section as a function of $E_0 = (k_0^2 + 1/(2\sigma_0^2))/2$. The plot should look like Fig. 5, You should also throw in the theoretical formula (derived in class!) for the

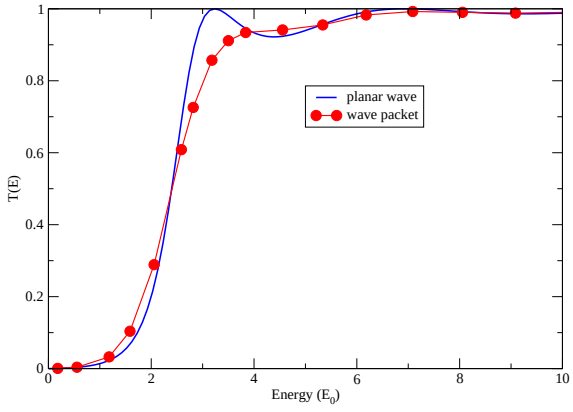

Figure 5: Transmission probabilities as a function of the expected energy E_0. In blue, the theoretical transmission coefficient for a plane wave of the same energy is shown.

transmission coefficient as a function of the energy of a plane wave with energy E, that in atomic units is:

$$T(E) = \begin{cases} \left[1 + \dfrac{V_0^2}{4E(V_0 - E)} \sinh^2\left(L\sqrt{2(V_0 - E)}\right)\right]^{-1} & E < V_0 \\ \left[1 + \dfrac{V_0^2}{4E(E - V_0)} \sin^2\left(L\sqrt{2(E - V_0)}\right)\right]^{-1} & E > V_0 \end{cases} \tag{24}$$

The oscillations on the transmission coefficient for the theoretical formula are clear in the plot, with the maxima showing where resonant transmission takes place. For the wave packet case, can you observe any oscillations? Are they as clear as the one for the plane wave solution? If not, argue a possible reason. Can you propose how to change the parameters of our simulation to reproduce the oscillations of the theoretical formula?

IMPORTANT:

You need to write a report in which all the requested tasks and plots are included and discussed, and all proposed questions are answered. You MUST include a file with the implementation of your code so that I can take a look at it. Once all is completed, upload it into the **assignment link** that will be ready in Aula Global.

Recommended date for upload: **December 22, 2020**.

Final date for upload: **January 10, 2021 @12:00AM**.