Schrödinger Partial Differential Equations

Numerical solutions and plots

Alejandro Galván Daniel Quiles Bernat Ramis

December 2019

1 Introduction

1.1 Objective

The aim of this work is to investigate the applicability of numerical methods to solve the Schrödinger partial differential equations. We started by developing two discretized versions of the one-dimensional equation: the first one according to Euler's method, and the second one with the more stable Crank-Nicolson method. Later, we also deduced the Crank-Nicolson equations for the case of two spatial dimensions.

After implementing these methods in Matlab, we began modelling some simple quantum systems. As we obtained simulations that perfectly matched the theory behind them, we moved on to more difficult or unconventional systems in the hopes of finding more interesting results, as well as discovering the strengths and weaknesses of numerical PDE solving.

1.2 PDE's

Partial differential equations are a powerful tool to describe and predict the behaviour of a system underlying a study. They provide the system instructions about how to change over time, space or any other variable.

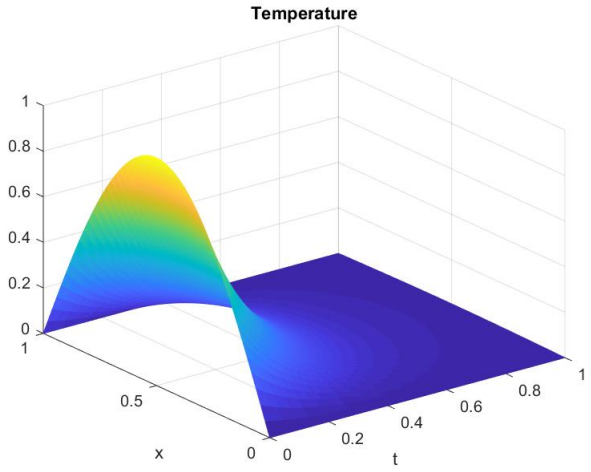
Usually, in order to get that intuition behind PDE's, the heat propagation equation is given as an example.

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u$$

To simplify, the same PDE for one dimension is

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$$

where α is a proportionality constant. The intuition is that the more temperature difference of a given point compared with its surroundings (the more 'curved' the temperature function is), the more the temperature at that point varies over time. We built a plot of the situation.



The figure depicts a hot 1-dimensional object (that is hotter in points near to its centre) and how it cools down, as it is in contact with a cooler source.

Schrödinger's equation is not as simple as this one, thus less intuitive: it involves complex numbers and more funcion terms, however we hope this example illustrates that a PDE leads the behaviour of a system.

1.3 Natural Units

Natural units were originally proposed by George Johnstone Stoney, in 1881, when he noticed that electric charge is quantized, and derived units of length, time, and mass by normalizing G, c, and the electron charge, e, to 1.

They are known as natural units because they are defined exclusively in terms of universal physical constants, such that these constants take the numerical value of 1 when expressed in terms of these units.

These units come in handy because they simplify a lot the computations as we get rid of many constants (we normalise them to 1). Our main reason to use them was that, with the international system, the value of $\hbar$ is so small that we cannot work with it numerically.

However, we still found problems when trying to use plausible values for the different measurements (for example, time units became way too small). As the aim of this project is to treat and study quantum systems qualitatively, the magnitudes used are, by no means, scaled to reality. For us, it is more important the proportions and shapes of the solutions and their connection to theory.

2 Infinite potential well

2.1 One particle

In this section we are going to analyse a particle confined in a box, and we are going to compare the solutions with both methods, to check its stability.

This model describes a particle moving freely in a small space surrounded by impenetrable barriers, so that the particle can only be found inside the box. It is a very useful example to compare the result we would obtain in classical and quantum systems.

As we saw in class, any eigenfunction must satisfy some conditions: (i) Be singled valued (ii) Be finite (iii) Be continuous (iv) Have the first partial derivative with respect to space continuous (be of class $\mathcal{C}^1$).

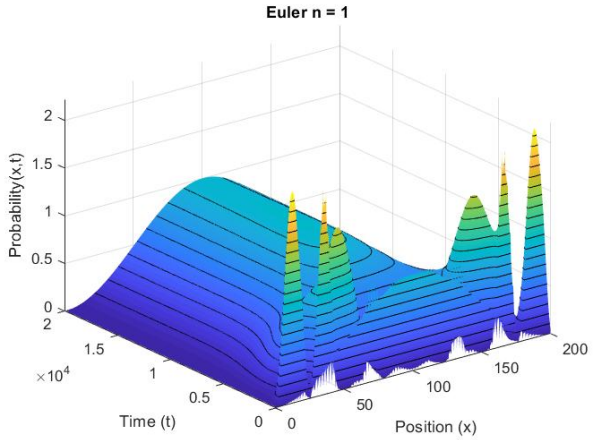
In this case the particle is confined in the box, so the eigenfunction outside the box must be 0, and to satisfy (i), it must be 0 also in the limits of the box.

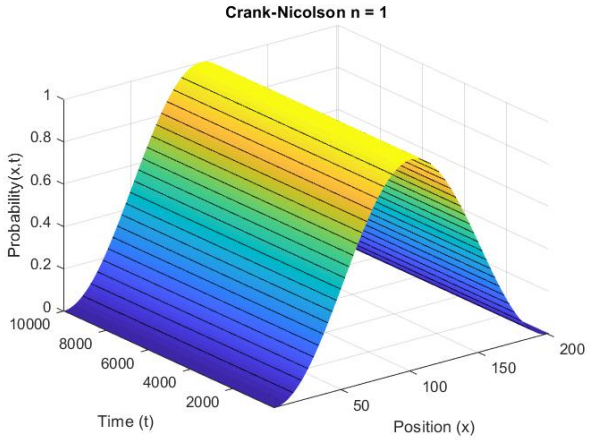
A general free particle in 1 dimension has an equation $\Psi(x) = Ae^{ikx} + Be^{-ikx}$, which is the addition of the two possibilities, particle travelling right, and particle travelling left.

So that imposing the previous conditions, we get that $\Psi = Csin(kx)$ where C is the normalization constant and k is quantized, implying that energy is also quantized.

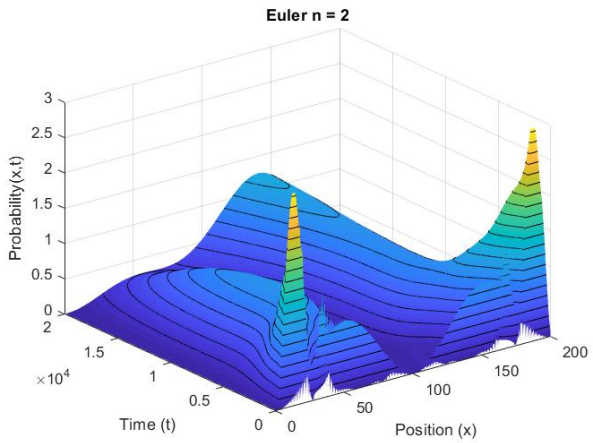
$$K_n = \frac{\sqrt{2mE_n}}{\hbar} \qquad\qquad E_n = \frac{n^2\pi^2\hbar^2}{2ma^2}$$

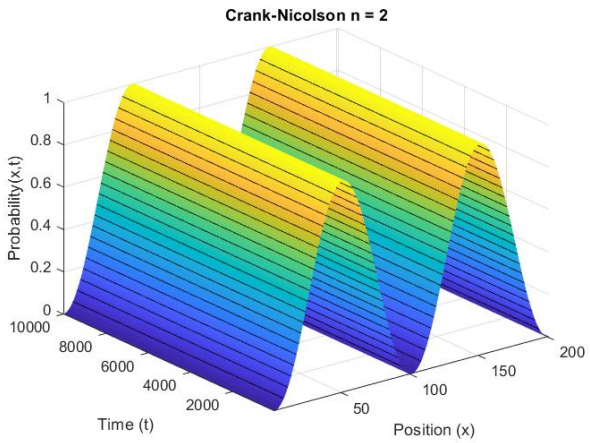
In this model, since we have analytical solutions, it is easy to compare the stability of both methods. In both cases, we will introduce as initial conditions the analytic solution we obtained above, and then we are going to check its stability over time.





As it can be seen comparing both plots, Euler's method is much more unstable than Crank-Nicolson. In case n = 1, the first one takes time to stabilize despite having started with the stable solution. However, in other cases such as n = 2, it seems to reach a stable solution, but with the probability associated with the case n = 1, in general. That shows that it is not a very robust method.





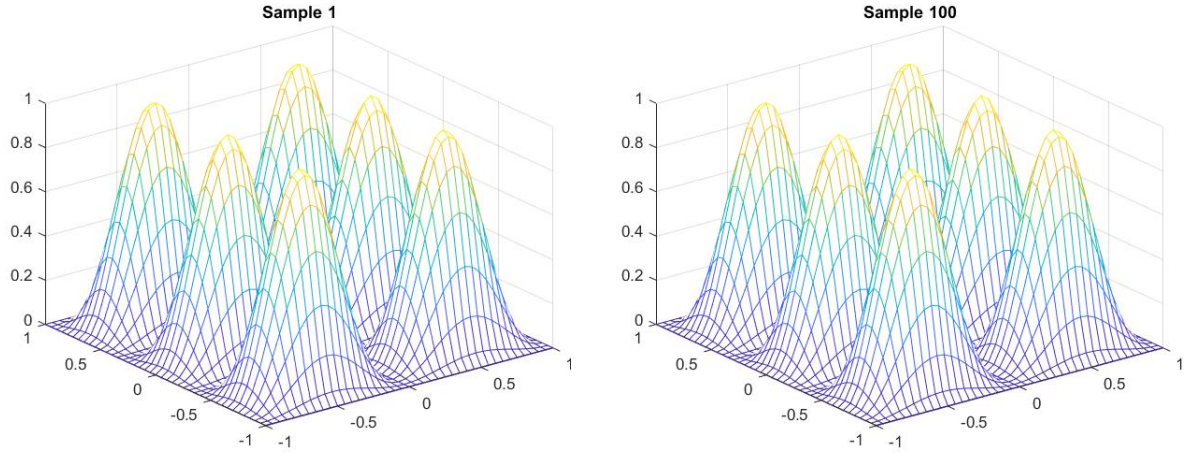
Instead, Crank-Nicolson's method is perfectly stable for those initial conditions, for n = 1 and for the

other possible energy states.

Nevertheless, we are going to use both methods in some other cases. Crank-Nicolson is more stable but slower, because it solves a system of equations in each iteration. Even, sometimes it is necessary to reduce precision in order to avoid long periods of the program run.

2.2 Two particles

This is an analogous situation to the previous one, but considering two particles instead of one. In the first case, we are going to consider that they are distinguishable particles, and for that reason $\Psi(x_1, x_2) = \Psi_\alpha(x_1)\Psi_\beta(x_2)$ where α and β are the quantum states. In this case, as in the infinite potential well with one particle, we know the analytic eigenfunction, so our intention is to check the stability of Crank-Nicolson 2D, in order to see if it is precise enough to use it in other cases where we do not know the analytic solution to the Schrödinger Equation.



As it can be observed in the plots, the probability function is stable over time. However, it loses a bit of definition because the numerical method solves a 3D system, with 2 dimensions for the position, and one for the time. If we use Δx and Δt very small, the program exceeds the allowed maximum capacity for storage.

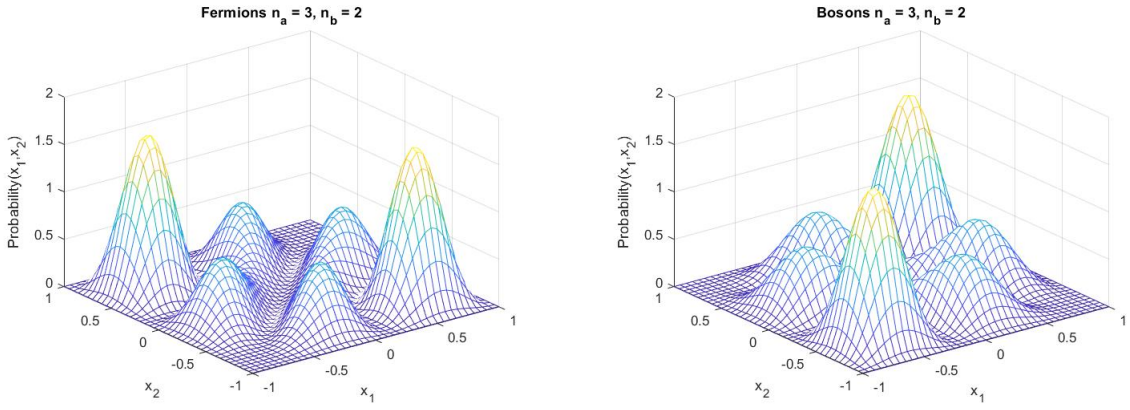
It is also interesting to highlight that plots for two distinguishable particles in a one dimensional box, can also be thought of as the plots of the probability function of a particle in a two dimensional box; with an energy state associated to one axis and one to the other one.

Now, in order to assure the stability of the method, we are going to analyse its stability with some different initial conditions. In these cases we are going to think of two indistinguishable particles in a one dimensional box.

As we have studied, in order to preserve its indistinguishability, it is necessary to consider the case where particle 1 has the quantum state α and particle 2 the quantum state β, so $\Psi = \Psi_\alpha(r_1)\Psi_\beta(r_2)$ and the case where particle 1 has the quantum state β and particle 2 the quantum state α and so $\Psi = \Psi_\beta(r_1)\Psi_\alpha(r_2)$. As we know that eigenfunctions are linear, a reasonable option is to either add both functions or subtract one from the other. It is easy to check that imposing this eigenfunction, indistinguishability is preserved.

To sum up, we get $\Psi = \frac{1}{\sqrt{2}}\left[\Psi_\alpha(r_1)\Psi_\beta(r_2) \pm \Psi_\beta(r_1)\Psi_\alpha(r_2)\right]$ which is the eigenfunction that describes two fundamental particles. Bosons, such as photons, are described by the symmetric function (+), since there can be found two particles with the same quantum state. Fermions, such as electrons, are described by the antisymmetric function (-), since two particles of this kind can not be found with the same quantum

state.



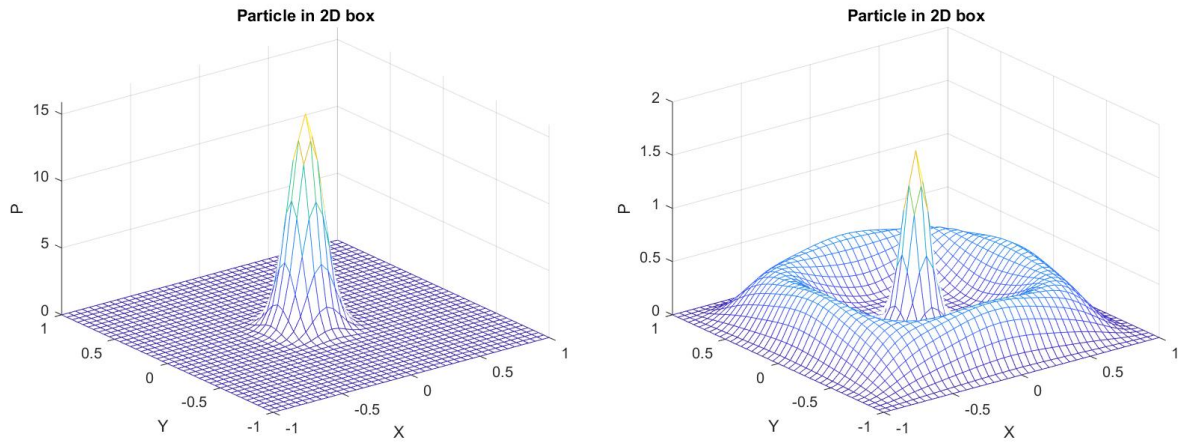
As it can be seen in these plots with the same quantum states, each elementary particle has its different probability function and, in order not to fill the space with repeated plots, we have only added one example of each one, but the program is stable along time in many other cases. It is worth noticing that, in the case of fermions, the probability of finding the two particles on the same spot is zero, while for bosons it is perfectly allowed.

2.3 One particle in a 2D box

As we have seen in the previous section, the plots can also be thought of as one particle in a 2D box. In this section, we are going to think of a particle in a 2 dimensional box, and we are going to change the initial conditions to the ones for a quite localized particle at x = 0; and with an initial velocity that may be modified at our own will.

As initial conditions, we have used $\psi(x) = \frac{1}{\sqrt{0.1\sqrt{\pi}}} e^{ikx} e^{\frac{-x^2}{0.04}}$, which we obtained from the article [3], and created a 2-dimensional version of them: $\Psi(x_1, x_2) = \psi(x_1)\psi(x_2)$

In this case we are going to plot the results for k = 0.



The thing that caught our attention at first was that in the very first iterations, some kind of central waves are formed. The expected behaviour was that the probability function got flatter uniformly. We

figured out that this phenomena is due to the relatively big value of δt (for smaller values *Matlab* did not run), which means that a part of the wave function has enough time to reach the wall and bounce back, causing interference. In an infinite domain, the waves should not be formed, which is what we observed in some 1D enviroments.

As time passes by, gorgeous patterns take shape because of the reflection of the wave against the wall.

3 Step potential

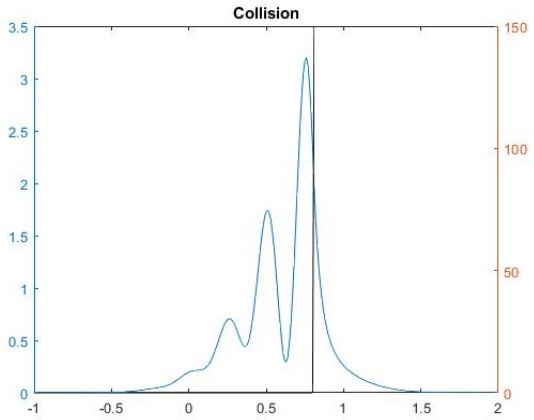
In the following section, we will try to simulate the behaviour of an electron which collides against a step potential. An analytical study of the time-independent problem yields:

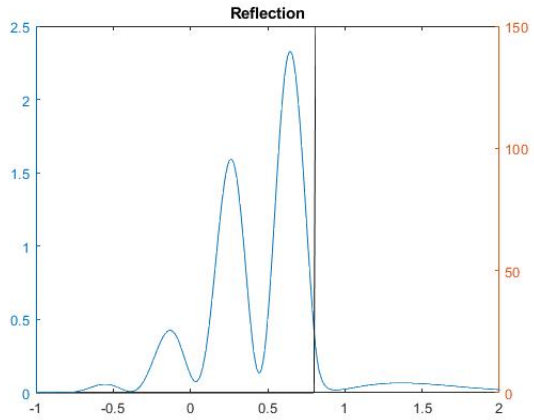
$$\psi(x) = \begin{cases} \frac{D}{2}(1 + ik_2/k_1)e^{ik_1 x} + \frac{D}{2}(1 - ik_2/k_1)e^{-ik_1 x} & x \leq 0 \\ De^{-k_2 x} & x \geq 0 \end{cases}$$

for a potential 0 at $x < 0$ and V_0 at $x > 0$, and $E < V_0$. For the case $E > V_0$, we get

$$\psi(x) = \begin{cases} Ae^{ik_1 x} + A\frac{k_1 - k_2}{k_1 + k_2}e^{-ik_1 x} & x \leq 0 \\ A\frac{2k_1}{k_1 + k_2}e^{-k_2 x} & x \geq 0 \end{cases}$$

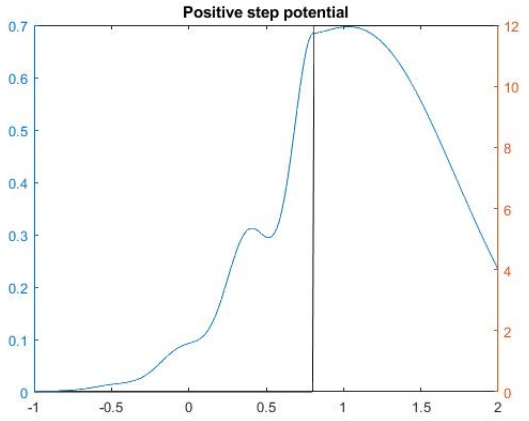
Since we wanted to see if our numerical methods could coincide with the theoretical results, we solved the problem with Crank Nicolson and the same initial wave function as in the previous case, which represents a quite concentrated travelling wave. In the following graphs, we see the probability function for two different times and $E < V_0$:

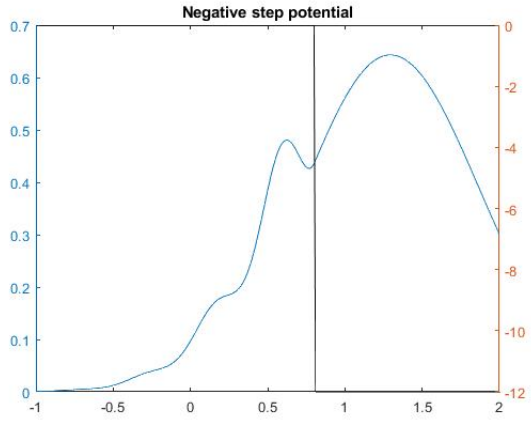




In the first figure, we see how the wave collides and a little bit of it penetrates. If one takes the analytical solutions mentioned above and calculates the probability function for $x \geq 0$, the result is a decreasing exponential, in agreement with our plot. In the second figure, we see that, after a while, the probability of finding a particle that has surpassed the step potential is disappearing. Provided that the theory predicts that the transmission coefficient is zero, this seems to be a satisfactory result.

On the other hand, we have the case $E > V_0$. For this scenario, we wanted to check a very shocking fact: the transmission and reflection coefficients are the same independently of the sign of V_0. That is, the chances of the particle being transmitted are the same for a V_0 step as for a $-V_0$ step. In order to check it, we calculated the integral of the probability function on the left hand side of the discontinuity and on the right hand side. These are plots of the positive and the negative step potentials, at the same moment in time after the wave runs into the wall:





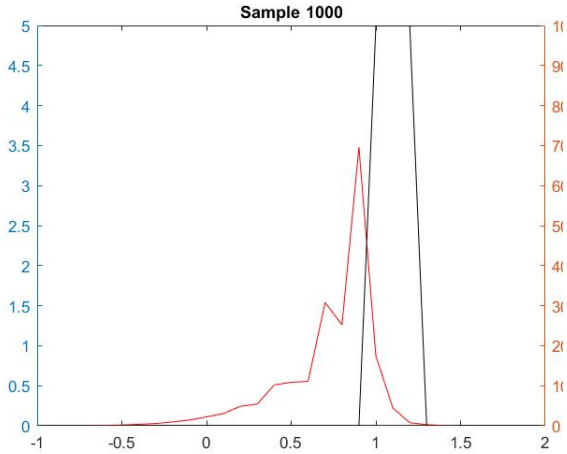
We found that, at the time the plots were taken, the integrals of the probability on each side were 26.38% and 73.62% for a V_0 step and 25.7% and 74.3% for a $-V_0$ step. Thus, our results match the theory: the transmission coefficients we obtain are almost the same independently of the sign of V_0. The wave gets partially reflected not because the potential grows, but because there is a discontinuity.

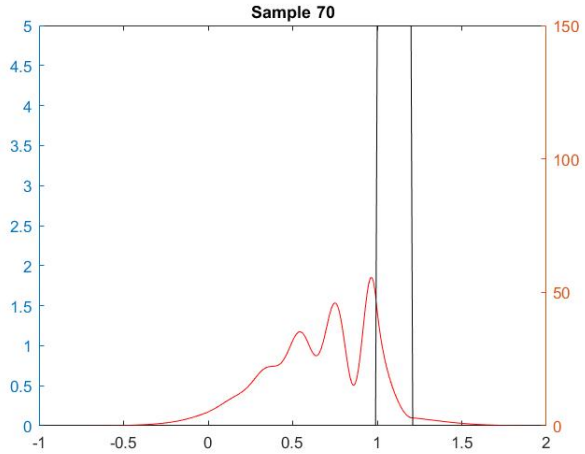
4 Barrier potential

4.1 Time independent

This case we are going to study the behaviour of a particle that arrives to a region with a barrier potential. It is essentially quite similar to the step potential, but composing two step potentials.

As we did in the previous case, we are going to analyse a particle whose eigenfunction is quite centred in x = 0; and with an initial velocity that can be modified, and the idea is to compare Euler's and C-N methods between them, and compare the results with the ones obtained using the TISE.



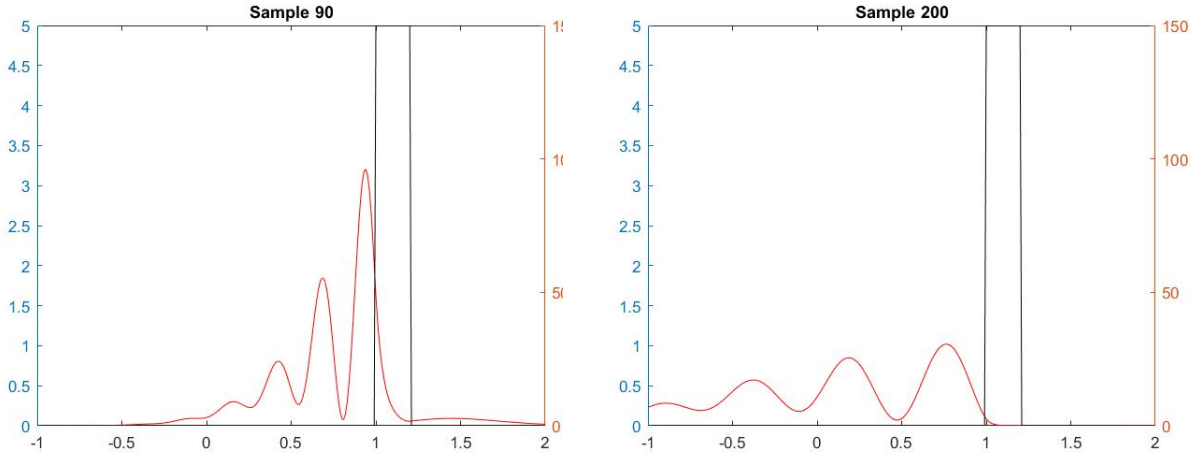


It can be observed comparing both methods that Euler's method (left plot) and Crank-Nicolson (right plot) are qualitatively similar, but the second one gets a better accuracy. The main reason for that loss of definition is that in the iterative process low order numers like Δx^2 appear dividing. In this context, the

C-N method is more robust than Euler's, therefore better at reducing numerical instability. This implies that in Euler's method we obtain nonsensical results when we implement Δx too small.

To explain this fact, we recall that at each iteration, Euler's method provides the value of Ψ_i^{n+1} using just one equation for it, whereas C-N solves the whole system of equations, finding the value of Ψ_i^{n+1} in three different equations of the system.

So, to analyse the results we are going to use the ones obtained with C-N. We have selected some of the plots to analyse the interesting facts.



With these two plots we can see the probability of finding the particle over time. The first conclusion we can extract is that the particle gets more delocalized along time, which is the result we expected. Besides, as we saw in class, the particle has a chance to cross the barrier in spite of having less energy than the potential barrier (it can be transmitted by tunnel effect).

The second plot shows how the part of the probability function that crossed the barrier has already travelled away towards the right, whereas the part that got reflected has bounced back and heads towards the left.

4.2 Time dependent

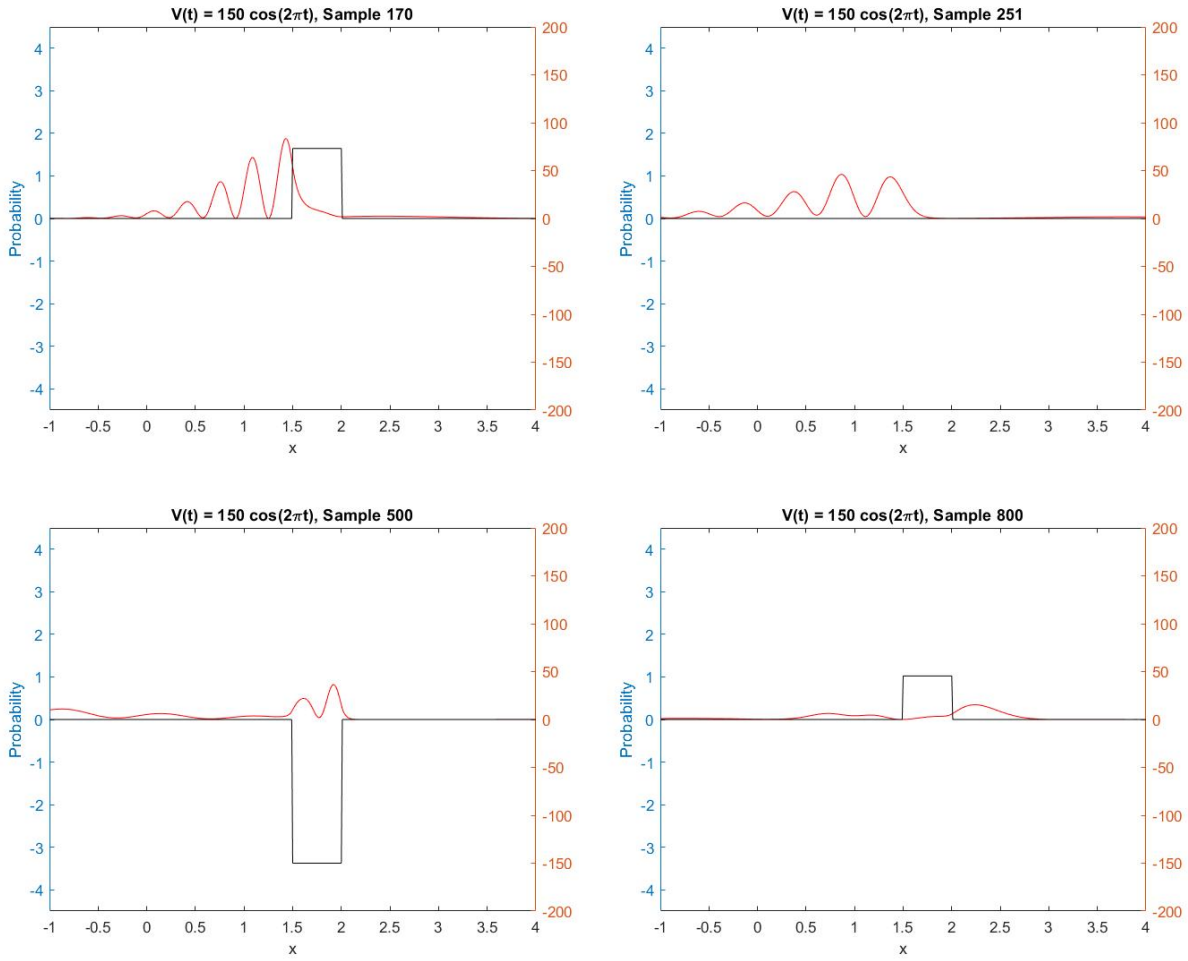
This is an analogous situation, but with a time dependent potential barrier. We thought that it was an interesting situation, as it was a case that we could not study soving the TISE analitically. We have implemented it directly with C-N as it was the method that offered better results in the previous case.

We have used a barrier potential variable in time with a cosine function, although the program could be easily modified to study different time dependent potential functions.

When the potential is high, the particle will probably get reflected, although there is a certain probability that it will pass through quantum tunneling.

When the barrier potential is near to 0, the particle can cross as if it was a free particle, but then, when the potential decreases, the portion of probability trapped "inside" the barrier, behaves like in the finite potential well case, with a lower probability near the limits of the well, but non zero. This happens because it may penetrate as in the step potential.

In conclusion, the results obtained by the numerical method are in good agreement with what we would expect applying the analytical solution used in constant potential problems.



5 Potential generated by a charged particle

We wanted to experiment with a potential more complicated than just a finite sum of heavyside functions, because it's something that is tough to treat analytically. The idea is to simulate the interaction between a proton and an electron.

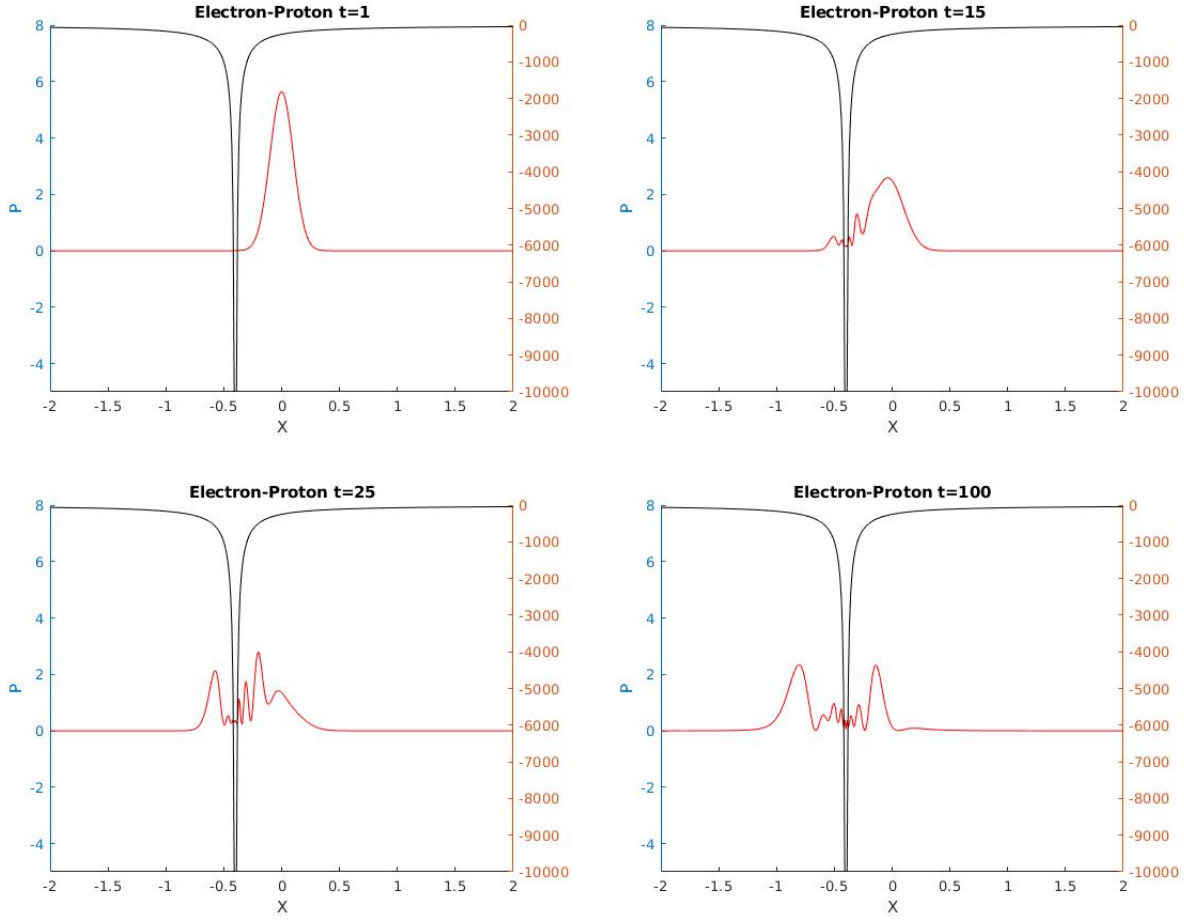
Let's suppose that the proton is static (it gets no acceleration because of the electron) and that the electron is deployed somewhere near the proton. Of course this situation does not fully fit reality, since we are restricting the study to one dimension. Nevertheless, this experiment has some interesting points to be analysed.

With the corresponding modifications, our program generates the plots shown in the next page. We also get simulations, although they can't be included in this document.

In the pictures below, one can build an idea about how the probability of finding the electron near the proton evolves over time. As we are only considering the electric interaction, it is clear that the electron tends to stay close to the proton by charge attraction.

The 3D plot shows the evolution through time of the probability over the space.

Let's see what happens if we consider the same situation but replacing the proton for a motionless negative particle. All in all, the potential changes sign.



As expected, the probability of one of the electrons runs away from the location of the other electron. Picking long-term plots gives no relevant information, as our X axis if finite and acts just like a confining box.

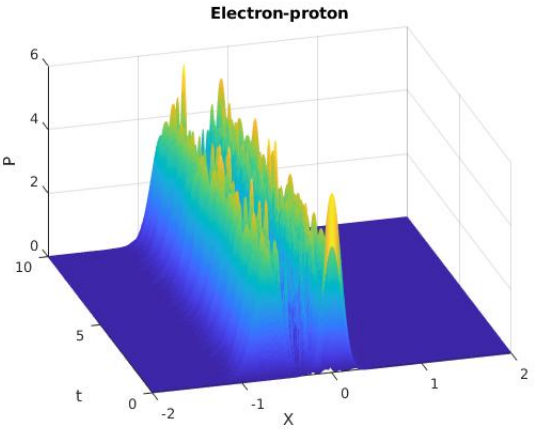
Surely, the situation changes significantly for different magnitudes of the potential. Here we plot some relevant cases.

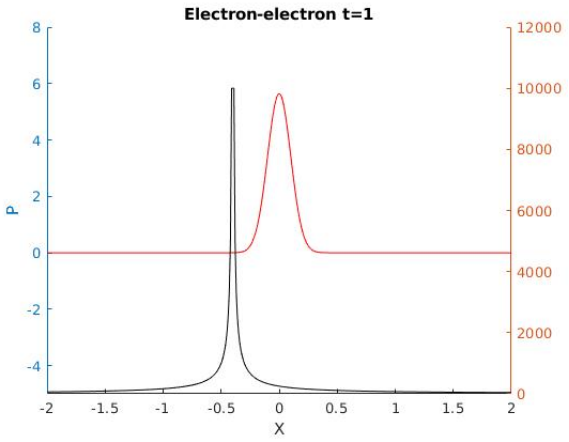
Even though we keep thinking of a pair electron-proton, this is just to have an idea of the situation. Obviously, changing the potential function would imply having different particles.

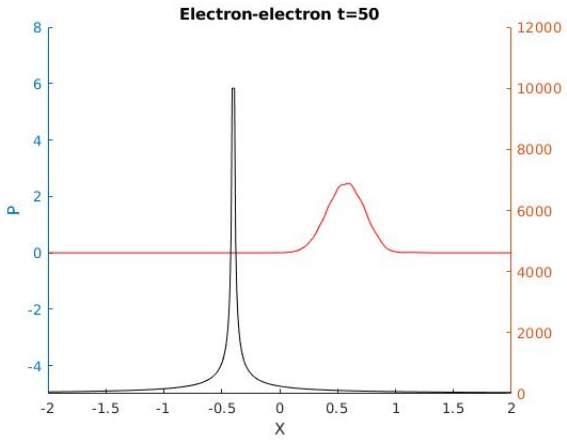
An interesting phenomenon caught our attention in the case of a relatively "deep" potential. We observed that after the probability associated to the electron approaches location of the proton and it doesn't have enough energy to get through, it bounces back some distance. It is then when it feels once again forced to move towards lower potentials, hitting again the peak of potential to periodically repeat a similar behaviour.

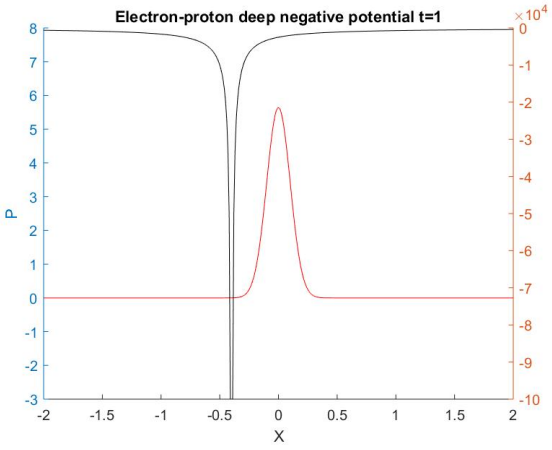
Focusing on the case of a less influent negative potential, we see that the probability function presents two clear maxima: one around the initial position (the electron would tend to stay at rest, without being lured) and another around the depression (the electron would have been lured).

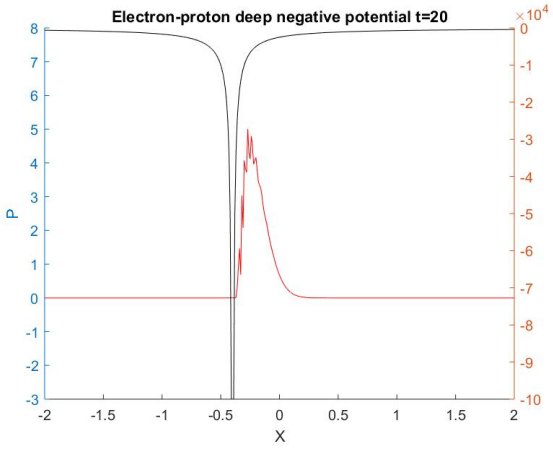
These situations, specially the ones with considerable potential, help us undestand why in some cases approximating potentials due to particles (such as atoms) to rectangular potential barriers is not that crazy, because for the right potentials the behaviour of the probability density is not that different.

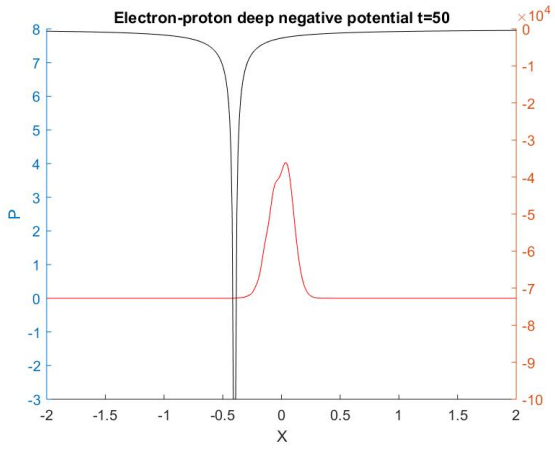

Electron-proton
P
t
X

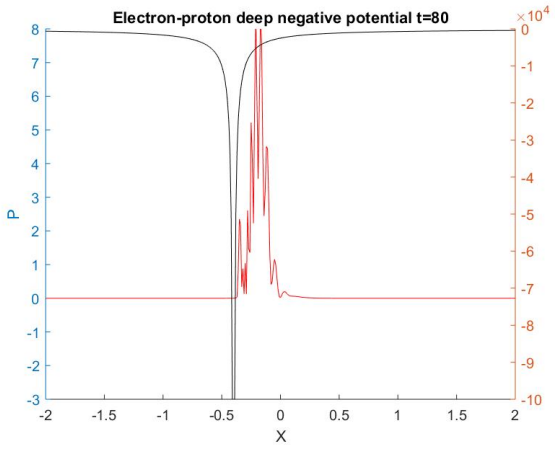

Electron-electron t=1
P
X

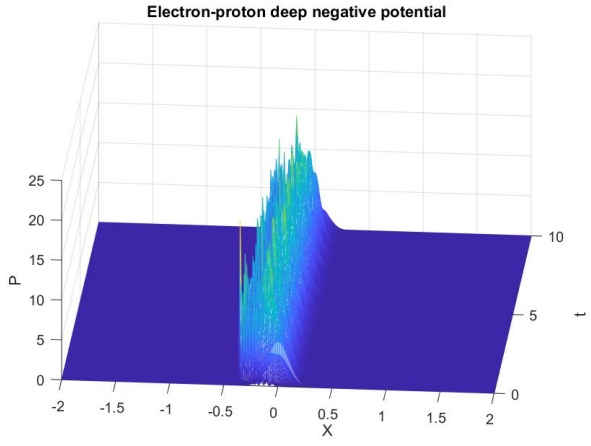

Electron-electron t=50
P
X

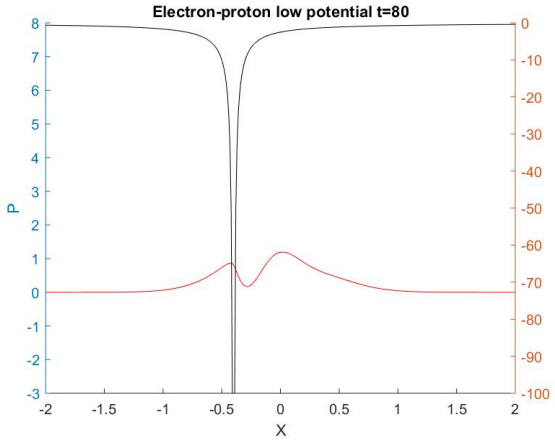

Electron-proton deep negative potential t=1
P
X


Electron-proton deep negative potential t=20
P
X


Electron-proton deep negative potential t=50
P
X
×10^4


Electron-proton deep negative potential t=80
P
X
×10^4


Electron-proton deep negative potential
P
X
t


Electron-proton low potential t=80
P
X

6 Particle in a one-dimensional lattice

6.1 Description

With the idea of applying our programs to a more complex quantum mechanical system, in this section we will study the particle in a one-dimensional lattice problem. This is a model designed to understand the behaviour of an electron in a crystal, in the simplified case of a 1D lattice of positive ions. This creates a periodic potential independent of time.

We will start by discussing the main analytical results that have been obtained on this topic; later, we will try to solve the system numerically and compare our approximations to the theoretical predictions.

6.2 Theoretical approach

6.2.1 Useful results

In the following lines, we will prove Bloch's theorem for the general, 3-dimensional case. However, a simpler justification of the 1-dimensional case like the one given by Kittel[15] would also be enough for our purpose.

Definition:

A wave in a potential lattice is said to be a Bloch wave if its wavefunction can be described by the form:

$$\Psi(\vec{r}) = e^{i\vec{k}\cdot\vec{r}}u(\vec{r})$$

where $\vec{r}$ is the position vector, $\vec{k}$ is the Bloch wave vector and u is a periodic function with the same spatial periodicity as the potential.

Theorem:

Bloch's theorem states that the energy eigenstates for a particle in a potential lattice are described by Bloch waves.

Proof:

One of the main tools in our proof will be the primitive lattice vectors $\vec{a}_1$, $\vec{a}_2$, $\vec{a}_3$. If we shift our position by any integer combination of them (translation operator) $\vec{T}_{n_1,n_2,n_3} = n_1\vec{a}_1 + n_2\vec{a}_2 + n_3\vec{a}_3$, we will end up in the starting potential distribution. We will also use the reciprocal lattice vectors $\vec{b}_1$, $\vec{b}_2$, $\vec{b}_3$, which verify the property $\vec{a}_j \cdot \vec{b}_k = 2\pi\delta_{jk}$ (where δ_{jk} is the Kronecker delta). Now we will prove a useful lemma:

Lemma: If a wave function Ψ is an eigenstate of $\vec{T}_{n_1,n_2,n_3} \forall n_1, n_2, n_3$, then Ψ is a Bloch wave.

Proof: If Ψ is an eigenstate of all translation vectors, in particular,

$$\Psi(\vec{r} + \vec{a}_j) = \lambda_j \Psi(\vec{r})$$

for $j = 1, 2, 3$, where $\lambda_j \in \mathbb{R}$. We can find three numbers θ_1, θ_2, θ_3 such that $e^{2\pi i\theta_j} = \lambda_j$, and then

$$\Psi(\vec{r} + \vec{a}_j) = e^{2\pi i\theta_j}\Psi(\vec{r})$$

Defining $\vec{k} = \theta_1\vec{b}_1 + \theta_2\vec{b}_2 + \theta_3\vec{b}_3$ and the function $u(\vec{r}) = e^{-i\vec{k}\cdot\vec{r}}\Psi(\vec{r})$, we get

$$u(\vec{r}+\vec{a}_j) = e^{-i\vec{k}\cdot(\vec{r}+\vec{a}_j)}\Psi(\vec{r}+\vec{a}_j) = [e^{-i\vec{k}\cdot\vec{r}}e^{-i\vec{k}\cdot\vec{a}_j}][e^{2\pi i\theta_j}\Psi(\vec{r})] = e^{-i\vec{k}\cdot\vec{r}}e^{-i\theta_j 2\pi}e^{2\pi i\theta_j}\Psi(\vec{r}) = e^{-i\vec{k}\cdot\vec{r}}\Psi(\vec{r}) = u(\vec{r})$$

Consequently, u has the same spatial period as the lattice and, since $\Psi(\vec{r}) = e^{i\vec{k}\cdot\vec{r}}u(\vec{r})$, we conclude that Ψ is a Bloch wave.

We are now ready to prove Bloch's theorem. Since our potential lattice has translational symmetry, any translation operator $\vec{T}_{n_1,n_2,n_3}$ commutes with the Hamiltonian operator and with any other translation operator. As a consequence, there is a simultaneous eigenbasis for the Hamiltonian and every possible $\vec{T}_{n_1,n_2,n_3}$. The wavefunctions in this basis are both energy eigenstates (because they are eigenstates of the Hamiltonian) and Bloch waves (because of the lemma proven above), *qed*.

6.2.2 Quantization of the Bloch factor

In the one-dimensional lattice we intend to study, potential is represented by a periodic function; therefore, according to Bloch's theorem, we can write the wavefunction of a particle the following way:

$$\Psi(x) = e^{ikx}u(x)$$

where $u(x+a) = u(x)$. Since we are treating an ideally infinite ion lattice, we can represent it as a ring through the use of the Born-Von Kármán boundary conditions:

$$\Psi(0) = \Psi(L)$$

being $L \gg a$ the total length of the lattice. The total amount of ions is $N = L/a$. Plugging this into the boundary condition and applying Bloch's theorem, we find that Bloch factor k is quantized:

$$\Psi(0) = \Psi(L) \implies e^{ik0}u(0) = e^{ikL}u(L) \implies u(0) = e^{ikL}u(Na) = e^{ikL}u(0) \implies$$

$$e^{ikL} = 1 \implies kL = 2\pi n \implies k = \frac{2\pi n}{L}$$

6.2.3 Kronig-Penney model

The Kronig-Penney model is an idealized version of the one-dimensional lattice problem which consists of a periodic array of potential wells in the locations of the ions. Here, we will use a slightly different notation than the one chosen by Kronig and Penney in their original work, in order to make it fit with the names of the variables employed in the sections above.

Thanks to Bloch's theorem, we only need to find a solution for a single period and prove that it verifies some boundary conditions. We will call a the length of the period, b will be the separation between two consecutive wells and V_0 the height of the barrier between the wells. We have the Schrödinger equation:

$$\frac{\partial^2\Psi}{\partial x^2} = -\gamma^2(E - V(x))\Psi, \qquad \gamma^2 = \frac{2m}{\hbar^2}.$$

We can consider a period from $x = -b$ to $x = a - b$ and, provided that we will later study the limit $V_0 \to \infty$, we may assume $0 < E < V_0$. We find ourselves in the well-known step potential problem, with the following time-independent solutions:

$$\Psi(x) = \begin{cases} Ae^{i\alpha x} + Be^{-i\alpha x} & 0 < x < a - b \\ Ce^{i\beta x} + De^{-i\beta x} & -b < x < 0 \end{cases}$$

where $\alpha = \gamma\sqrt{E}$, $\beta = \gamma\sqrt{V_0 - E}$. Applying Bloch's theorem, we take out e^{ikx} from both expressions and we get:

$$u(x) = \begin{cases} Ae^{i(\alpha-k)x} + Be^{-i(\alpha+k)x} & 0 < x < a - b \\ Ce^{i(\beta-k)x} + De^{-i(\beta+k)x} & -b < x < 0 \end{cases}$$

The constants A, B, C, D must be chosen so that the solution is continuous and smooth for $x = 0$, while from the periodicity of u we conclude that $u(-b) = u(a - b)$ and $\frac{\partial u}{\partial x}(-b) = \frac{\partial u}{\partial x}(a - b)$. The linear homogeneous system of equations that derives from these requirements is:

$$\begin{bmatrix} 1 & 1 & -1 & -1 \\ \alpha & -\alpha & -\beta & \beta \\ e^{i(\alpha-k)(a-b)} & e^{-i(\alpha+k)(a-b)} & -e^{-i(\beta-k)b} & -e^{i(\beta+k)b} \\ (\alpha-k)e^{i(\alpha-k)(a-b)} & -(\alpha+k)e^{-i(\alpha+k)(a-b)} & -(\beta-k)e^{-i(\beta-k)b} & (\beta+k)e^{i(\beta+k)b} \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

For the system to have non-trivial solutions, the determinant must be zero; therefore, the following equation must hold:

$$cos(ka) = cos(\beta b)cos[\alpha(a-b)] - \frac{\alpha^2+\beta^2}{2\alpha\beta}sin(\beta b)sin[\alpha(a-b)]$$

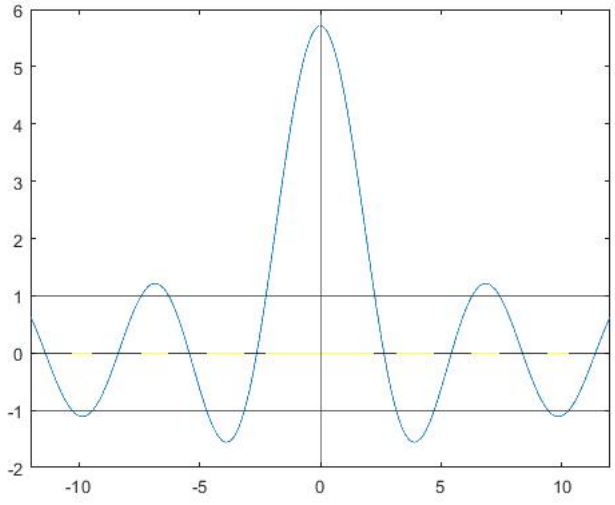
Passing to the limit where $V_0 \to \infty$ and $b \to 0$ while mantaining $\beta^2 b$ finite and defining

$$P = \lim_{V_0 \to -\infty,\ b \to 0} \frac{\beta^2 ab}{2}$$

we reach

$$cos(ka) = cos(\alpha a) + P\frac{sin(\alpha a)}{\alpha a}$$

a transcendental equation. In order to find the possible values of αa, we will plot the right hand side of the equation assuming for P the value $3\pi/2$:



Where the horizontal axis represents αa and the vertical one, $cos(\alpha a) + P\frac{sin(\alpha a)}{\alpha a}$. Since the left hand side of the expression is $cos(ka)$, the values of αa for which the ordinates are above 1 or below -1 cannot verify the equation (they are plotted in yellow). Moreover, using the quantized expression we found for the Bloch factor $k = \frac{2\pi n}{L}$ and approaching the infinite lattice limit ($L \to \infty$) we can vary k continously; as a result, any value between [-1,1] will be allowed. Consequently, as α depends on E, the energy levels which an electron in a lattice may have form a spectrum composed of continuous portions separated by finite intervals.

This was the conclusion obtained by Kronig and Penney, and it gives rise to the idea of electronic bandstructure. This concept is key in the study of electrical conduction in ionic and metallic lattices; a material is a better or worse conductor depending on the band gap that electrons have to surpass in order to jump from the valence band to the conduction band.

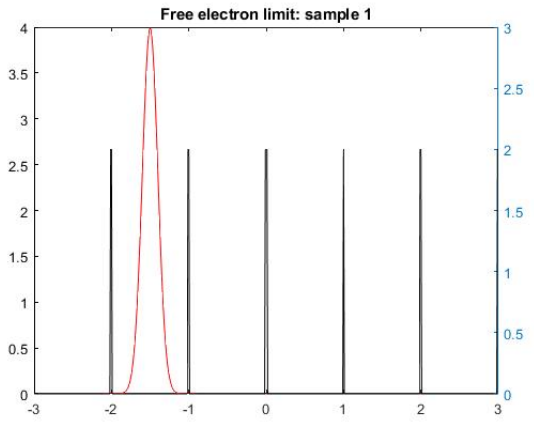
Finally, we will briefly study the effect of varying P. If it vanishes, the curve of the figure tends to $cos(\alpha a)$ and there are no forbidden regions. This result makes sense, since $P \to 0 \implies \beta^2 b \to 0 \implies V_0 b \to 0$ means that the potential barriers between wells are much more "thin" than they are "tall". In other words, we are approaching the free electron limit, where all energy values are permitted. If we increase P, the gaps appear, and they are smaller for larger values of αa (electrons with higher energy, thus faster, can pass the potential barriers more easily). When $P \to \infty$, the distance between wells grows and electrons cannot jump from one to another. The only allowed values for αa are now the points $n\pi$ with $n \in \mathbb{Z}^*$. Replacing the expression of α and solving for E, we get:

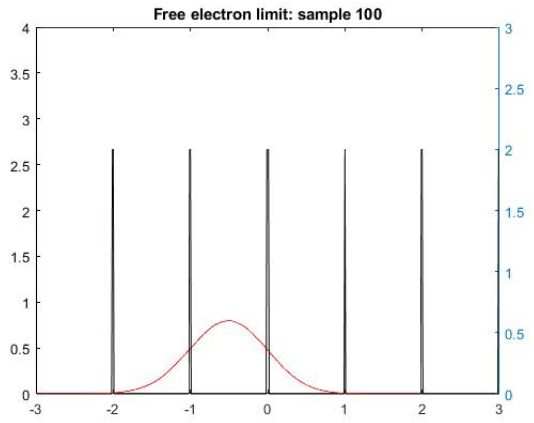
$$\frac{\sqrt{2mE}}{\hbar} a = n\pi \implies E = \frac{n^2 h^2}{8ma^2}, \qquad n \in \mathbb{N}$$

which is the energy of an electron in an infinite potential well. That is, the electron has become bound.

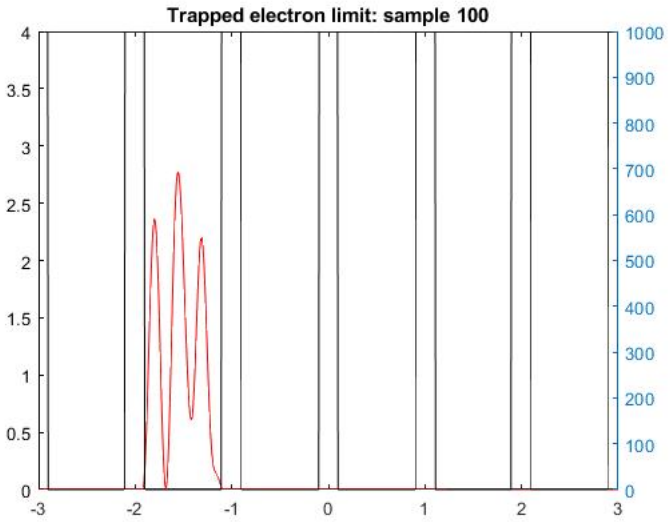
6.3 Numerical results

For our numerical approximations of the problem, we implemented the Kronig-Penney model with six potential wells representing six ions of the lattice. We began studying if the limits $P \to 0$ and $P \to \infty$ really corresponded to a free electron and a bound one. With that purpose, we first created a lattice with thin, small potential barriers and then one with very thick, tall ones. In both cases, the same initial conditions were used: a very concentrated wave located at one potential well and moving towards the right.



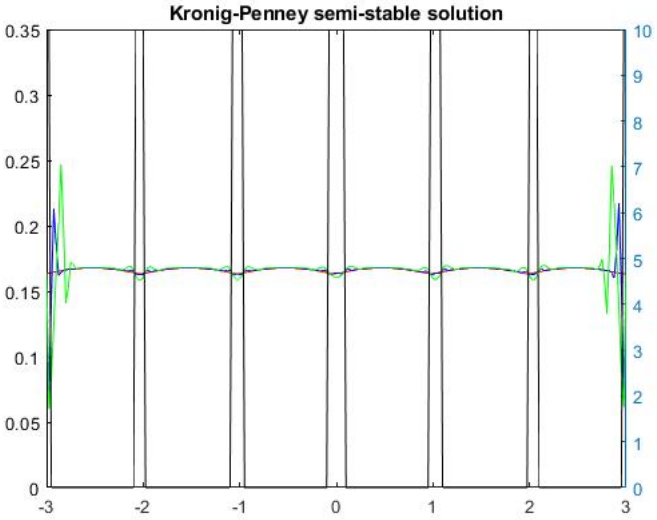


As we can see, in this first example we obtain the desired results: the wave propagates as in the free-particle case, becoming less localized with time. In the $P \to \infty$ limit, the simulations also agree with the theory: the wave function cannot escape from one of the ions, and it starts bouncing back and forth between the potential walls:



Finally, we wanted generate some initial conditions according to those developed by Kronig and Penney, in order to check if they remained stable. Creating a wave function that fulfilled all the requirements proved to be much tougher than one would expect: we had to develop algorithms which searched for energy values that annulated the determinant of the matrix explained in the previous section, and even then any approximation led to catastrophic numerical errors.

After many attempts, we obtained some good initial conditions. We introduced them into the Matlab code; however, the implementation of the Born-Von Kármán boundary conditions was not satisfactory. In the end, we could not avoid some numerical errors due to our lattice being finite.



In the figure, we show three different moments over time: in red, we can see the initial probability distribution. After 20 iterations, the blue line is obtained: the central part remains unchanged, but an undesired perturbation is generated in the borders of the lattice. After 50 iterations, the green plot shows how the perturbation is slowly travelling towards the centre and the middle points have already begun to move a bit as a consequence of the errors in the extrems. If the program runs for a longer time, the

perturbation keeps moving until it reaches $x = 0$ from both sides. However, if we bear in mind that the lattice is supposed to have infinitely many ions, the problems in the boundaries would not affect the centre and the solution would be, indeed, stable.

7 Numerical Methods and Codes

7.1 Euler's Explicit Method

This one is a simple recursive method to compute solutions for PDE's. We start by discretizing space and time and, in one spatial dimension, we use the notation Ψ_i^n for the value of function Ψ at point i and time n.

The approximated derivatives for the function will be:

$$\frac{\partial \Psi}{\partial t} \approx \frac{\Psi_i^n - \Psi_i^{n-1}}{\Delta t}, \qquad \frac{\partial \Psi}{\partial x} \approx \frac{\Psi_i^n - \Psi_{i-1}^n}{\Delta x} \implies \frac{\partial^2 \Psi}{\partial x^2} \approx \frac{\Psi_{i+1}^n - 2\Psi_i^n + \Psi_{i-1}^n}{\Delta x^2}$$

In order to solve a PDE of the form

$$\frac{\partial \Psi(x,t)}{\partial t} = F\left(u, x, t, \frac{\partial \Psi(x,t)}{\partial x}, \frac{\partial^2 \Psi(x,t)}{\partial x^2}\right)$$

Euler's method transforms it into:

$$\frac{\Psi_i^n - \Psi_i^{n-1}}{\Delta t} = F_i^n\left(u, x_i, t_n, \frac{\Psi_i^n - \Psi_{i-1}^n}{\Delta x}, \frac{\Psi_{i+1}^n - 2\Psi_i^n + \Psi_{i-1}^n}{\Delta x^2}\right)$$

from where we can isolate Ψ_i^{n+1}. In the Schrödinger equation with time-independent potential case, we have

$$F\left(u, x, t, \frac{\partial \Psi(x,t)}{\partial x}, \frac{\partial^2 \Psi(x,t)}{\partial x^2}\right) = \frac{i\hbar}{2m}\frac{\partial^2 \Psi(x,t)}{\partial x^2} - \frac{i}{\hbar}V(x)\Psi(x,t)$$

Applying the discretization,

$$\frac{\Psi_i^n - \Psi_i^{n-1}}{\Delta t} = \frac{i\hbar}{2m}\frac{\Psi_{i+1}^n - 2\Psi_i^n + \Psi_{i-1}^n}{\Delta x^2} - \frac{i}{\hbar}V(x)\Psi_i^n \implies$$

$$\implies \Psi_i^{n+1} = \Psi_i^n + i\Delta t\left(\frac{\hbar}{2m}\frac{\Psi_{i+1}^n - 2\Psi_i^n + \Psi_{i-1}^n}{\Delta x^2} - \frac{V(x)}{\hbar}\Psi_i^n\right)$$

The main problem we had with this method was that the Δx^2 in the denominator created great numerical instability, therefore preventing us from employing precise partitions of the x axis.

7.2 Crank-Nicolson Method (1D)

Crank-Nicolson is a stable method to get numerical solutions of PDE's. We needed to implement it because Euler's explicit method did not provide satisfactory results. In fact, it can be seen as a mixture of Euler's implicit an explicit methods, as the recurrency is given by

$$\frac{\Psi_i^{n+1} - \Psi_i^n}{\Delta t} = \frac{1}{2}\Big[F_i^{n+1}(x, t, \Psi, \frac{\partial \Psi}{\partial x}, \frac{\partial \Psi}{\partial x^2}) - F_i^n(x, t, \Psi, \frac{\partial \Psi}{\partial x}, \frac{\partial \Psi}{\partial x^2})\Big] \forall i \in 1, \ldots, length(\vec{x})$$

or using Schrödinger Equation expression

$$i\hbar\frac{\Psi_i^{n+1}-\Psi_i^n}{\Delta t}=\frac{1}{2}\Big[\frac{-\hbar^2}{2m}\frac{\Psi_{i+1}^{n+1}-2\Psi_i^{n+1}+\Psi_{i-1}^{n+1}}{\Delta x^2}+V(x)\Psi_i^{n+1}+\frac{-\hbar^2}{2m}\frac{\Psi_{i+1}^{n}-2\Psi_i^{n}+\Psi_{i-1}^{n}}{\Delta x^2}+V(x)\Psi_i^{n}\Big]$$

Manipulating this expression in order to separate values of Ψ at time n and $n+1$ we get to

$$\frac{\hbar^2}{2m\Delta x^2}\Psi_{i+1}^{n+1}+\left(2\frac{i\hbar}{\Delta t}-2\frac{\hbar^2}{2m\Delta x^2}-V(x)\right)\Psi_i^{n+1}+\frac{\hbar^2}{2m\Delta x^2}\Psi_{i-1}^{n+1}=$$
$$-\frac{\hbar^2}{2m\Delta x^2}\Psi_{i+1}^{n}+\left(2\frac{i\hbar}{\Delta t}+2\frac{\hbar^2}{2m\Delta x^2}+V(x)\right)\Psi_i^{n}-\frac{\hbar^2}{2m\Delta x^2}\Psi_{i-1}^{n}$$

Taking $r:=\frac{\hbar^2}{2m\Delta x^2}$, $A_1(x):=\frac{2i\hbar}{\Delta t}-2r-V(x)$ and $A_2(x):=\frac{2i\hbar}{\Delta t}+2r+V(x)$ the expression becomes

$$r\Psi_{i+1}^{n+1}+A_1\Psi_i^{n+1}+r\Psi_{i-1}^{n+1}=-r\Psi_{i+1}^{n}+A_2\Psi_i^{n}-r\Psi_{i-1}^{n}$$

Considering this recursive relation $\forall i\in 1,\dots,length(\vec{x})=m$ one can write

$$\begin{bmatrix} r & A_1(x_2) & r & 0 & 0 & & \\ 0 & r & A_1(x_3) & r & 0 & & \\ 0 & 0 & r & A_1(x_4) & r & & \\ & & & & \ddots & & \\ & & & 0 & r & A_1(x_{m-1}) & r \end{bmatrix}\begin{bmatrix}\Psi_1^{n+1}\\ \Psi_2^{n+1}\\ \Psi_3^{n+1}\\ \vdots\\ \Psi_m^{n+1}\end{bmatrix}=$$

$$\begin{bmatrix} -r & A_2(x_2) & -r & 0 & 0 & & \\ 0 & -r & A_2(x_3) & -r & 0 & & \\ 0 & 0 & -r & A_2(x_4) & -r & & \\ & & & & \ddots & & \\ & & & 0 & -r & A_2(x_{m-1}) & -r \end{bmatrix}\begin{bmatrix}\Psi_1^{n}\\ \Psi_2^{n}\\ \Psi_3^{n}\\ \vdots\\ \Psi_m^{n}\end{bmatrix}$$

As we are in the assumption that Ψ_i^n are known, the unknown variables are $\Psi_2^{n+1},\dots,\Psi_{m-1}^{n+1}$ (since we fix the bounding values of Ψ_1^n and $\Psi_m^n\forall n$).

$$\begin{bmatrix} A_1(x_2) & r & 0 & 0 \\ r & A_1(x_3) & r & 0 \\ 0 & r & A_1(x_4) & r \\ & & & \ddots \\ & 0 & r & A_1(x_{m-1}) \end{bmatrix}\begin{bmatrix}\Psi_2^{n+1}\\ \Psi_3^{n+1}\\ \Psi_4^{n+1}\\ \vdots\\ \Psi_{m-1}^{n+1}\end{bmatrix}=$$
$$\begin{bmatrix} A_2(x_2) & -r & 0 & 0 & \\ -r & A_2(x_3) & -r & 0 & \\ 0 & -r & A_2(x_3) & -r & \\ & & & \ddots & \\ & & 0 & -r & A_2(x_{m-1}) \end{bmatrix}\begin{bmatrix}\Psi_1^{n}\\ \Psi_2^{n}\\ \Psi_3^{n}\\ \vdots\\ \Psi_m^{n}\end{bmatrix}-\begin{bmatrix}r\Psi_1^{n+1}\\ 0\\ \vdots\\ 0\\ r\Psi_m^{n+1}\end{bmatrix}$$

yields, which is clearly a tridiagonal system of equations, that can be solved via a number of suitable algorithms of linear complexity. This gives a unique solution at each time n. To write it compactly,

$$M\Psi^{n+1} = N\Psi^n - \vec{n}$$

7.3 Crank-Nicolson Method (2D)

If we consider more dimensions, Schrödinger equation turns into

$$i\hbar\frac{\partial\Psi}{\partial t} = -\frac{\hbar^2}{2m}\nabla^2\Psi + V(\mathbf{r},t)\Psi$$

and the construction of the method for 2D looks like

$$i\hbar\frac{\Psi_{i,j}^{n+1}-\Psi_{i,j}^{n}}{\Delta t} = \frac{1}{2}\Big[\frac{-\hbar^2}{2m}\Big(\frac{\Psi_{i+1,j}^{n+1}-2\Psi_{i,j}^{n+1}+\Psi_{i-1,j}^{n+1}}{\Delta x^2}+\frac{\Psi_{i,j+1}^{n+1}-2\Psi_{i,j}^{n+1}+\Psi_{i,j-1}^{n+1}}{\Delta y^2}\Big)$$
$$+V(x)_{i,j}\Psi_{i,j}^{n+1}+\frac{-\hbar^2}{2m}\Big(\frac{\Psi_{i+1,j}^{n}-2\Psi_{i,j}^{n}+\Psi_{i-1,j}^{n}}{\Delta x^2}+\frac{\Psi_{i,j+1}^{n}-2\Psi_{i,j}^{n}+\Psi_{i,j-1}^{n}}{\Delta y^2}\Big)+V(x)_{i,j}\Psi_{i,j}^{n}\Big]$$

where the i index refers to the position in the x axis and the j index to the y axis. So following a similar procedure to the 1D case, and considering a uniform discrete grid with $\Delta x = \Delta y$, one can get to

$$A_1\Psi_{i,j}^{n+1} + r\Big(\Psi_{i+1,j}^{n+1}+\Psi_{i-1,j}^{n+1}+\Psi_{i,j+1}^{n+1}+\Psi_{i,j-1}^{n+1}\Big) =$$
$$A_2\Psi_{i,j}^{n} - r\Big(\Psi_{i+1,j}^{n}+\Psi_{i-1,j}^{n}+\Psi_{i,j+1}^{n}+\Psi_{i,j-1}^{n}\Big)$$

where $r = \frac{\hbar^2}{2m\Delta x^2}$, $A_1 = \frac{2i\hbar}{\Delta t} - 4r - V_{i,j}$ and $A_2 = \frac{2i\hbar}{\Delta t} + 4r + V_{i,j}$. Or matricially,

$$\begin{bmatrix} A_{1(2,2)} & r & 0 & \dots & r & 0 & & & \\ r & A_{1(2,3)} & r & & & & & & \\ 0 & r & \ddots & r & & & \ddots & & \\ \vdots & & r & A_{1(2,m-1)} & 0 & & & r & \\ r & & & 0 & A_{1(3,2)} & r & & & \\ 0 & \ddots & & & r & \ddots & & & \\ & & & & & & \ddots & r & \\ & & & r & & & r & A_{1(m-1,m-1)} \end{bmatrix} \begin{bmatrix} \Psi_{2,2}^{n+1} \\ \Psi_{2,3}^{n+1} \\ \vdots \\ \Psi_{2,m-1}^{n+1} \\ \Psi_{3,2}^{n+1} \\ \vdots \\ \vdots \\ \Psi_{m-1,m-1}^{n+1} \end{bmatrix} =$$
$$\begin{bmatrix} A_{2(2,2)} & -r & 0 & \dots & -r & 0 & & & \\ -r & A_{2(2,3)} & -r & & & & & & \\ 0 & -r & \ddots & -r & & & \ddots & & \\ \vdots & & -r & A_{2(2,m-1)} & 0 & & & -r & \\ -r & & & 0 & A_{2(3,2)} & -r & & & \\ 0 & \ddots & & & -r & \ddots & & & \\ & & & & & & \ddots & -r & \\ & & & -r & & & -r & A_{2(m-1,m-1)} \end{bmatrix} \begin{bmatrix} \Psi_{2,2}^{n} \\ \Psi_{2,3}^{n} \\ \vdots \\ \Psi_{2,m-1}^{n} \\ \Psi_{3,2}^{n} \\ \vdots \\ \vdots \\ \Psi_{m-1,m-1}^{n} \end{bmatrix}$$

using $m = length(x)$, x discrete space vector. It is implicitly supposed that the border conditions make sure the wave function equals zero in the limits, otherwise a correcting vector should we added to the right-hand side. In essence, it is a tridiagonal matrix system (except for some zeroes) with one extra upper diagonal and an extra lower diagonal. The recursive relation provided a '3D' matrix system, that we decided to resize to a 2D matrix system in order to solve it using *Matlab*. The problem is that we get $(m-2)$ equations for the system to be solved. That makes our code substantially slower and restricts the possibilities.

7.4 Base Codes

To solve Schrödinger PDE's numerically, we designed some rewritable *Matlab* programs to observe and analyse the results. We first did a program involving Euler's explicit method. We realised that it was not stable enough, so we decided to implement Crank-Nicholson's method. This one was harder to design and slower to run, however the results were far more satisfying.

```
deltax=0.01; %Divisions of x
deltat=0.001; %Didvisions of time

n0 = 3;      %Energy state
a = 3;       %Size of the box
m = 1;       %Mass of the particle (It cancels out, so it doesn't affect)

x=0:deltax:a;                    %Vector of space points
E = (n0^2*pi^2*1/(2*m*a^2));     %Energy of the particle
k = sqrt(2*m*E);                 %Associated wavenumber
Psi(1,1:length(x))=sin(k*x);     %Initial conditions

%Normalization of the function in order than (Psi*Psi.')*deltax = 1
b=Psi(1,:)*Psi(1,:).';
z=1/sqrt(b*deltax);
Psi(1,:)=z*Psi(1,:);

t=0:deltat:10; %Vector of time points

%Boundary conditions: in the limits of the well, Psi = 0;
Psi(2:length(t),1)=0;
Psi(2:length(t),length(x))=0;

%Implementing C-N
r=1/(2*deltax^2);
a1=(2*1i/deltat-2*r)*ones(1,length(x)-2);
a2=(2*1i/deltat+2*r)*ones(1,length(x)-2);
M=diag(a1)+diag(r.*ones(1,length(x)-3),-1)+diag(r.*ones(1,length(x)-3),1);
N=diag(a2)+diag(-r.*ones(1,length(x)-3),-1)+diag(-r.*ones(1,length(x)-3),1);
%n=zeros(1,length(x)-2).';
%n(1,1)=-r*Psi(i,1);  %In the boundary they equal zero
%n(1,end)=-r*Psi(i,end;)
for i=2:length(t)
    Psi(i,2:length(x)-1)=M\(N*Psi(i-1,2:length(x)-1).'); %+n if necessary

    b=Psi(i,:)*Psi(i,:).'; %To normalize(Psi*Psi_conjugated)*deltax=1
    z=1/sqrt(b*deltax);
    Psi(i,:)=z*Psi(i,:);
end

%Calculate the probability for each space/time point.
for i=1:length(t)
    for k=1:length(x)
        MPsi3(i,k)=abs(Psi(i,k))^2;
```

```
    end
end

% Create surface plot
surf(MPsi3(:,:))
% Remove edge lines a smooth colors
shading interp
% Hold the current graph
hold on
% Add the contour graph to the pcolor graph
contour3(MPsi3(:,:),20,'k')
xlabel('Position (x)')
ylabel('Time (t)')
zlabel('Probability(x,t)')
title(['Crank-Nicolson n = ' num2str(n0)])
axis([0 a*100 0 10000])
% Return to default
hold off
```

Listing 1: `box_cn.m`

Here we have the basic code that uses Crank-Nicolson 1D. In this case, it is applied to a particle confined in a box, with the initial conditions of the stable solution over the time. Modifications can be done to add other potential, initial conditions, Δx and Δt values...

In the annex some other examples can be found. To test them, it is only necessary to copy them and execute on Matlab.

8 Conclusions

There is no doubt that numerical methods are a powerful tool to study quantum mechanical systems. In the basic cases, they provide us with accurate approximations of reality and constitute a wonderful visual support for a student who is starting to study analytically the Schrödinger equation.

We must not forget, however, that there are some drawbacks when we attempt to treat more complex systems. As we saw in the one-dimensional lattice section, the impossibility to reproduce concepts like infinity or continuity without some level of error may result in unstable solutions. Also, the use of realistic data is really difficult, because they may differ in tens of orders of magnitude, creating huge numerical inaccuracies.

Nonetheless, the main advantage of these methods is that, in cases where an analytical study would become an absolute nightmare, we can obtain reasonable approximations with relative ease. These perks are clearly exemplified in the time dependent and the curved potentials, where the simulations give us a good intuition of how the system would behave.

References

[1] Planck units - *wikipedia.*

[2] Solving TDSE using finite difference methods - *R. Becerril, F.S. Guzmán, A. Rendón-Romero, S. Valdez-Alvarado.*

[3] Solution of the Schrödinger equation by Labview - *Óscar Durán Avendaño, Carlos Ramírez-Martín.*

[4] Crank-Nicolson method - *wikipedia.*

[5] Numerical solution of the TDSE - *Christoph Wachter.*

[6] Heat PDE Crank-Nicolson Matlab - *Zientziateka.*

[7] The Quantum Mechanics Visualisation Project - *University of St Andrews.*

[8] Elementary particle - *wikipedia.*

[9] Eisberg, R. M., Resnick, R. (1985). *Quantum physics of atoms, molecules, solids, nuclei, and particles* (2nd ed.). Wiley.

[10] Quantum Mechanics of Electrons in Crystal Lattices - *R. de L. Kronig, W.G. Penney.*

[11] Particle in a one-dimensional cristal lattice - *wikipedia.*

[12] Bloch's theorem - *Andrei research group, Rutgers University.*

[13] Bloch waves - *wikipedia.*

[14] Simple Proofs of Bloch's Theorem

[15] Kittel, C. (1996). *Introduction to Solid State Physics* (7th ed.). New York, USA: Wiley.

[16] Crystal momentum - *wikipedia.*

A APPENDIX: Base codes

A.1 Two indistinguishable particles in a 1D box

```
close all
deltax=0.05;
deltat=0.05;

% Quantum state of the particles, limits of the box and symmetry of the
% function is fixed. Symmetric (A = 0); Antisymmetric (A = 1).
na = 3;
nb = 2;
a = 2;
A = 1;

x1=-a/2:deltax:a/2; % Space points vector.
Ea = (na^2*pi^2*1/(2*a^2));
ka = sqrt(2*Ea);
x2 = -a/2:deltax:a/2;
Eb = (nb^2*pi^2*1/(2*a^2));
kb = sqrt(2*Eb);
c = sqrt(2/a);  % Normalization constant
C = sqrt(1/2);  % Normalization constant for the whole function (symmetric or
    antisymmetric)
L=length(x1);

if mod(na,2) == 1
     Psi1a(1,1:length(x1))=c*cos(ka*x1); %Initial conditions (the first index is the time
    )
     Psi2a(1,1:length(x2))=c*cos(ka*x2);
else
     Psi1a(1,1:length(x1))=c*sin(ka*x1);
     Psi2a(1,1:length(x2))=c*sin(ka*x2);
end

if mod(nb,2) == 1
     Psi1b(1,1:length(x1))=c*cos(kb*x1);
     Psi2b(1,1:length(x2))=c*cos(kb*x2);
else
     Psi1b(1,1:length(x1))=c*sin(kb*x1);
     Psi2b(1,1:length(x2))=c*sin(kb*x2);
end

for i = 1:length(x1)
    for k = 1:length(x2)
        if A == 0
             Ps(1,i,k) = C*(Psi1a(1,i)*Psi2b(1,k)+Psi2a(1,k)*Psi1b(1,i));
        else
             Ps(1,i,k) = C*(Psi1a(1,i)*Psi2b(1,k)-Psi2a(1,k)*Psi1b(1,i));
        end
    end
end


t=0:deltat:10; % Time points vector
T=length(t);

Ps(1:T, 1:L, 1)=0; % Boundary conditions
Ps(1:T, 1:L, L)=0;
Ps(1:T, 1, 1:L)=0;
Ps(1:T, L, 1:L)=0;

Psi=zeros(1,L^2);
```

```
for i=2:L-1
    Psi(1,(i-2)*(L-2)+1:(i-1)*(L-2))=Ps(1,i,2:L-1);
end

% Equation
r=1/(2*deltax^2);
a1=(2*j/deltat)-4*r;
a2=(2*j/deltat)+4*r;
M=zeros((L-2)^2, (L-2)^2);
N=zeros((L-2)^2, (L-2)^2);
for k=2:L-1
    M=diag(a1*ones(1,(L-2)^2))+diag(r*ones(1,(L-2)^2-1),-1)+diag(r*ones(1,(L-2)^2-1),1)+
    diag(r*ones(1,(L-2)^2-L+2),-L+2)+diag(r*ones(1,(L-2)^2-L+2),L-2);
    N=diag(a2*ones(1,(L-2)^2))+diag(-r*ones(1,(L-2)^2-1),-1)+diag(-r*ones(1,(L-2)^2-1),1)
    +diag(-r*ones(1,(L-2)^2-L+2),-L+2)+diag(-r*ones(1,(L-2)^2-L+2),L-2);
    for p=2:L-2
       M((p-1)*(L-2),(p-1)*(L-2)+1)=0;
       M((p-1)*(L-2)+1,(p-1)*(L-2))=0;
       N((p-1)*(L-2),(p-1)*(L-2)+1)=0;
       N((p-1)*(L-2)+1,(p-1)*(L-2))=0;
    end
end

for i=2:T
    Psi(i,1:(L-2)^2)=M\(N*Psi(i-1,1:(L-2)^2).'); %+n if necessary
    for k=2:L-1
        Ps(i,k,2:L-1)=Psi(i,(k-2)*(L-2)+1:(k-1)*(L-2));
    end

    b=sum(sum(Ps(1,:,:).*conj(Ps(1,:,:)))); %Normalization (Psi*Psi_transpuesta)*deltax=1
    z=1/(sqrt(b)*deltax);
    Ps(1,:,:)=z*Ps(1,:,:);
end

prob = zeros(T,L,L);
for i=1:T % Computation of probability at each space-time point to plot
    for k=1:L
        for p=1:L
            prob(i,k,p)=abs(Ps(i,k,p))^2;
        end
    end
end

for i = 1:length(t) % Plot along time (it is stable, so it does not change along time)
    probt=reshape(prob(i,:,:),[L,L]);
    mesh(x1,x2,probt);
    axis([-a/2 a/2 -a/2 a/2 0 2]);
    n1=num2str(na);
    n2=num2str(nb);
    if A == 0
        title (['Bosons n_a = ' n1 ', n_b = ' n2]);
    else
        title (['Fermions n_a = ' n1 ', n_b = ' n2]);
    end
    xlabel('x_1')
    ylabel('x_2')
    zlabel('Probability(x_1,x_2)')
    pause(0.1)
end
```

Listing 2: `indistinguishable.m`

A.2 One particle in a 2D box

```
close all
deltax=0.05; % Space divisions
deltat=0.05; % Time divisions
x=-1:deltax:1; % Space points vector
y=-1:deltax:1;
L=length(x);

t=0:deltat:10; % Time points vector
T=length(t);

for k=1:length(x)
    Psi1(1,k)=1/sqrt(.1*sqrt(pi))*exp(sqrt(-1)*0*x(k))*exp(-x(k)^2/4/.01);
end
for k=1:length(x)
    Psi2(1,k)=1/sqrt(.1*sqrt(pi))*exp(sqrt(-1)*0*x(k))*exp(-x(k)^2/4/.01);
end

for i = 1:length(x)
    for k = 1:length(x)
        Ps(1,i,k) = Psi1(1,k)*Psi2(1,i);
    end
end

Ps(1:T, 1:L, 1)=0; % Boundary conditions ( = 0)
Ps(1:T, 1:L, L)=0;
Ps(1:T, 1, 1:L)=0;
Ps(1:T, L, 1:L)=0;

b=sum(sum(Ps(1,:,:).*conj(Ps(1,:,:)))); %Normalization (Psi*Psi_transpuesta)*deltax=1
z=1/(sqrt(b)*deltax);
Ps(1,:,:)=z*Ps(1,:,:);

Psi=zeros(1,L^2);
for i=2:L-1
    Psi(1,(i-2)*(L-2)+1:(i-1)*(L-2))=Ps(1,i,2:L-1);
end

% Equation
r=1/(2*deltax^2);
a1=(2*1i/deltat)-4*r;
a2=(2*1i/deltat)+4*r;
M=zeros((L-2)^2, (L-2)^2);
N=zeros((L-2)^2, (L-2)^2);
for k=2:L-1
    M=diag(a1*ones(1,(L-2)^2))+diag(r*ones(1,(L-2)^2-1),-1)+diag(r*ones(1,(L-2)^2-1),1)+
    diag(r*ones(1,(L-2)^2-L+2),-L+2)+diag(r*ones(1,(L-2)^2-L+2),L-2);
    N=diag(a2*ones(1,(L-2)^2))+diag(-r*ones(1,(L-2)^2-1),-1)+diag(-r*ones(1,(L-2)^2-1),1)
    +diag(-r*ones(1,(L-2)^2-L+2),-L+2)+diag(-r*ones(1,(L-2)^2-L+2),L-2);
    for p=2:L-2
       M((p-1)*(L-2),(p-1)*(L-2)+1)=0;
       M((p-1)*(L-2)+1,(p-1)*(L-2))=0;
       N((p-1)*(L-2),(p-1)*(L-2)+1)=0;
       N((p-1)*(L-2)+1,(p-1)*(L-2))=0;
    end
end

for i=2:T
    Psi(i,1:(L-2)^2)=M\(N*Psi(i-1,1:(L-2)^2).'); %+n si es necesario
    for k=2:L-1
        Ps(i,k,2:L-1)=Psi(i,(k-2)*(L-2)+1:(k-1)*(L-2));
    end

```

```
    b=sum(sum(Ps(1,:,:).*conj(Ps(1,:,:)))); %Normalization(Psi*Psi_transpuesta)*deltax=1
    z=1/(sqrt(b)*deltax);
    Ps(1,:,:)=z*Ps(1,:,:);
end

prob = zeros(T,L,L);
for i=1:T % Computation of probability at each space-time point to plot
    for k=1:L
        for p=1:L
            prob(i,k,p)=abs(Ps(i,k,p))^2;
        end
    end
end

for i = 1:length(t)
    probt=reshape(prob(i,:,:),[L,L]);
    mesh(x,y,probt);
    axis([-1 1 -1 1 0 8]);
    c=num2str(i);
    title (['Sample ' c]);
    pause(0.1)
end
```

Listing 3: box$_2D.m$

A.3 Step potential

```
close all
deltax=0.01; % Space divisions
deltat=0.001; % Time divisions
x=-12:deltax:6; %Space points vector

for k=1:length(x)
    Psi(1,k)=1/sqrt(.1*sqrt(pi))*exp(sqrt(-1)*10*x(k))*exp(-x(k)^2/4/.01);
end

b=Psi(1,:)*Psi(1,:)'; %Normalization(Psi*Psi_transpuesta)*deltax=1
z=1/sqrt(b*deltax);
Psi(1,:)=z*Psi(1,:);
t=0:deltat:1; %Time points vector

Psi(2:length(t),1)=0; %Boundary conditions
Psi(2:length(t),length(x))=0;

%Potential
for k=1:length(x)
    if x(k)<0.8
        V(k)=0;
    else
        V(k)=150;
    end
end

%Equation
r=1/(2*deltax^2);
a1=(2*j/deltat-2*r)*ones(1,length(x)-2)-V(2:end-1);
a2=(2*j/deltat+2*r)*ones(1,length(x)-2)+V(2:end-1);
M=diag(a1)+diag(r.*ones(1,length(x)-3),-1)+diag(r.*ones(1,length(x)-3),1);
N=diag(a2)+diag(-r.*ones(1,length(x)-3),-1)+diag(-r.*ones(1,length(x)-3),1);
%n=zeros(1,length(x)-2)';
%n(1,1)=-r*Psi(i,1);  % They are equal to zero on the edges
%n(1,end)=-r*Psi(i,end;)
```

```
for i=2:length(t)
    Psi(i,2:length(x)-1)=M\(N*Psi(i-1,2:length(x)-1).'); %+n if necessary

    b=Psi(i,:)*Psi(i,:)'; %Normalize (Psi*Psi_transpuesta)*deltax=1
    z=1/sqrt(b*deltax);
    Psi(i,:)=z*Psi(i,:);
end

for i=1:length(t) %% Computation of probability at each space-time point to plot
    if (i-fix(i/100)*100)==0
        i
    end
    for k=1:length(x)
        MPsi2(i,k)=abs(Psi(i,k))^2;
    end
end

probl=0;
for k=1:480
    probl=probl+MPsi2(120,k);
end
probr=0;
for k=481:801
    probr=probr+MPsi2(120,k);
end

N=500;  % Until what sample it is shown
paso=2;

for i=1:paso:N
yyaxis right
plot(x,V,'k-')

yyaxis left
plot(x,MPsi2(i,:),'r-')
c=num2str(i);
axis([-1 2 0 5])
title(['Sample ' c]) % Title
pause(0.03)
end
```

Listing 4: `step_cn.m`

A.4 Barrier potential (time dependent)

```
close all;
deltax=0.01;
deltat=0.001;
x=-10:deltax:14; %Space points vector

Psi= zeros(1,length(x));
for k=1:length(x)
    Psi(1,k)=1/sqrt(.1*sqrt(pi))*exp(sqrt(-1)*10*x(k))*exp(-x(k)^2/4/.01);
end

b=Psi(1,:)*Psi(1,:)'; %Normalize
z=1/sqrt(b*deltax);
Psi(1,:)=z*Psi(1,:);
t=0:deltat:3; %Time points vector

Psi(2:length(t),1)=0; %Boundary conditions
Psi(2:length(t),length(x))=0;
```

```
18
19 %Potential
20 V = zeros(length(t), length(x));
21 for i = 1:length(t)
22     for k = 1:length(x)
23          if x(k)<1.5
24             V(i,k)=0;
25          elseif x(k)>2
26             V(i,k)=0;
27          else
28             V(i,k)=150*cos(2*pi*t(i));
29          end
30     end
31 end
32
33 r=1/(2*deltax^2);
34 for i=2:length(t)
35     a1=(2*1i/deltat-2*r)*ones(1,length(x)-2)-V(i,2:end-1);
36     a2=(2*1i/deltat+2*r)*ones(1,length(x)-2)+V(i,2:end-1);
37     M=diag(a1)+diag(r.*ones(1,length(x)-3),-1)+diag(r.*ones(1,length(x)-3),1);
38     N=diag(a2)+diag(-r.*ones(1,length(x)-3),-1)+diag(-r.*ones(1,length(x)-3),1);
39     Psi(i,2:length(x)-1)=M\(N*Psi(i-1,2:length(x)-1).');
40
41     b=Psi(i,:)*Psi(i,:)'; %Normalize
42     z=1/sqrt(b*deltax);
43     Psi(i,:)=z*Psi(i,:);
44 end
45
46 MPsi2 = zeros(length(t), length(x));
47 for i=1:length(t) %Probability
48     if (i-fix(i/100)*100)==0
49         i
50     end
51     for k=1:length(x)
52         MPsi2(i,k)=abs(Psi(i,k))^2;
53     end
54 end
55
56 N=2000;
57 paso=1;
58
59 for i=1:paso:N
60 yyaxis right
61 plot(x,V(i,:),'k-')
62 ylim([-200 200])
63
64 yyaxis left
65 plot(x,MPsi2(i,:),'r-')
66 c=num2str(i);
67 axis([-1 4 -4.5 4.5])
68 title(['V(t) = 150 cos(2\pit), Sample ' c])
69 xlabel('x')
70 ylabel('Probability')
71 pause(0.001)
72 end
```

Listing 5: `pot_var.m`

A.5 Charged particle

```
1 close all
2 deltax=0.01;    %space division
```

```
deltat=0.001;   %time division

x=-2:deltax:2;  %space vector

%initial conditions (first index is time)
for k=1:length(x)
    Psi(1,k)=1/sqrt(.1*sqrt(pi))*exp(sqrt(-1)*0*x(k))*exp(-x(k)^2/4/.01);
end
t=0:deltat:10;                  %time vector

Psi(2:length(t),1)=0;           %border conditions
Psi(2:length(t),length(x))=0;

for k=1:length(x)
    if x(k)<-0.4-0.1*deltax
        V(k)=1e3/(x(k)+0.4);
    elseif x(k)>-0.4+0.1*deltax
        V(k)=-1e3/(x(k)+0.4);
    else
        V(k)=V(k-1);
    end
end

%system of equations. if the V is depends on t, this part
%goes inside the for
r=1/(2*deltax^2);
a1=(2*j/deltat-2*r)*ones(1,length(x)-2)-V(2:end-1);  % -V(x) if necessary
a2=(2*j/deltat+2*r)*ones(1,length(x)-2)+V(2:end-1);  % +V(x) if necessary

M=diag(a1)+diag(r.*ones(1,length(x)-3),-1)+diag(r.*ones(1,length(x)-3),1);
N=diag(a2)+diag(-r.*ones(1,length(x)-3),-1)+diag(-r.*ones(1,length(x)-3),1);
%if the border conditions are not 0, this part must be added
%n=zeros(1,length(x)-2).';
%n(1,1)=-r*Psi(i,1);
%n(1,end)=-r*Psi(i,end;)
for i=2:length(t)
    Psi(i,2:length(x)-1)=M\(N*Psi(i-1,2:length(x)-1).');    %+n if necessary

    b=Psi(i,:)*Psi(i,:)';   %to rescale, sum(Psi*Psi_transpose)*deltax=1
    z=1/sqrt(b*deltax);
    Psi(i,:)=z*Psi(i,:);
end

for i=1:length(t)
   %computation of probability at each space-time point to plot
    for k=1:length(x)
        prob(i,k)=abs(Psi(i,k))^2;
    end
end
%plot
mesh(x,t,prob);

N=4000;  %Until what paso is simulated
paso=1;

figure
for i=1:paso:N
yyaxis right
plot(x,V,'k-')

yyaxis left
plot(x,prob(i,:),'r-')
axis([-2 2 -5 5])
c=num2str(i);
```

```
title(['Sample ' c]) %Title
pause(0.1)
end
```

Listing 6: `atom_def.m`

A.6 Lattice

```
% The following program generates initial conditions for the lattice problem
clear all
%Initial data:
E1=0.5;
E2=9.1;
a=1;
b=0.05;
m=0.1;
n=1;
V0=-10;
K=2*pi/6*n;

inc=0.1;
E=E1;
E_min=E1;
d_min=1e20;

% Minimization of the determinant:
while d_min>1e-20 && inc>1e-20
    clear d;
    clear E;
    n=0;
for kk=E1:inc:E2
    n=n+1;
    E(n)=kk;
    alfa=sqrt(2*m*E(n));
    beta=sqrt(2*m*(E(n)+V0));
    e1=exp(1i*(alfa-K)*(a-b));
    e2=exp(-1i*(alfa+K)*(a-b));
    e3=exp(-1i*(beta-K)*b);
    e4=exp(1i*(beta+K)*b);
    A=[1 1 -1 -1; alfa -alfa -beta beta; e1 e2 -e3 -e4; (alfa-K)*e1 -(alfa+K)*e2 -(beta-K
    )*e3 (beta+K)*e4];
    d(n)=abs(det(A));
end

 d_min=min(d);
    for nn=1:length(d)
        if d(nn)==d_min
            E_min=E(nn);
        end
    end
    E1=E_min-inc;
    E2=E_min+inc;
    inc=inc/10;
end

% Optimal energy and value of the determinant:
E_min
d_min
```

Listing 7: `initial_conditions.m`

```
clear all
close all
deltax=0.025; %Space division
deltat=0.0001; %Time division
x=-3.025:deltax:3.025; %Space points vector

%Data rendered by the program above:
E=5.979;
a=1;
bb=0.05;
m=0.1;
nn=1;
V0=-10;
K=2*pi/6*nn;

%Solving the system of equations:
alfa=sqrt(2*m*E);
beta=sqrt(2*m*(V0+E));

e1=exp(1i*(alfa-K)*(a-bb));
e2=exp(-1i*(alfa+K)*(a-bb));
e3=exp(-1i*(beta-K)*bb);
e4=exp(1i*(beta+K)*bb);
AA=[1 1 -1 -1; alfa -alfa -beta beta; e1 e2 -e3 -e4; (alfa-K)*e1 -(alfa+K)*e2 -(beta-K)*
    e3 (beta+K)*e4];
BB=AA(2:4,2:4);
abs(det(BB))
Br=-[alfa; e1; (alfa-K)*e1];

A=1;
C=linsolve(BB,Br);

u=zeros(243);
for kk=121:160
    if x(kk)>0
        u(kk)=A*exp(1i*(alfa-K)*x(kk))+C(1)*exp(-1i*(alfa+K)*x(kk));
    else
        u(kk)=C(2)*exp(1i*(beta-K)*x(kk))+C(3)*exp(-1i*(beta+K)*x(kk));
    end
end
for kk=2:40
    u(kk)=u(kk+120);
end
for kk=41:80
    u(kk)=u(kk+80);
end
for kk=81:120
    u(kk)=u(kk+40);
end
for kk=161:200
    u(kk)=u(kk-40);
end
for kk=201:242
    u(kk)=u(kk-80);
end


for kk=2:242
    Psi(1,kk)=exp(i*K*x(kk))*u(kk);
end
Psi(1,1)=Psi(1,241);
Psi(1,243)=Psi(1,3);

b=Psi(1,:)*Psi(1,:)'; %Normalize
```

```
z=1/sqrt(b*deltax);
Psi(1,:)=z*Psi(1,:);
t=0:deltat:3; %Time points vector

%Potential
for k=1:length(x)
    if x(k)<=-2.975
        V(k)=10;
    elseif x(k)<-2.075
        V(k)=0;
    elseif x(k)<=-1.975
        V(k)=10;
    elseif x(k)<-1.075
        V(k)=0;
    elseif x(k)<=-0.975
        V(k)=10;
    elseif x(k)<-0.075
        V(k)=0;
    elseif x(k)<=0.075
        V(k)=10;
    elseif x(k)<0.975
        V(k)=0;
    elseif x(k)<=1.075
        V(k)=10;
    elseif x(k)<1.975
        V(k)=0;
    elseif x(k)<=2.075
        V(k)=10;
    elseif x(k)<2.975
        V(k)=0;
    else V(k)=10;
    end
end

%The system of equations:
paso=1

r=1/(2*deltax^2);
a1=(2*j/deltat-2*r)*ones(1,length(x)-2)-V(2:end-1);
a2=(2*j/deltat+2*r)*ones(1,length(x)-2)+V(2:end-1);
M=diag(a1)+diag(r.*ones(1,length(x)-3),-1)+diag(r.*ones(1,length(x)-3),1);
N=diag(a2)+diag(-r.*ones(1,length(x)-3),-1)+diag(-r.*ones(1,length(x)-3),1);
n=zeros(1,length(x)-2)';
n(1)=-r*Psi(1,1);
n(end)=-r*Psi(1,end);

for i=2:length(t)
    Psi(i,2:length(x)-1)=M\(N*Psi(i-1,2:length(x)-1).'+n); %+n si es necesario
    Psi(i,1)=Psi(i,length(x)-2);
    Psi(i,end)=Psi(i,3);
    b=Psi(i,:)*Psi(i,:)';  %Normalize
    z=1/sqrt(b*deltax);
    Psi(i,:)=z*Psi(i,:);
    n(1)=-r*Psi(i,1);      %Born-Von Karman boundary conditions
    n(end)=-r*Psi(i,end);
    i
end

paso=2
for i=1:length(t)
    %computation of probability at each space-time point to plot
    if (i-fix(i/100)*100)==0
        i
    end
```

```
    for k=1:length(x)
        MPsi2(i,k)=abs(Psi(i,k))^2;
    end
end

N=4000;
paso=2;

figure

yyaxis left

plot(x,MPsi2(1,:))
axis([-3 3 0 1])
hold

%axis([-1 2 0 5])
for i=1+paso:paso:N

yyaxis right

plot(x,V,'k-')

yyaxis left

plot(x,MPsi2(i-paso,:),'w-')
plot(x,MPsi2(i,:),'r-')
c=num2str(i);
title(['Muestra ' c]) %Title
pause(0.01)
end
```

Listing 8: `lattice_cn.m`