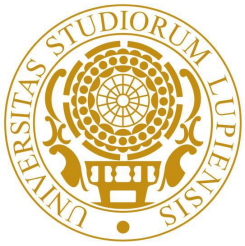


Università del Salento

Dipartimento di Matematica e Fisica

"Ennio De Giorgi"

Corso di laurea in Fisica

TESI DI LAUREA

Soluzione numerica dell'equazione di Schrödinger dipendente dal tempo

Numerical solution of the time-dependent Schrödinger equation.

Laureando:
Walter Martemucci

Relatore:
Prof. Giampaolo CO'

Anno Accademico 2020/2021

Contents

Abstract

The purpose of the thesis is to discuss the numerical solutions of a major example of parabolic partial differential equations, the time-dependent Schrödinger equation which belongs to the quantum mechanical framework.
The first chapter serves as introduction to some basic concepts of partial differential equations.
The second chapter is an analysis of the finite difference explicit method applied to the diffusion equation in the classical limit.
The third chapter deals with the more refined implicit method and introduce the Crank-Nicolson alternative strategy.
Lastly, the fourth chapter provides numerical solutions to the time-dependent Schrödinger equation given by the Crank-Nicolson approach. The results of each case will be treated according to the errors and stability of the algorithms applied.

Sintesi
Lo scopo di questa tesi è discutere alcuni dei possibili metodi per la risoluzione numerica dell'equazione di Schrödinger dipendente dal tempo.
Il primo capitolo serve da introduzione e fornisce informazioni su gli aspetti matematici fondamentali delle equazioni differenziali parziali in fisica.
Il secondo e il terzo capitolo forniscono una disamina dello schema esplicito, implicito e Crank-Nicolson per il metodo delle differenze finite, inoltre sono analizzate le caratteristiche di accuratezza e stabilità e sono forniti degli esempi di applicazioni numeriche all'equazione classica della diffusione.
Nel quarto capitolo si applica il metodo numerico ritenuto più interessante alla soluzione dell'equazione di Schrödinger dipendente dal tempo.
Si sono scelte come condizioni al contorno quelle di Dirichlet per studiare l'evoluzione temporale di un pacchetto d'onda Gaussiano applicando diversi potenziali, per alcuni di questi sono state fornite delle soluzioni analitiche per il confronto.

Chapter 1

Partial differential equations

In mathematics, partial differential equations involves the partial derivatives of a function of several independent variables.

Partial differential equations are involved in the description of the vast majority of physical situations where quantities vary in space and time.

Some of these phenomena are heat, fluid dynamics, electrostatics, electrodynamics, elasticity, diffusion and quantum mechanics (Schrödinger waves).

In all but the simplest cases, the analytical solution is impracticable and so, in order to obtain quantitative results, numerical methods must be adopted.

In a typical numeric treatment, the dependent variables (such as temperature) are described by their values at discrete points of a lattice of the independent variables (e.g. space and time) and, by appropriate discretization, the partial differential equation is reduced to a large set of difference equations [Koo86].

1.1 Classification of partial differential equations

Many physically interesting partial differential equation equations are of second order and their classification is crucial in the choice for the numerical solution.

Here, a direct jump to the subject of the study is convenient for the purpose of readability. A partial differential equation is called linear if it is linear the unknown function and its derivatives. The order is given by the term with the highest derivatives. For a function U the general linear second order partial differential equation in two independent variables take on the form:

$$A(x,y)U_{xx} + 2B(x,y)U_{xy} + C(x,y)U_{y,y} + ...(\text{lower order terms}) = F(x,y,U,U_x,U_y)$$

(assuming $U_{xy} = U_{yx}$).

Depending on the values that functions A, B or C could take, we define three type of equations:

Elliptic

$$AC > B^2$$

An example is the Laplace's equation, in which $A = C = 1$ e $B = 0$.

$$U_{xx} + U_{yy} = 0 \ .$$

Roughly speaking, elliptical equations involve second order derivatives in each of the independent variables, each one having the same sign when all terms in the equation are grouped together.

Hyperbolic

$$AC < B^2$$

a possible example is the one dimensional wave equation in cui $A = 1, C = -1/c^2, B = 0$.

$$U_{xx} = \frac{1}{c^2} U_{tt}$$

Hyperbolic equations involve second derivatives of opposite signs.

Parabolical

$$AC = B^2$$

e.g. heat or diffusion equation $A = \beta, B = C = 0$.

$$F(x, t, U, U_x, U_t) = U_t = \beta U_{xx} \ .$$

Parabolic equations involve only a first-order derivative in one variable and second order derivatives in the remaining variables.

1.2 Examples for parabolic partial differential equations

1.2.1 Well-posed partial differential equations

The essential characteristics for partial differential equations to be well-posed are the following:

1. a solution exists,

2. the solution is unique,

3. the solution depends continuously on the data.

Each of these features can be shown for the following examples.

A typical parabolic partial differential equation which can encountered in physical situations is the **diffusion equation**

$$\frac{\partial \phi}{\partial t} = \nabla \cdot (D \nabla \phi) + S, \tag{1.1}$$

where ∇ refers to $\mathbf{r} \equiv (x, y, z)$ is the 3-D space coordinate, t is time, ϕ(r,t) is the unknown function to be solved for, D is the diffusion coefficient and determines how fast ϕ changes in time while S is a source function, and the **time-dependent Schröedinger equation**

$$i\hbar \frac{\partial}{\partial t}\phi(\mathbf{r}, t) = -\frac{\hbar^2}{2m}\nabla^2 \phi(\mathbf{r}, t) + V(\mathbf{r})\phi(\mathbf{r}, t) \tag{1.2}$$

where V is the potential, m the particle mass, $\hbar = \frac{h}{2\pi}$ is the Planck constant.

1.2.2 Initial and boundary value problems

With only a first-order derivative in time, only one initial condition is needed, while the second-order derivative in space leads to a demand for two boundary conditions, that is the reason why these problems are generally of the initial value type with respect to time and a boundary value type with respect to space, as in the case of heat equation. The value of the field ϕ is given at an initial time and the purpose is to find it at a later time, the evolution being subject to spatial boundary conditions as in the cases taken into account:

-**Diffusion case**, the temperature or heat flux is specified on some surfaces:
Dirichlet boundary conditions, where $\phi(t, x_{bound})$ is given,
Neumann boundary conditions, flux through the boundary is given,
No-flow boundary conditions, no flux through the boundary.

-**Quantum case**, the Schrödinger wave function vanishes at very large distances.

Chapter 2

Explicit finite difference method

Let's start the discussion by treating *diffusion in one dimension with a uniform diffusion constant and a null source function.*

I assume the variable x to vary between 0 and 1 and consider Dirichlet boundary conditions that specify the values of the field $\phi(x,t)$ at the end points of this interval.

The problem is symmetric about $x = \frac{1}{2}$ and so we need the solution only for $0 \leq x \leq \frac{1}{2}$.

Applying an appropriate scaling, it's obtained from (1.1) its analogue

$$\frac{\partial \phi}{\partial t} = \frac{\partial^2 \phi}{\partial x^2} + S(x,t) \ . \tag{2.1}$$

The first step in the discretization procedure is to replace the domains by sets of mesh points and produce collections of discrete numerical approximations of the derivative.

2.1 Naive discretization

We proceed by approximating the spatial derivatives by finite differences on a uniform lattice, covering the region of interest, of N+1 points having equal spacing $\Delta x = \frac{1}{N}$ (h is used in the code instead of Δx).

While the time derivative is approximated by the simplest first-order difference formula assuming a time step Δt.
It's quite fundamental to introduce a value, whom importance will later be clear (see appendixB),

$$\mu = \frac{\Delta t}{(\Delta x)^2} \ . \tag{2.2}$$

Using the superscript n to label the time step ($\phi^n \equiv \phi(t_n),\ t_n = n\Delta t$) imply that equation (2.1) can be approximated as:

$$\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} = \frac{1}{(\Delta x)^2}(\delta^2 \phi^n)_i + S_i^n \ , \tag{2.3}$$

where δ^2 is the second-order difference operator (see appendixA),

$$(\delta^2 \phi)_i \equiv \phi_{i+1} - 2\phi_i + \phi_{i-1} \ . \tag{2.4}$$

At $i = 1$ and $i = N$, this equation involves the Dirichlet boundary conditions specifying ϕ_0 and ϕ_N.

The basic idea behind the explicit differencing scheme is to calculate the state of a system at a later time from the state of the system at the current time and equation (2.3) is a good example of explicit algorithm, since given ϕ^n at one time, solving for ϕ^{n+1} at the next time.

In an obvious matrix notation, we have:

$$\phi^{n+1} = (1 - \mathcal{H}\Delta t)\phi^n + S^n \Delta t \ , \tag{2.5}$$

where the action of the operator $\mathcal{H}$ is defined by

$$(\mathcal{H}\phi)_i \equiv -\frac{1}{(\Delta x)^2}(\delta^2 \phi)_i \ . \tag{2.6}$$

Extended matrix notation for the difference operator δ^2:

$$\begin{pmatrix} -2 & 1 & & & & & \\ 1 & -2 & 1 & & & & \\ & \cdot & \cdot & \cdot & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & 1 & -2 & 1 \\ & & & & & 1 & -2 \end{pmatrix}$$

2.2 Implementation

The explicit scheme has been applied with $\phi(0) = \phi(N) = 0$ as boundary conditions and $S = 0$, where we seek to find ϕ^{n+1}. It's given an initial condition of a Gaussian centered about $x = 1/2$,

$$\phi(x, t = 0) \ = \ e^{-20(x-\frac{1}{2})^2} - e^{-20(x-\frac{3}{2})^2} - e^{-20(x+\frac{1}{2})^2} \ , \tag{2.7}$$

second and third exponential functions ensure, respectively, the boundary conditions at x=1 and x=0.

The FORTRAN code applies (2.4).

"PHI" as stated in the code designates the numerical solution of the methods adopted while "EXACT" refers to the analytical solution of a spreading Gaussian,

$$\phi(x,t) \;=\; \tau^{-\frac{1}{2}}[e^{\frac{-20(x-\frac{1}{2})^2}{\tau}} - e^{\frac{-20(x-\frac{3}{2})^2}{\tau}} - e^{\frac{-20(x+\frac{1}{2})^2}{\tau}}]; \quad \tau = 1 + 80t.$$

Results for three different time values are displayed in figure (2.1).

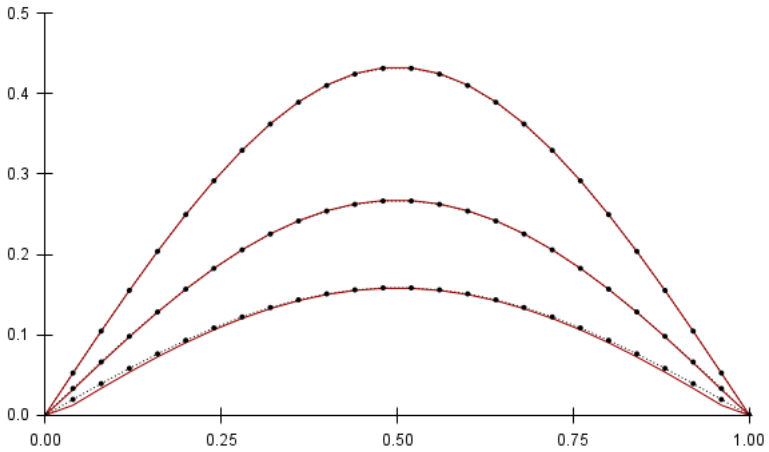


Figure 2.1: Density (amount of substance per unit volume: $n = \frac{N}{V}$) versus space (length). Results from the explicit algorithm for the one-dimensional diffusion of a Gaussian in a fixed lattice spacing Δx=0.04, with the same time steps $\Delta t = 0.00075$ at different times (from above then scrolling down, $t = \frac{200}{3}\Delta t, t = 130\Delta t, t = 200\Delta t$). Numerical solutions (dashed lines and black dots) and exact results (solid red line) are plotted in the chart.

These time steps are quite small compared to the natural time scale of the solution. That's a good idea trying to increase the time step to 0.001 but things escalates quickly because a non-physical instability develops in the numerical solution, that rapidly acquires violent oscillations from one lattice point to another soon after $t = 0.05$.

2.3 Numerical analysis

We have assumed that the explicit finite difference method provide reasonable approximations to the exact solution of the partial differential equation.

The investigation of this assumption is an important part of the numerical analysis and we shall outline the nature of the problems in a general way. We are concerned with four related properties: compatibility, convergence, accuracy and stability.

Compatibility
In deriving the explicit equation we neglected higher-order terms in the approximation series, these constitute a **truncation error** (see appendix A).

We require that the truncation error should tend to zero as Δx and Δt approach zero. If this is not so, the difference scheme is said to be incompatible or inconsistent with the partial differential equation.

In this case, the difference solution is not likely to approach the solution sought because of the higher instability the system develops when smaller time (or space) steps are taken figures (2.2), (2.3).

Convergence
Assuming compatibility is ensured, it is necessary to answer the question of whether the solution of the difference equations converges to the solution of the partial differential equation as the grid size approaches zero (**discretization error**) .

Let EXACT represent the exact analytical solution of a partial differential equation, with x and t independent variables, and NUM the exact solution of the approximating difference equations.

The finite-difference solution is said to be convergent when NUM tends to EXACT as Δx and Δt both tend to zero.

Once established the convergence, it should be possible to decrease the difference $EXACT - NUM$, and hence improve the difference solution by decreasing Δx or Δt or both.

Fox and Mayers[Fox68]

Accuracy and stability
If it were possible to carry out calculations to an infinite number of decimal places and if the initial and boundary data were specified exactly, the numerical calculations would produce the exact solution NUM of the difference equations. In practice, of course, each calculation is carried out to a finite number of figures and **round off errors** are introduced.

Rounding error is the difference between the result produced by a given algorithm using exact arithmetic and the result produced by the same algorithm using finite-precision, rounded arithmetic.

The solution actually computed is not NUM but PHI which we can call the "numerical solution".

The subject of stability is concerned with the possible build up of errors in the calculation which would cause PHI to be significantly different from NUM.

A set of finite-difference equations is said to be stable when the cumulative effect of all rounding errors is negligible. If the magnitudes of the errors introduced at the various grid points are each less than Δx or Δt, then the finite difference equations are usually considered stable if the maximum value of $NUM - PHI$ tends to zero as Δx or Δt

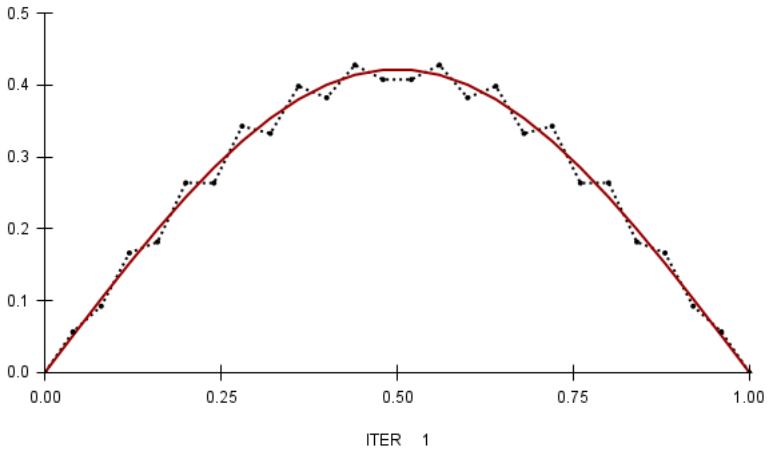


Figure 2.2: Same physical quantities and symbols used in fig.2.1. Result from the explicit algorithm for the one-dimensional diffusion of a Gaussian in a fixed lattice spacing Δx=0.04, with time steps $\Delta t = 0.001$ at time $t = 52\Delta t$.

tends to zero, and does not increase exponentially with the number of time steps in the computation.

More extensive treatments of these matters are given, e.g. by Smith [Smi65], Fox [Fox62], Mitchell [Mit69] and Crank [Cra75]

Figure 2.2 shows us that the value of $\mu = \frac{1}{2}$ is critical because of the fact that numerical solutions agree reasonably well with the analytical ones for $\mu \leq \frac{1}{2}$ while they develop oscillations quite after that critical point.

Numerical errors are proportional to the time step and the square of the lattice spacing that is called Courant-Friedrichs-Lewy condition for one dimensional case (see appendixB for more details).

If still a larger value of μ is taken,figure 2.3, the numerical solutions bears no resemblance to the analytical solutions which means that the algorithm has become unstable in the sense that errors increase without limit.

This represents a severe limitation because we are forced to take a large number of small time steps and many steps are required before something interesting happens.

In order to understand what is happening here let us call ψ_λ the eigenfunctions of the discrete operator $\mathcal{H}$ with eigenvalues ϵ_λ. Since $\mathcal{H}$ is a hermitian operator, the eigenvalues are real and the eigenvectors can be chosen to be orthonormal. The finite-dimensional spectral theorem (see appendix A) says that any hermitian matrix can be diagonalized by a unitary matrix, and that the resulting diagonal matrix has only real entries.

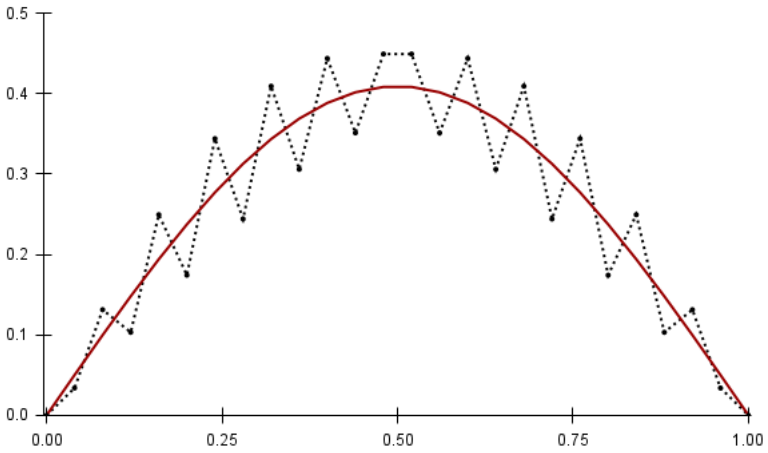


Figure 2.3: Same physical quantities and symbols used in fig.2.1. Result from the explicit algorithm for the one-dimensional diffusion of a Gaussian in a fixed lattice spacing Δx=0.04, with time steps $\Delta t = 0.001$ at time $t = 55\Delta t$.

We can expand the solution at any time in the basis as

$$\phi^n = \sum_\lambda A^n_\lambda \psi_\lambda \tag{2.8}$$

The exact time evolution is given by

$$\phi^n = e^{-\mathcal{H}n\Delta t}\phi^0, \tag{2.9}$$

so that each component of the solution should evolve as

$$A^n_\lambda = e^{-\epsilon_\lambda n\Delta t}A^0_\lambda \tag{2.10}$$

This corresponds to the correct behaviour of the diffusion equation, where short wavelenght components (with larger eigenvalues) disappear rapidly as the solution plot smooths out.

However, equation (2.5) shows that explicit scheme develop the expansion coefficients as

$$A^n_\lambda = (1 - \epsilon_\lambda \Delta t)^n A^0_\lambda. \tag{2.11}$$

As long as Δt is chosen small enough, the factor in the equation above approximates the exact result of $e^{-n\epsilon\lambda\Delta t}$ and the short-wavelength components turn off with time.

If, time step is too large, the quantities $1 - \epsilon_\lambda\Delta t$ have an absolute value larger than unity and the corresponding components of the initial solution, even if present only due to very small numerical round off errors, are successively amplified with each time step and soon grow to dominate.

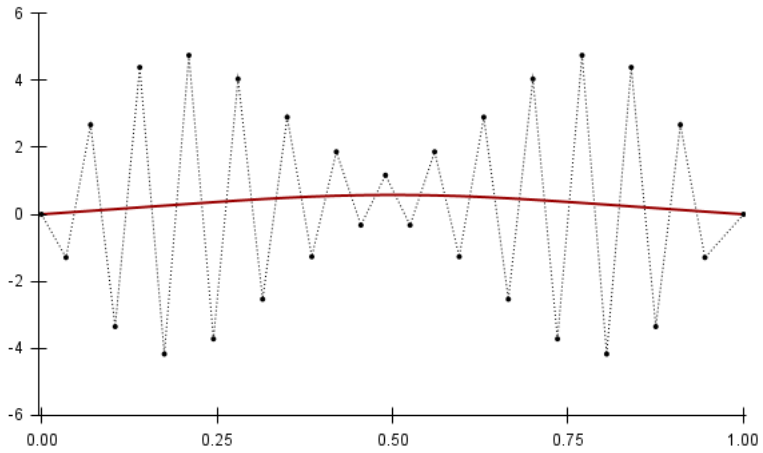


Figure 2.4: Same physical quantities and symbols used in fig.2.1. Result from the explicit algorithm for the one-dimensional diffusion of a Gaussian in a fixed lattice spacing Δx=0.035, with time steps $\Delta t = 0.00075$ at time $t = 0.025$.

In order to quantify the limit on Δt we use the knowledge of the eigenvalues of $\mathcal{H}$ that are known analytically in this simple model problem.

It is possible to verify that the functions

$$(\psi_\lambda)_i = sen\frac{\lambda\pi i}{N} \tag{2.12}$$

are un-normalized eigenfunctions of $\mathcal{H}$ (see appendix C for more detailed calculations) with the correct boundary conditions on a lattice of N+1 points for λ=1,2,....,N-1 and that the associated eigenvalues are

$$\epsilon_\lambda = \frac{4}{(\Delta x)^2} sen^2\frac{\lambda\phi}{2N}. \tag{2.13}$$

The largest eigenvalue of $\mathcal{H}$ is $\epsilon_{N-1} \approx 4(\Delta x)^{-2}$, which corresponds to an eigenvector that alternates in sign from one lattice point to the next.
Requiring $|1 - \epsilon_{N-1}\Delta t| < 1$ then restricts Δt to be less than $\frac{(\Delta x)^2}{2}$.

The instability sets in for smaller time steps if the spatial lattice is made finer and it's easy to show that because of the equation (2.2) that links spatial and time steps.

We can see the result of a finer lattice in figure 2.4.

Chapter 3

Implicit schemes

One way around the instability of the previous explicit algorithm is to perform a replacement on the general form of equation (2.3). The replacing consists in substituting the second space derivative with that of the solution at the new time. That is,

$$\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} = \frac{1}{(\Delta x)^2}(\delta^2 \phi^{n+1})_i + S_i^n \ . \tag{3.1}$$

This is an implicit scheme, since the unknown, ϕ^{n+1}, appears on both sides of the equation and we can solve for it as

$$\phi^{n+1} = \frac{1}{1 + \mathcal{H}\Delta t}[\phi^n + S^n \Delta t] \ . \tag{3.2}$$

We have defined these two operators, δ^2 and $\mathcal{H}$, in Eqs. (2.4) and (2.6).

This algorithm is practically equivalent to equation (2.5) to lowest order in Δt.

The great advantage of the implicit method consist in its unconditional stability, so we are able to choose the size of the time step according to accuracy rather than stability.

However, it is quite much better in that larger time steps than the explicit scheme (see equations (2.5), (2.13)), as the operator $(1 + \mathcal{H}\Delta t)^{-1}$ has eigenvalues $(1 + \epsilon_\lambda \Delta t)^{-1}$, all of whose moduli are less than 1 for any Δt. [Hig02]

Such as all components of the solution decrease with each time step avoiding amplifications which imply instability.

Although this decrease is inaccurate (i.e. not exponential) for the most rapidly oscillating components, those should not be large in the initial conditions if the spatial discretization is accurate.

For the slowly varying components of the solution corresponding to small eigenvalues, the evolution closely approximates the exponential at each time step.

3.1 Inversion of tri-diagonal matrices

A potential drawback of the implicit scheme (3.2) is that it formally requires the solution of a set of linear equations (tri-diagonal matrix) at each time step to find ϕ^{n+1}; this is equivalent to the application of the inverse of the matrix $1 + \mathcal{H}\Delta t$.

Since the inverse itself is time-independent, it might be found only once at the beginning of the calculation and then used for all times, but application still requires of order N^2 operations if done directly.

We are provided with an algorithm of Gaussian elimination and backward substitution (direct method) that could easily be applied and it does not incurs unpleasant complications by R. Varga. [Var62] The only possible shortcoming of the algorithm would be instability with respect to growth of rounding errors, but when matrix A is diagonally dominant and tri-diagonal it has been proved that the direct method is extremely stable with respect to the growth of such type errors. [Wil61]

This provides a very efficient solution (of order N operations) of a tri-diagonal system of equations such as the one proposed by equation (3.2), this algorithm is only well behaved if the matrix is diagonal dominant.

Let us consider the tri-diagonal linear system of equations $A\phi = b$ for the unknown ϕ_i, which is pretty much similar to our implicit system $A\phi^{n+1} = \phi^n + S^n\Delta t$ that we can freely rearrange to express every single component evolution with a linear equation:

$$A_i^- \phi_{i-1} + A_i^0 \phi_i + A_i^+ \phi_{i+1} = b_i \ . \tag{3.3}$$

Here, the $A^{\pm,0}$ are the only non-vanishing elements of A, which represents the inverse matrix $1 + \Delta t\mathcal{H}$ ($\mathcal{H}$ hermitian operator), and the b_i are the known quantities. This is the form of the problem posed by the evaluation of (3.2) for ϕ^{n+1}, where ϕ_0 and ϕ_N are given by the Dirichlet boundary conditions. In particular,

$$b_i = \phi_i^n + S_i^n \Delta t, \ A_i^0 = 1 + \frac{2\Delta t}{h^2}, \ A_i^\pm = -\frac{\Delta t}{h^2} \ . \tag{3.4}$$

To solve this system of equations, we assume that the solution satisfies a one term forward recursion relation of the form

$$\phi_{i+1} = \alpha_i \phi_i + \beta_i \ , \tag{3.5}$$

where the α_i and β_i are the coefficients to be determined.

Substituting the above formula into (3.3), we have

$$A_i^- \phi_{i-1} + A_i^0 \phi_i + A_i^+ (\alpha_i \phi_i + \beta_i) = b_i \ , \tag{3.6}$$

which can be solved for ϕ_i to yield

$$\phi_i = \gamma_i A_i^- \phi_{i-1} + \gamma_i (A_i^+ \beta_i - b_i) \ , \tag{3.7}$$

with

$$\gamma_i = -\frac{1}{A_i^0 + A_i^+ \alpha_i} \ . \tag{3.8}$$

Upon comparing equation (3.9) with (3.7), [Var62] we can identify the following backward recursion relations for the α_i and β_i (see appendix A for an in-depth explanation):

$$\alpha_{i-1} = \gamma_i A_i^- \ ; \tag{3.9}$$

$$\beta_{i-1} = \gamma_i (A_i^+ \beta_i - b_i) \ . \tag{3.10}$$

The calculations of the quantities above is equivalent to the reduction by Gaussian elimination of the matrix equation to a new matrix with unit diagonal entries (see appendix A).

As long as we move backwards (it is an upper triangular matrix) we can determine the α_i and β_i starting from the simplest one which involves ϕ_{N-1} and we move to the second equation which involves ϕ_{N-1} and ϕ_{N-2} and so on down to 0.

The starting values to be used are

$$\alpha_{N-1} = 0, \ \beta_{N-1} = \phi_N \ , \tag{3.11}$$

which guarantee the correct value of ϕ at the last lattice point (boundary). Having determined these coefficients, we can then use the recursion relation (3.5) in a forward sweep from $i = 0$ to $i = N-1$ to determine the solution, with the starting value ϕ_0 known from the boundary conditions.

The FORTRAN code implements the algorithm (3.2) for the diffusion problem we have been considering.

Figures 3.1 and 3.2 show some results and their errors, where it is clear that much more accurate solutions can be obtained using larger time step than with the explicit method.

The scheme is always numerically stable and convergent but more numerically intensive as it requires solving a system of numerical equations on each time step (see appendix A and appendix B).

However the increase in computational effort per time step is only about a factor of two.

Interesting is the fact that α_i and γ_i are independent of $b - i$, so they do not need to be computed at every time step inversion but it is more efficient to calculate them once, at the beginning, and then store the values; only the β_i have to be computed for each inversion (because it is dependent on ϕ_n).

3.2 Some tests and alternative algorithm

Further investigation on the accuracy and stability of the implicit algorithm are shown in figures 3.3, where tests are carried out involving changing in lattice spacing (Δx).

A major gain in accuracy could be achieved if we had used the average of the second space derivatives at the two time steps involved in equation (3.1),

$$\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} = \frac{1}{h^2}(\delta^2 \frac{1}{2}[\phi^{n+1} + \phi^n])_i + S_i^n \ , \tag{3.12}$$

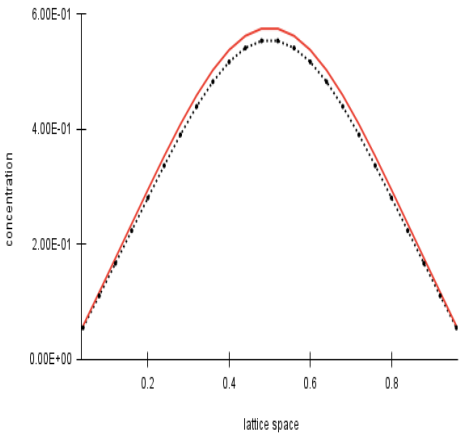
so that the time evolution is given by

$$\phi^{n+1} = \frac{1}{1 + \frac{1}{2}\mathcal{H}\Delta t}[(1 - \frac{1}{2}\mathcal{H}\Delta t)\phi^n + S^n \Delta t] \ , \tag{3.13}$$

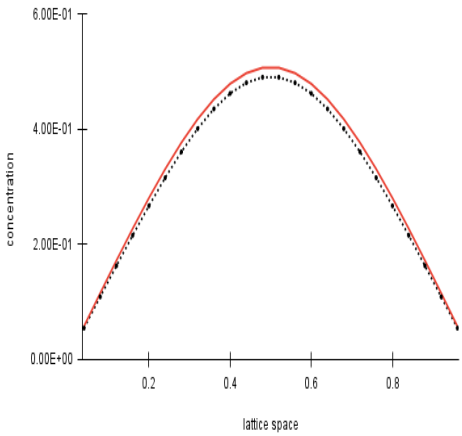
this is as good as the implicit scheme in its base version, because the components of the solution are diminished by factors whose absolute values are less than one.

The alternative approach we have just introduced is known as Crank-Nicolson method. [Cra75]

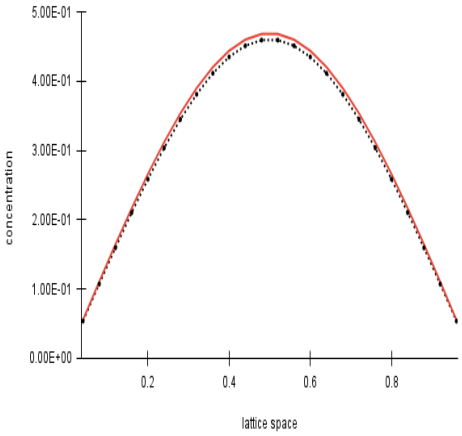
To summarize, usually the Crank–Nicolson scheme is the most accurate scheme for small time steps. For larger time steps, the implicit scheme works better since it is less computationally demanding. The explicit scheme is the least accurate and can be unstable, but is also the easiest to implement and the least numerically intensive.



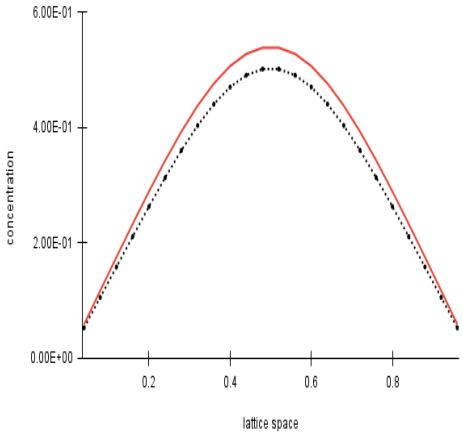
(a) $\Delta x = 0.04, \Delta t = 0.005, t = 6\Delta t$.



(b) $\Delta x = 0.04, \Delta t = 0.005, t = 8\Delta t$.

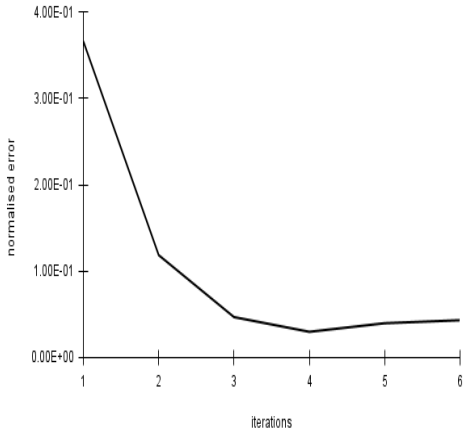


(c) $\Delta x = 0.04, \Delta t = 0.003, t = 15\Delta t$.

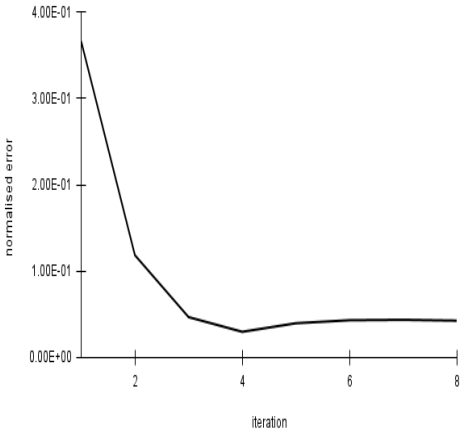


(d) $\Delta x = 0.04, \Delta t = 0.04, t = 4\Delta t$.

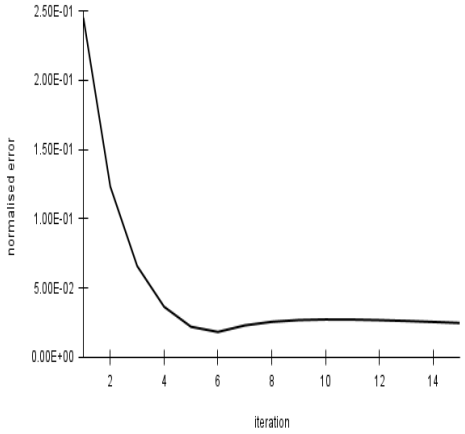
Figure 3.1: Solutions of the implicit algorithm, numerical (black line) and exact (red line), performing changes in time and time step



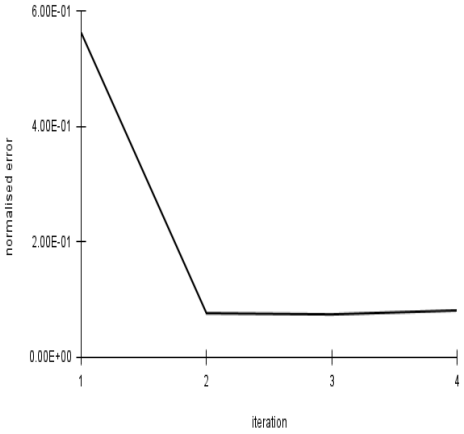
(a) $\Delta x = 0.04, \Delta t = 0.005, t = 6\Delta t$.



(b) $\Delta x = 0.04, \Delta t = 0.005, t = 8\Delta t$.

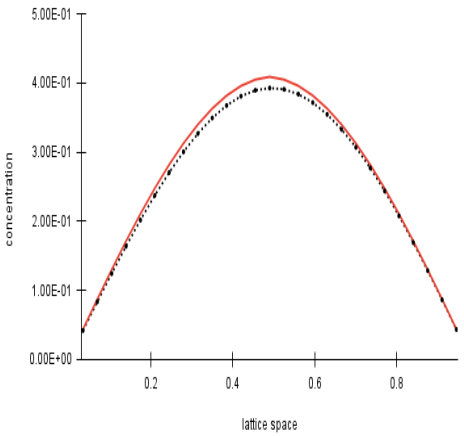


(c) $\Delta x = 0.04, \Delta t = 0.003, t = 15\Delta t$.

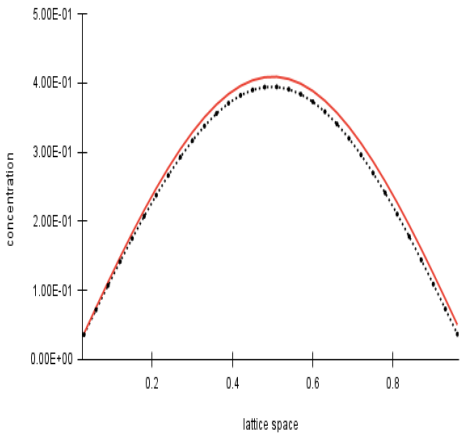


(d) $\Delta x = 0.04, \Delta t = 0.04, t = 4\Delta t$.

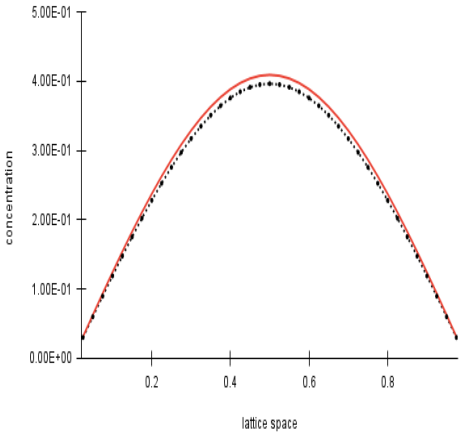
Figure 3.2: Trend of normalised error over time of the implicit scheme performing changes in time and time step.



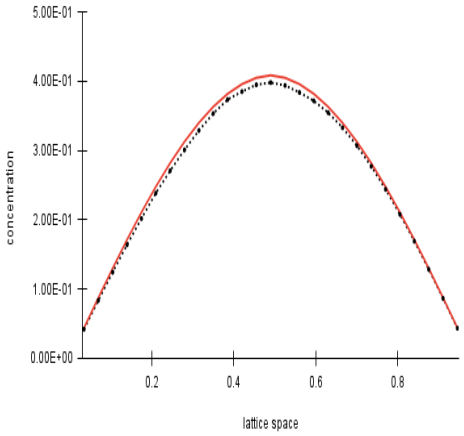
(a) $\Delta x = 0.035, \Delta t = 0.005, t = 12\Delta t$.



(b) $\Delta x = 0.030, \Delta t = 0.005, t = 12\Delta t$.



(c) $\Delta x = 0.025, \Delta t = 0.005, t = 12\Delta t$.



(d) $\Delta x = 0.020, \Delta t = 0.005, t = 12\Delta t$.

Figure 3.3: Solutions of the implicit algorithm, numerical (black line) and exact (red line), performing changes in the lattice step.

Chapter 4

Time dependent Schrödinger equation

It is now time to move to a quantum mechanical subject, as it could be the motion of a particle in a one dimensional domain.

The evolution of a complex wave function $\phi(x,t)$, given its value at some initial points, under the one-dimensional Schrödinger equation can be carried out with a numerical treatment of

$$i\hbar\frac{\partial}{\partial t}\phi = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2}\phi + V(x)\phi = -\frac{\hbar^2}{2m}\nabla^2\phi + V\phi, \tag{4.1}$$

or, alternatively, using the Hamiltonian operator $\mathcal{H} = -\frac{\hbar^2}{2m}\nabla^2 + V$

$$i\hbar\frac{\partial\phi}{\partial t} = \mathcal{H}\phi.$$

In our calculations we consider a particle with the electron mass $m = 0.511MeV$.
Dividing all terms of (4.1) by i

$$\frac{\partial\phi}{\partial t/\hbar} = -i\left[\frac{-\hbar^2}{2m}\nabla^2 + V\right]\phi, \tag{4.2}$$

where one can see that time is measured in $\hbar$ units ($\hbar \approx 6.582119 * 10^{-16} eV * s$).

Applying time discretization, as we have already done

$$\frac{\partial\phi}{\partial t/\hbar} => \frac{\phi^{n+1} - \phi^n}{\Delta T}, \tag{4.3}$$

defining $T = t/\hbar$.

As we are studying the time evolution in **one dimension**, the space discretization becomes

$$\nabla^2\phi => \frac{\partial^2}{\partial x^2}\phi => \frac{[\delta^2\phi]}{(\Delta x)^2} = \frac{[\phi_{j+1} + \phi_{j-1} - 2\phi_j]}{(\Delta x)^2}. \tag{4.4}$$

Subscripts and superscripts are used in the same manner as the diffusion problem (2.2).

Then we can rightfully choose to apply the Crank-Nicolson method to (4.2) instead of the intuitive, but less stable, explicit method we'll discuss later on that.

It is useful to introduce the value $\mathcal{V} = \frac{2m}{\hbar^2}V$ and make use of the operator $\mathcal{H}$

$$\mathcal{H}_j\phi_j = \frac{\hbar^2}{2m}(-\frac{1}{(\Delta x)^2}[\phi_{j+1} + \phi_{j-1} - 2\phi_j] + \mathcal{V}_j\phi_j) \ , \tag{4.5}$$

under the hypothesis, that the action on different discrete times can be described as the average of the application over two times.

Applying the space-time discretization to equation 4.2 it suddenly becomes

$$\frac{\phi_j^{n+1} - \phi_j^n}{\Delta T} = -\frac{i}{2}\left(\mathcal{H}_j\phi_j^{n+1} + \mathcal{H}_j\phi_j^n\right). \tag{4.6}$$

Now

$$\phi_j^{n+1} - \phi_j^n = -\frac{i}{2}(\mathcal{H}_j\phi_j^{n+1} + \mathcal{H}_j\phi_j^n)\Delta T, \tag{4.7}$$

so we move apart terms with a different time superscript

$$\phi_j^{n+1} + \frac{i}{2}\mathcal{H}_j\phi_j^{n+1}\Delta T = (1 - \frac{i}{2}\mathcal{H}_j\Delta T)\phi_j^n, \tag{4.8}$$

setting out a different configuration

$$[1 + \frac{i}{2}\mathcal{H}_j\Delta T]\phi_j^{n+1} = [1 - \frac{i}{2}\mathcal{H}_j\Delta T]\phi_j^n, \tag{4.9}$$

ultimately one gets the recursive algorithm:

$$\phi_j^{n+1} = [\frac{1 - \frac{i}{2}\mathcal{H}_j\Delta T}{1 + \frac{i}{2}\mathcal{H}_j\Delta T}]\phi_j^n = [\frac{2}{1 + \frac{i}{2}\mathcal{H}_j\Delta T} + 1]\phi_j^n. \tag{4.10}$$

Both explicit and implicit method are unsatisfactory, for the first one, the time evolution is given by

$$(1 - i\mathcal{H}\Delta T),$$

so it is unstable for any value of ΔT because, as we already know from appendix B, the eigenvalues of this operator,

$$(1 - i\epsilon_\lambda\Delta T),$$

have moduli

$$(1+\epsilon_\lambda^2(\Delta T)^2)^{\frac{1}{2}}$$

far greater than unity (ϵ_λ has been calculated in appendix B).

Implicit method is stable for all ΔT because the eigenvalues moduli of the operator,

$$(1+\epsilon_\lambda^2(\Delta T)^2)^{-\frac{1}{2}},$$

are obviously less than one, but operator itself is troublesome since it lacks of a crucial property of the exact solution, being unitary (see appendix E first paragraph).

On the other hand, 4.10 is unitary and conserves the norm of the wavefunction and so on the error order could be found from the Taylor series

$$[\frac{1-\frac{i}{2}\mathcal{H}_j\Delta T}{1+\frac{i}{2}\mathcal{H}_j\Delta T}] = (1-i\frac{\Delta T}{2\hbar}\mathcal{H}-\frac{(\Delta T)^2}{4\hbar^2}\mathcal{H}^2+...)(1-i\frac{\Delta T}{2\hbar}\mathcal{H}) = 1-\frac{i\Delta T}{\hbar}\mathcal{H}-\frac{(\Delta T)^2}{2\hbar^2}\mathcal{H}^2 = e^{-\frac{i\Delta T}{\hbar}\mathcal{H}}+O((\Delta T)^3)$$

4.1 Crank-Nicolson algorithm

Let us define the new funtion χ_j^n as:

$$\chi_j^n = \frac{2}{(1+\frac{i}{2}\Delta T\mathcal{H}_j)}\phi_j^n, \tag{4.11}$$

then following (4.5)

$$\chi_j^n + \Delta T\frac{i}{2}\frac{\hbar^2}{2m}(-\frac{1}{(\Delta x)^2}(\chi_{j+1}^n+\chi_{j-1}^n-2\chi_j^n)+\mathcal{V}_j\chi_j^n) = 2\phi_j^n, \tag{4.12}$$

and dividing previous equation by $-i(\frac{\hbar^2}{2m})\frac{\Delta T}{(\Delta x)^2}$ we obtain

$$\chi_{j+1}^n+\chi_{j-1}^n+\left[2i(\frac{2m}{\hbar^2})\frac{(\Delta x)^2}{\Delta T}-2-(\Delta x)^2\mathcal{V}_j\right]\chi_j^n = 4i\frac{(\Delta x)^2}{\Delta T}(\frac{2m}{\hbar^2})\phi_j^n, \tag{4.13}$$

this is the form of a tri-diagonal system, as we have previously seen in Chapter 3

$$A_j^+\chi_{j+1}^n + A_j^0\chi_j^n + A_j^-\chi_{j-1}^n = b_j^n, \tag{4.14}$$

with obvious definition of $A_j^\pm, A_j^0, b_j^n$. Assuming the solution satisfies the one term forward recursion relation of the form

$$\chi_{j+1}^n = \alpha_j^n\chi_j^n + \beta_j^n, \tag{4.15}$$

where α_j, β_j have to be specified, we turn back to (4.14) substituting (4.15)

$$A_j^+(\alpha_j^n\chi_j^n + \beta_j^n) + A_j^0\chi_j^n + A_j^-\chi_{j-1}^n = b_j^n, \tag{4.16}$$

$$(A_j^+\alpha_j^n + A_j^0)\chi_j^n = b_j^n - \beta_j^n A_j^+ - A_j^-\chi_{j-1}^n. \tag{4.17}$$

We obtain

$$\chi_j^n = \gamma_j^n[A_j^-\chi_{j-1}^n + A_j^+\beta_j^n - b_j^n] \tag{4.18}$$

where

$$\gamma_j^n = -\frac{1}{A_j^0 + A_j^+\alpha_j^n}.$$

Upon comparing (4.15) and (4.17) we are able to find the backward recursion relations:

$$\alpha_{j-1}^n = \gamma_j^n A_j^-, \tag{4.19}$$

$$\beta_{j-1} = \gamma_j^n(A_j^+\beta_j^n - b_j^n). \tag{4.20}$$

It is plain to realize that α_j and γ_j are time-independent while b_j is time-dependent through ϕ^n (see 4.13).

The numerical strategy is to calculate α_j and γ_j once for all while it is possible to build $b_j^{n=0}$ with $\phi_j^{n=0}$ then we get to $\beta_j^{n=0}$ thanks to (4.20).

It is just as simple as before to find $\chi_j^{n=1}$ with α, β, γ from (4.18) then ϕ_j^n is given by (4.11)

$$\phi_j^n = \frac{1}{2}[\chi_j^n + \frac{i}{2}\frac{\Delta T}{(\Delta x)^2}(\frac{-\hbar^2}{2m})(\chi_{j+1}^n + \chi_{j-1}^n - 2\chi_j^n) + \frac{i}{2}\Delta T V_j\chi_j^n]. \tag{4.21}$$

4.2 Additional features

Boundary conditions Boundary conditions here are given continuous but they will be implemented in the code using the discrete set of mesh points.

The one dimensional domain range goes from 0 to L, where $L \in \mathbb{R}$

$$\phi(0) = \phi(L) = 0.$$

From the above values it is plain to get the values of the matrix coefficients on the right edge of the discrete domain ($i \in [0, N] \subset \mathbb{N}$)

$$\alpha_N = 0;\ \beta_N = 0;\ \gamma_N = -\frac{1}{A_N^0}.$$

Normalization condition The probability to find the wave (ϕ) between the boundaries imposed by the problem has to be

$$P = \int |\phi_j(x)|^2 dx = 1.$$

In order to get correct wave function results the initial conditions has to be normalized; to do so, a subroutine has been implemented in the code adopting the extended Simpson's rule for integration (appendixE).

4.3 Results

The code describes a particle with electron mass moving through a one dimensional lattice under different potentials.

A variety of input data can be entered in the numerical code, among them, the space (Δx) and time step (Δt) amplitudes (for the obvious discretization), the width of the domain, the shape of the initial wavepacket (Gaussian or Lorentzian initial conditions) specifying the spatial width and the average position which can be set up on the lattice and evolved under the Dirichlet boundary conditions (effectively embedding our system inside an infinite square well).

The trials are done with a Gaussian wavepacket at $t = 0$, the real part of the starting condition is a function of the form

$$\phi(x, t = 0) = K \exp\left[-\mu(x - x_0)^2\right], \tag{4.22}$$

where μ, controls the width of function, x_0, gives the position of the centre of the peak, the packet lacks the expression for the initial momentum.

The constant K is defined by the normalisation condition.

Constant potential

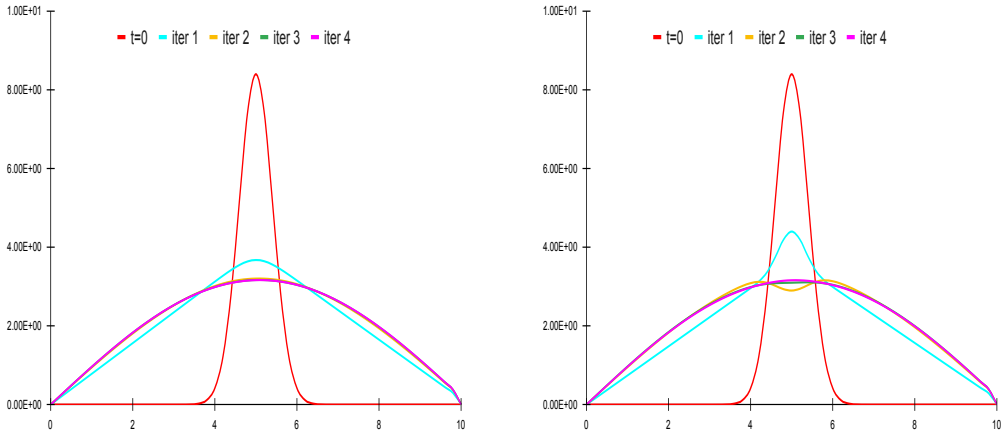
Several analytical forms of the potential can be defined in the one dimensional point lattice, starting from the constant zero potential whose results are given in figure 4.1.

The figure shows the evolution of the norm of a wave function $|\phi(x, t)|$ during total time of $T = 4\Delta t$, where the result of each iteration has been labelled with a different color, two panels with two different time steps (then different total time) have been included.

It is an interesting idea to compare the results obtained from the numerical methods to the analytical solutions of the time dependent Schrödinger equation, as it has been done in the previous chapters.

Since the potential is not time dependent (nor $\mathcal{H}$) but it is a function of position $V(x)$, the formal solution of time dependent Schrödinger equation is given by

$$\phi(x, t) = e^{\frac{-i\mathcal{H}t}{\hbar}} \phi(x, 0), \tag{4.23}$$



(a) Δt=0.1 (b) Δt=1

Figure 4.1: Numerical solutions $|\phi(x,t)|$ whose dimension units is $[length]^{-\frac{1}{2}}$ for a Gaussian wave packet in an one dimensional box (infinite square well). The common input data are $\mu = 3$, $x_0 = 5$, $\Delta x = 0.2$ and $V = 0$ eV. Red plot represents the starting packet $\phi(x,0)$ then the subsequent four iterations have been plotted.

where the space and the time dependence of the complete wave function are contained in separate factors and $\phi(x,0) \in \mathbb{C}$ and is a function of x only that solves the time-independent Schrödinger equation

$$(-\frac{\hbar}{2m}\frac{d^2}{dx^2} + V(x))\phi(x,0) = E\phi(x,0).$$

Therefore, a stationary state of energy, $E \in \mathbb{R}$, takes the form

$$\phi(x,t) = e^{\frac{-iEt}{\hbar}}\phi(x,0) \tag{4.24}$$

where $\phi(x,0)$ depends on the initial condition (starting wave packet).

Such a state is called stationary because its observables are time independent indeed the time dependence of the stationary state is carried by the exponential prefactor.

A problem with zero potential is know to have stationary states of energy E_i for a particle of m mass

$$\phi_i(x) = e^{\pm ikx} \tag{4.25}$$

where $k = \sqrt{\frac{2mE}{\hbar^2}}$.

Then, the analytical solutions of the Schrödinger equation take the general form

$$\phi(x,t) = \sum_i A_i e^{ik_i x} e^{-\frac{i}{\hbar}E_i t} = \sum_i A_i e^{i\sqrt{\frac{2m(E)}{\hbar^2}}x} e^{-\frac{i}{\hbar}E_i t} \tag{4.26}$$

and imposing the Dirichlet boundary conditions ($\phi(0) = \phi(L) = 0$) for the case of a particle confined to a segment (basically an infinite square well), the solutions become

$$\phi(x,t) = \sum_i A_i \sin(k_i x) e^{-\frac{i}{\hbar}E_i t} = \sum_i A_i \sin(n_i \frac{\pi}{L} x) e^{-\frac{i}{\hbar}E_i t} \tag{4.27}$$

where $i \in \mathbb{N}$, k_i is the wave number (also expressed as $k = \frac{\sqrt{2m(E-V_0)}}{\hbar}$), $n_i \in \mathbb{N} - 0$, L is the length of the segment where the particle moves and

$$E_i = \frac{\hbar^2}{2m} k_i^2 = \frac{\hbar^2}{2m} (n_i \frac{\pi}{L})^2$$

[Coh73].

From the analysis of the diffusion case it is quite clear that results of time evolution magnifies the components with smaller eigenvalues of $\mathcal{H}$ and for long times only these one are significant despite their lower amplitudes; which means that, thanks to the the exponential factor, term with lower energy $n_i = 1$ grows to dominate.

Error analysis

The second space derivatives for a stationary states in a zero potential is given by (4.25)

$$\frac{d^2\phi_i(x)}{dx^2} = -k^2\phi(x), \tag{4.28}$$

while three points approximation (appendixA) of the numerical treatment

$$\frac{\phi(x+\Delta x) - 2\phi(x) + \phi(x-\Delta x)}{(\Delta x)^2} = -\frac{2(1 - cos(k\Delta x))}{(\Delta x)^2}\phi(x) \tag{4.29}$$

$$\approx -k^2(1 - \frac{(k\Delta x)^2}{12})\phi(x), \tag{4.30}$$

gives slightly different results than derivation.

It could be given an estimate of the error subtracting (4.30) to (4.28) in modulus

$$\left| \frac{d^2\phi_i(x)}{dx^2} - \frac{\phi(x+\Delta x) - 2\phi(x) + \phi(x-\Delta x)}{(\Delta x)^2} \right| = k^2 \frac{(\Delta x)^2}{12} = \frac{2mE(\Delta x)^2}{12\hbar^2} \tag{4.31}$$

in accordance to appendix C the error goes to zero as a second order factor.
Then it is convenient to introduce a new quantity in order to study the errors trend

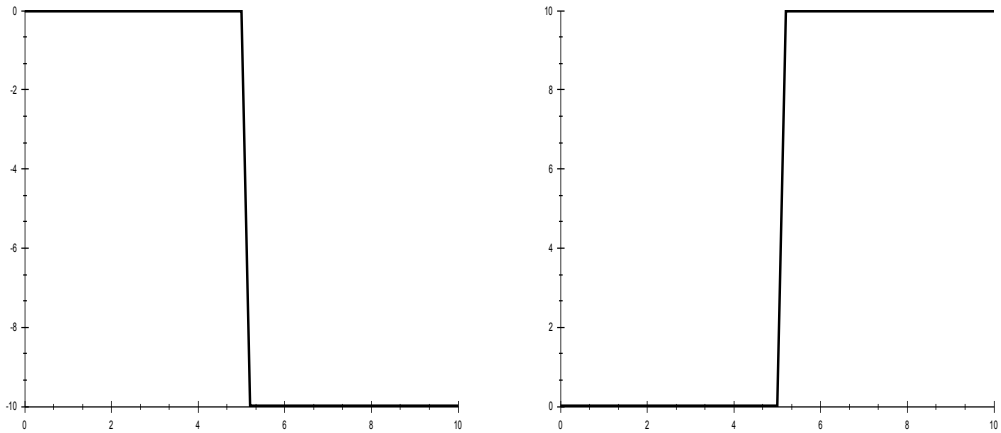
$$\epsilon = \frac{\sum_{i=1}^{N} |\phi(i) - \psi(i)|}{N}$$

where $\phi(i)$ is the numerical solution in a certain point of the lattice, $\psi(i)$ is the analytical solution (see appendix E), i denotes a specific lattice point and N is the total number of lattice points.

The evolution of such a quantity (ϵ) gives an estimate of the errors trend over time iterations. (ϵ refers to the case of zero potential shown in figure 4.1a).

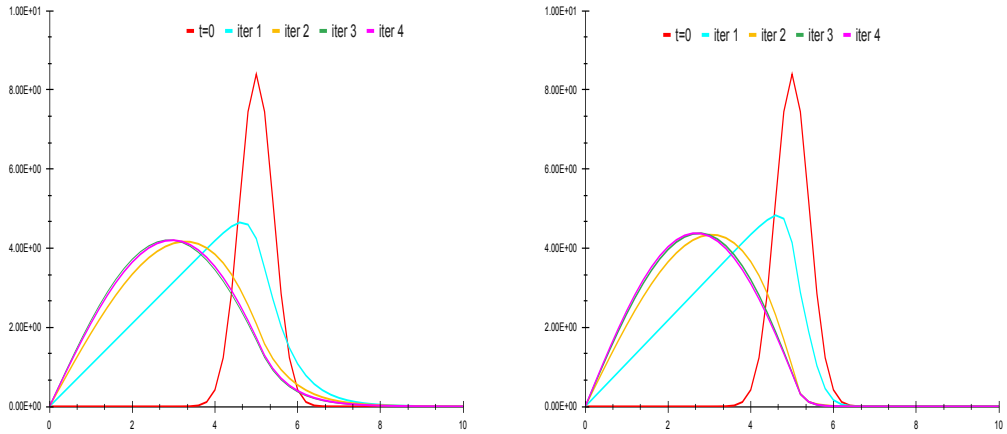
time iterations	ϵ
1	0.0111808
2	0.0002730
3	0.0207786
4	0.0000140
5	0.0002484
6	0.0000004
7	0.0000040
8	0.0000000
9	0.0000000
10	0.0000000

Errors slightly decrease over time iterations after few oscillations, that reminds of the accuracy of the Crank-Nicolson algorithm itself.



(a) V_0=-10 eV (b) V_0=10 eV

Figure 4.2: Energy($[\frac{(\hbar c)^2}{m(length)^2}] = \frac{eV\mathring{A}^2}{lenght^2}$) vs length. Well (left) and step (right) potential used in our calculations.



(a) Δt=0.1, Δx=0.2, V_0=10 eV (b) Δt=0.1, Δx=0.2, V_0=100 eV

Figure 4.3: Absolute value of the wave function $|\phi(x,t)|$ obtained by using the potential step of figure 4.2 for various time steps and space steps. Red plot represents the starting Gaussian packet $|\phi(x,0)|$ the other lines its time evolution. The starting Gaussian packet has the same features of that in figure 4.1

Step and well potential

Much more interesting are the cases of the square potential step (or well) whose analytical expressions are

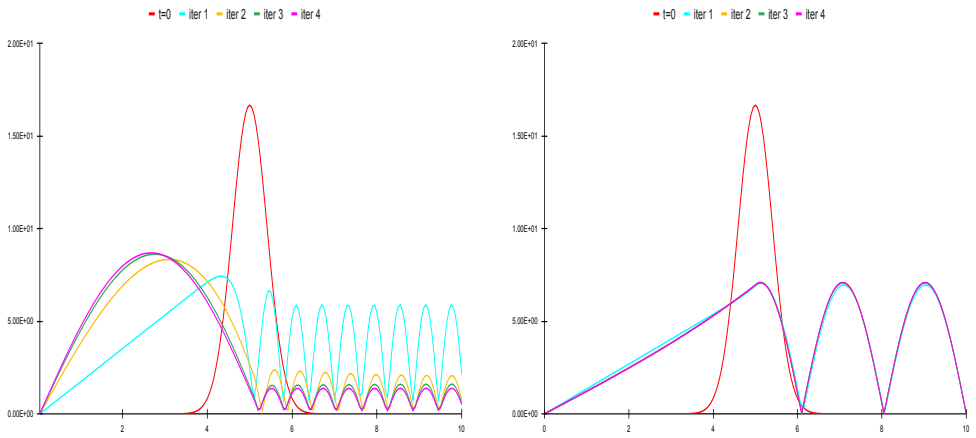
$$V(x) = \begin{cases} V_0 & \text{for } x > x_0 \\ 0 & \text{for } x < x_0 \end{cases}$$

where V_0 and x_0 stand respectively for the value of the potentials ($V_0 > 0$ step, $V_0 < 0$ well) and the position where the function change (figures 4.2 shows the potential plot). Tests have been carried out and their results are shown in figures 4.4 and 4.3.

The figures mentioned above show the evolution of the norm of a wave function during total time of $T = 4\Delta t$, where each iteration as been labelled with a diverse color and represent the norm of the wave function at different times.

In accordance to (4.22), the starting packet has been chosen to have $\mu = 3$ and $x_0 = 5$ while input value V_0 of the potential energy changes from negative values (well) to positive ones (step) and the magnitude is shown below each one figure.

It has been already pointed out that in certain situations (like the well potential) the wave function could develop certain rapid oscillations when the change in potential approaches that depends on the depth of the well itself, they are accurate and do not results from approximations or machine errors. [Gol67]



(a) Δt=0.001, Δx=0.05, V=-100eV (b) Δt=0.1, Δx=0.05, V=-10eV

Figure 4.4: Absolute value of the wave function $|\phi(x,t)|$ obtained by using the potential well of figure 4.2 for various time steps. The red line shows the initial Gaussian packet and the other lines its time evolution. On the left the time step is 0.001 while on the right panel is 0.1 in $\hbar$ units. Third picture has a shallower potential well.

The possibility that $E < V_0$ is also taken into account (Tunnel Effect)

$$k = \sqrt{\frac{2m(E-V_0)}{\hbar^2}} \in \mathbb{C}.$$

Solutions for the eigenvalues equation in the region where potential is V_0 are

$$\phi_i(x) = e^{\pm ikx},$$

where the boundary conditions have not yet been imposed.

In that case, analytical solutions (in the region potential where $V(x) \neq 0$) have the form

$$\phi(x,t) = \sum_i A_i \sin\left(k_i(x-L)\right)e^{-\frac{i}{\hbar}E_i t} = \sum_i A_i \sin\left(\frac{\sqrt{2m(E_i-V_0)}}{\hbar}(x-L)\right)e^{-\frac{i}{\hbar}E_i t}$$

where V_0 is space-dependent (from 4.27).

The absolute value of the wave function is damped in the case of step potential ($V_0 > E$) thanks to the sinusoidal factor

$$\sin\frac{\sqrt{2m(E-V_0)}}{\hbar}(x-L) = i\sinh\frac{\sqrt{2m(V_0-E)}}{\hbar}(x-L),$$

then the more the potential increases the faster the function tends to the zero value.

Contrary, in the well potential ($V_0 < 0 < E$), the absolute value of the wave function oscillates because of sinusoidal multiplicative factor.

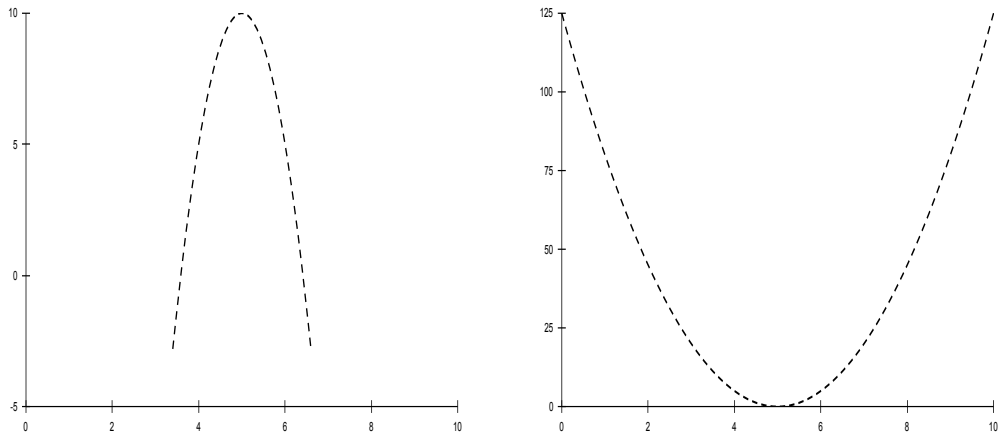
Error analysis

The error owing to the approximation of a second order derivative with a three points approximation (appendix A) in the numerical method could be obtained in the region of constant non zero potential in a similar fashion of (4.31):

$$\left| \frac{d^2\phi_i(x)}{dx^2} - \frac{\phi(x+\Delta x) - 2\phi(x) + \phi(x-\Delta x)}{(\Delta x)^2} \right| = \frac{2m|E-V_0|(\Delta x)^2}{12\hbar^2}. \tag{4.32}$$

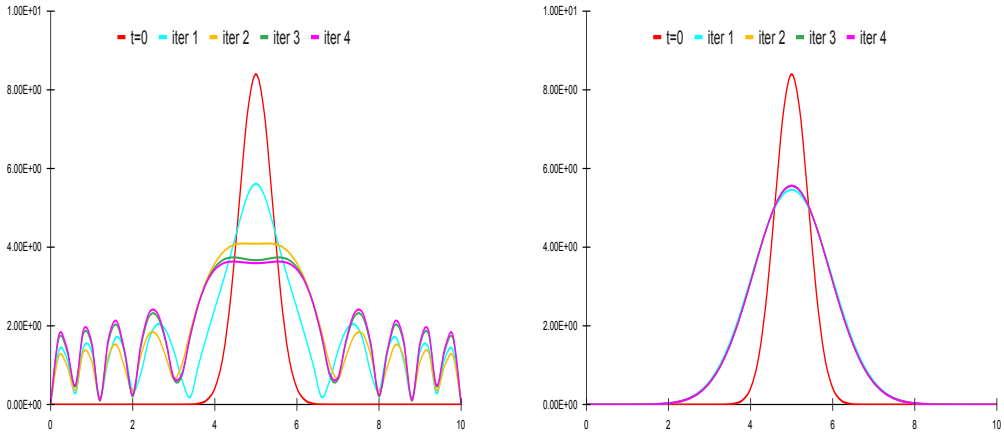
The second derivative approximation of three points has the space step (Δx) to serve as parameter. If the energy eigenstate rapidly varies in the space interval (Δx) the numerical approximation could gives an inaccurate estimate. In the eigenstates associated to an energy $E > V_0$ the number of oscillations is proportional to the difference (see fig.4.4) between the values of energy (E) and potential (V_0) so the errors found in (4.31) and (4.32) quantify those inaccuracies.

Also, it is practical to reintroduce the quantity ϵ to describe the errors trend of the step potential (figures 4.3a and 4.4c have been taken as example).



(a) V_0=-10eV (b) V_0=10eV

Figure 4.5: Energy($[\frac{(\hbar c)^2}{m(length)^2}] = \frac{eV\mathring{A}^2}{lenght^2}$) vs length for Parabolic potentials.



(a) V_0=-10 eV (b) V_0=10 eV

Figure 4.6: Absolute value of the wave function $\phi(x,t)$ obtained by using the parabolic potentials of figure 4.5. Red plot represents the starting Gaussian packet $\phi(x,0)$ the other lines its time evolution. The starting Gaussian packet got the same features of the one in figure 4.1

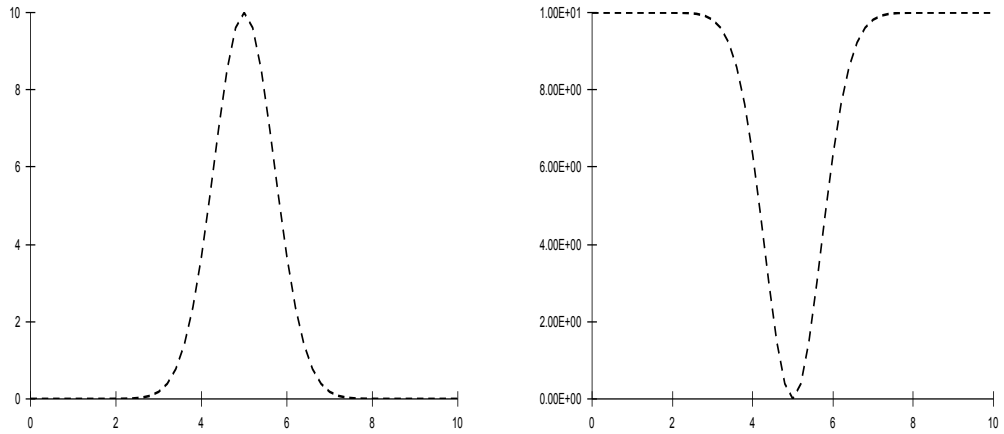
time iterations	$\epsilon(step)$	$\epsilon(well)$
1	0.0117916	0.3565752
2	0.0111099	0.0033694
3	0.1475116	0.0103695
4	1.5839038	0.0133698
5	0.0087906	0.0033590
6	0.0003740	0.0003906
7	0.0005755	0.0006395
8	0.0003351	0.0002140
9	0.0003398	0.0033698
10	0.0003377	0.0033698

Parabolic well and barrier potentials

Other interesting cases are the parabolic well and barriers, a much smoother function than the square ones, have the following expression

$$V(x) = \begin{cases} V_0((x-x_0)^2) & \text{for } V_0 > 0 \\ V_0((x-x_0)^2) - V_0 & \text{for } V_0 < 0 \end{cases}$$

where V_0 gives information about the orientation of the parabola and its width, while the



(a) V_0=-10eV (b) V_0=10eV

Figure 4.7: Energy($[\frac{(\hbar c)^2}{m(length)^2}] = \frac{eV\mathring{A}^2}{lenght^2}$) vs length for Gaussian potentials.

centre depends on the x_0 value.

Results from time evolution of the wave function in parabolic potentials (4.5 a-b) are shown in figure 4.6.

The potential plot of figure 4.5a shows a peculiar parabolic function which divides the graph in three region (starting from the left) alternating wells and barriers. The function is also centered ($x_0 = 5$) as the starting Gaussian values of the wave function.

The stationary solution peak in the centre is progressively lowered by the the action of the potential peak, but in the tails the solution develops peculiar oscillations which gradually degrade to the zero value at the boundaries.

This fluctuations are given by the negative applied potentials in the external regions, similarly of what happened in the square well situation where $V_0 < 0$.

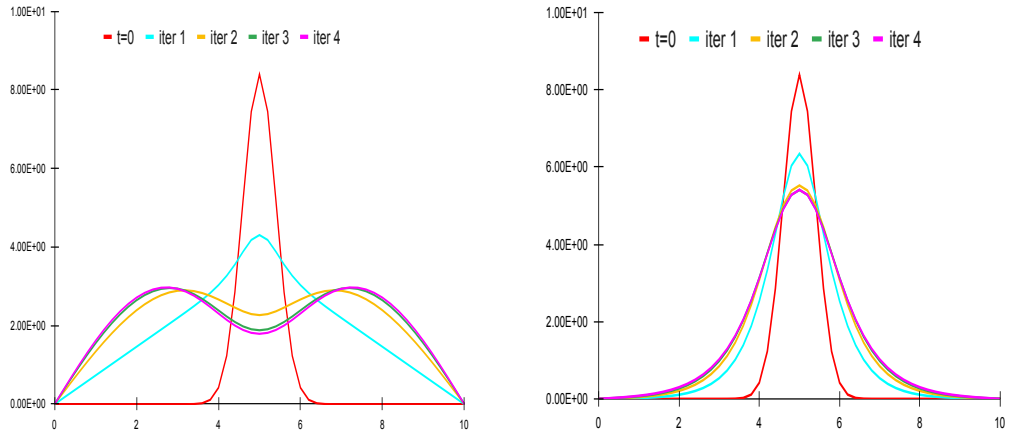
Figure 4.5b potential represents a parabolic well whose extremities tends to very large values, the potential itself encloses the wave function around the centre of the potential (which coincides with the wave function and so the lattice centre).

The stationary solution assumes the typical Gaussian function shape with larger spread until the greater potential ends up lowering to zero those values.

Gaussian well and barrier potentials

As a last example there are the Gaussian well and step potentials

$$V(x) = \begin{cases} V_0 \exp -(x - x_0)^2 & \text{for } V_0 > 0 \\ V_0 \exp -(x - x_0)^2 - V_0 & \text{for } V_0 < 0 \end{cases}$$



(a) Δt=0.1, Δx=0.2, V_0=-10eV (b) Δt=0.1, Δx=0.2, V_0=10eV

Figure 4.8: Absolute value of the wave function $|\phi(x,t)|$ obtained by using the Gaussian potentials of figure 4.7. Red plot represents the starting Gaussian packet $|\phi(x,0)|$ the other lines its time evolution. The starting Gaussian packet got the same features of the one in figure 4.1

where V_0 gives information about the orientation of the Gaussian and its width, while the centre depends on the x_0 value.

Results from time evolution of the wave function in Gaussian potentials (4.7) are shown in figure 4.8.

The potential plots of figure 4.7a-b show two Gaussian functions where $V_0 = \pm 10eV$ and $x_0 = 5$.

The first picture 4.8 represent the evolution of the stationary solution of the wavepacket which is dumped by the Gaussian potential (4.7a) in the centre of the lattice, similarly of what happened in the parabolic potential barrier, but contrary to the specified case, the function develops no oscillations because the Gaussian potential does not assume negative values going to the extremities.

Figure 4.8b represents a stationary solution peaked in the centre of the one dimensional lattice evolving with the typical Gaussian shape because of the confinement imposed by the potential itself which growing on the tails brings the solutions value to zero.

Conclusion

In the thesis, different approaches to the numerical treatment of parabolic partial differential equations have been discussed along with essential mathematical and computational features.
Diffusion equation and time-dependent Schrödinger equation have been analysed and, to find numerical solutions, two finite differences scheme have been applied.
The more intuitive and computationally affordable, explicit method, is also the less stable choice since the errors grow unboundedly outside the Courant limit condition but within those limits it has been proven to be convergent.
The implicit methods use instead the backward difference, and they are much more reliable because unconditionally stable, a possible drawback is due to the numerical intensive workload that consists of solving a system of equations on each time reiteration (this burden could be easily lessened with the algorithm introduced in the third chapter).
An alternative of the straightforward implicit algorithm is the Crank-Nicolson scheme of central difference, using the average of the second space derivatives, that is the most accurate scheme and also the most computationally demanding but nonetheless it was the first, and foremost, convenient choice.
Fortran codes applying, the previously exposed, finite difference schemes have been developed to study, strictly in one dimension, the classical diffusion equation and the quantum mechanical time-dependent Schrödinger equation, each solution has been discussed separately in the chapters.
Precise Dirichlet boundary conditions have been set to let evolve the starting Gaussian wavepacket in a carefully contained site in which a set of diverse potentials have been applied. For two of them, constant zero potential and piecewise constant potentials, comparisons with analytical solutions have been carried out.
In general, solutions tend to stabilize after a certain amounts of time to the minimum energy configuration because the net result of time evolution is to enhance the components with smaller eigenvalues of $\mathcal{H}$.
The errors analysis, carried out by comparison with the analytical known results, demonstrates the reliability of the algorithm developed, although every each computational method got pros and cons in terms of computational cheapness rather than stability and convergence, which are not independent features.

Possible future investigations consist of a more extensive study of the time-dependent evolution of Gaussian, Dirac delta function or Lorentzian wavepackets and their interactions with multiple and differently shaped barriers but also it could be developed second dimensional expansion to deal with a wide set of physical systems.

Appendix A

Mathematical steps

Numerical differentiation

Numerical differentiation describes algorithms for estimating the derivative of a function. Let us suppose that we we interested in the derivative at $x = 0$, $f'(0)$ (but the formula can be generalized for arbitrary x) and we know f on an equally-spaced lattice (h) of some x values:

$$f_n = f(x_n); x_n = nh(n = 0, \pm 1, \pm 2, ...),$$

We need to compute $f'(0)$ in terms of f_n.

First we expand f around $x = 0$ using a Taylor series:

$$f(x) = f_0 + xf' + \frac{x^2}{2!}f'' + \frac{x^3}{3!}f''' + ...$$

We note that:

$$f_{\pm 1} \equiv f(x = \pm h) = f_0 \pm hf' + \frac{h^2}{2}f'' \pm \frac{h^3}{6}f''' + O(h^4).$$

and so on for $f_{\pm 2}$, $f_{\pm 3}$, etc.

Simple subtraction can be carried out between f_1 and f_{-1},

$$f' = \frac{F_1 - F_{-1}}{2h} - \frac{h^2}{6}f''' + O(h^4) \approx \frac{f_1 - f_{-1}}{2h}.$$

This 3-points formula is exact because is a second-degree polynomial in the interval [-h,h]. This symmetric difference about $x = 0$ is more accurate than the below 2-points difference formulas:

$$f' = \frac{f_1 - f_0}{h} + O(h),$$

$$f' = \frac{f_0 - f_{-1}}{h} + O(h),$$

which assume that f is well approximated by a linear function over the intervals [0,h],[-h.0].

Formulas for higher derivatives can be obtained by taking appropriate combinations of equations $f_{\pm 1}$,$f_{\pm 2}$,etc. For our purposes we take into account the example of

$$f'' = \frac{f_1 - 2f_0 + f_{-1}}{h^2} + O(h^2).$$

that represents the action of $\frac{\delta^2}{(\Delta x)^2}$.

The entirety of this assumptions introduces in the algorithm the so-called truncation error related to the compatibility and convergence rate of the algorithm itself.

This particular truncation error implies that error decreases quadratically with grid size h.

High order methods are not necessarily always more accurate than low order methods; they simply have a more rapid convergence rate.

Direct algorithm for solving tri-diagonal matrix

Let us take the matrix equation

$$A\phi = b,$$

where A is a non singular NxN tri-diagonal matrix with non zero diagonal entries and b is a given column vector (N) of the form:

$$A = \begin{pmatrix} A_1^0 & A_1^+ & & & & & \\ A_2^- & A_2^0 & A_2^+ & & & & \\ & \cdot & \cdot & \cdot & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & & & A_{N-1}^+ \\ & & & & & A_N^- & A_N^0 \end{pmatrix}$$

The simple algorithm consist in calculating two quantities, we are going to define immediately, that is equivalent to the reduction Gaussian elimination of the matrix equation to a new matrix being upper-triangular with unit diagonal entries.

$$w_i = \frac{A_i^+}{A_i^0 - A^- w_{i-1}}, \qquad 2 \leq i \leq N-1$$

$$g_i = \frac{b_i - A_i^- g_{i-1}}{A_i^0 - A_i^- w_{i-1}}, \quad 2 \le i \le N,$$

where

$$w_1 = \frac{A_1^+}{A_1^0} \quad g_1 = \frac{b_1}{A_1^0}.$$

The components ϕ_i of the solution vector are the given recursively by the backward relation

$$\phi_i = g_i - w_i \phi_{i+1}, \quad \phi_N = g_N, \quad 1 \le i \le N-1.$$

Transposing the previous relation in its corresponding forward fashion we are able to identify the coefficients of the algorithm in section 3.1 **??** :

$$\phi_{i+1} = -\frac{1}{w_i}\phi_i + \frac{g_i}{w_i}$$

where $\alpha = \frac{1}{w_i}$ and $\beta = \frac{g_i}{w_i}$

Alternative algorithm, Crank-Nicolson

Chapter 3 alternative algorithm, from equation (3.13) to a nearly form like (3.3):

$\phi^{n+1} = \frac{1}{1+\frac{1}{2}\mathcal{H}\Delta t}[(1 - \frac{1}{2}\mathcal{H}\Delta t)\phi^n + S^n \Delta t]$,

$(1 + \frac{1}{2}\mathcal{H}\Delta t)\phi^{n+1} = (1 - \frac{1}{2}\mathcal{H}\Delta t)\phi^n + S^n \Delta t$,

the action of the operator $\mathcal{H}$ has already been defined in (2.6)

$\phi^{n+1} - \frac{1}{2}\mu(\phi_{i-1}^{n+1} + \phi_{i+1}^{n+1} - 2\phi_i^{n+1}) = \phi^n + \frac{1}{2}\mu(\phi_{i-1}^n + \phi_{i+1}^n - 2\phi_i^n) + S^n \Delta t$,

μ was previously defined in equation (2.2)

$(1 + \mu)\phi^{n+1} - \frac{1}{2}\mu(\phi_{i-1}^{n+1} + \phi_{i+1}^{n+1}) = (1 - \mu)\phi n + \frac{1}{2}\mu(\phi_{i-1}^n + \phi_{i+1}^n) + S^n \Delta t.$

Appendix B

Stability matters

Eigenvalues and eigenvectors of tri-diagonal matrix ($\mathcal{H}$)

When a tri-diagonal matrix is also a Toeplitz one (a matrix in which each descending diagonal from left to right is constant)

$$A = \begin{pmatrix} a & b & & & & & \\ c & a & b & & & & \\ & \cdot & \cdot & \cdot & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & & & b \\ & & & & & c & a \end{pmatrix}$$

then there is a simple closed-form solutions for its eigenvalues [Nos13]:

$$\alpha_\lambda = a + 2(bc)^{\frac{1}{2}} \cos(\frac{\lambda\pi}{N})$$

where $\lambda = 1, ..., N-1$ because of the boundary conditions on a lattice of $N+1$ points.

Also a, b and c are the constant values in the three main diagonals, if we take the example above, in the matrix we are going to study, the values respectively are -2, 1 and 1.

Eigenvalues of operator $\mathcal{H}$ defined by (2.6) are

$$\epsilon_\lambda = -\frac{\alpha_\lambda}{(\Delta x)^2},$$

$$\epsilon_\lambda = \frac{1}{(\Delta x)^2} 2(1 - cos(\frac{\lambda\pi}{N})),$$

rearranging the equation in a more appropriate fashion using the fundamental Pythagorean identity and the duplication formulas ($\cos 2\alpha = \cos^2\alpha - \sin^2\alpha$)

we get equation (2.13).

Correspondent eigenvectors are of the forms:

$$(\psi_\lambda)_i = [(\frac{b}{c})^{\frac{1}{2}} \sin^2(\frac{i\lambda\pi}{N}), ..., (\frac{b}{c})^{\frac{N}{2}} \sin^2(\frac{N\lambda\pi}{N})],$$

from which we obtain equation (2.12) after a slight rearrangement.[Nos13]

Stability of iterative algorithm

Iterative algorithm generates successive values starting from some initial value (ϕ^0 in our case) according to an algorithm of the type

$$\phi^{n+1} = f(\phi^n)$$

for instance to approximate a time evolution. Let us consider a linear iteration equation which can be written in matrix form as

$$\phi^{n+1} = A\phi^n$$

if the matrix is the same for all steps then

$$\phi^n = A^n\phi^0$$

(The eigenvalues of A are solutions of the eigenvalue equation $A\phi = \alpha\phi$).

Now, we have to deal with errors that can be exponentially enhanced if A has at least one eigenvalue whose module is greater than 1.

On the other hand the algorithm is conditionally stable if for all eigenvalues $|\alpha| \leq 1$ holds.

Courant number and stability of the explicit algorithm

The algorithm of the explicit scheme generate successive values starting from the initial value ϕ^0 according to an iteration prescription, written in matrix form, of the type

$$\phi^{n+1} = A\phi^n,$$

where $A = (1 - \mathcal{H}\Delta t)$.

As we have previously seen the stability analysis of the explicit scheme requires to find the eigenvalues of the A matrix which we can rewrite in the form

$$A = 1 - \mu\delta^2$$

(μ and δ^2 have been defined in (2.2) and (2.4)) where δ^2 matrix form is listed in section 2.1 .

where the eigenvalues of $\mu\delta^2$ are stated in the first paragraph of this appendix

$$\Delta t\epsilon_\lambda = \frac{\Delta t}{(\Delta x)^2} 2(1 - \cos(\frac{\lambda\pi}{N})),$$

hence the eigenvalues of A are given by

$$\alpha_\lambda = 1 - \Delta t\epsilon_\lambda$$

and to achieve stability we need (for all eigenvalues)

$$|1 - \Delta t\epsilon_\lambda| < 1$$

which is true if

$$-1 < 1 - \frac{4\Delta t}{(\Delta x)^2} \sin^2(\frac{\lambda\pi}{2N}) < 1.$$

The maximum of the sine function is 1 so the right hand inequality is fulfilled while from the left one we have the stability condition (Courant-Friedrichs-Lewy condition) [Cou28]:

$$\mu = \frac{\Delta t}{(\Delta x)^2} < \frac{1}{2}.$$

Stability of the implicit algorithm

Consider the implicit method in matrix notation,

$$A\phi^{n+1} = \phi^n + S^n\Delta t,$$

where A stands for $(1 + \mu\delta^2)$ and it can be solved by

$$\phi^{n+1} = A^{-1}\phi^n + A^{-1}S^n\Delta t.$$

The eigenvalues of A are (similarly to A in the explicit case)

$$\alpha_\lambda = 1 + \Delta t\epsilon_\lambda = 1 + \frac{4\Delta t}{(\Delta x)^2} sen^2(\frac{\lambda\pi}{2N}),$$

where ϵ_λ are the eigenvalues of $\mathcal{H}$ (2.6), and they have values slightly higher than 1.

Therefore eigenvalues of A^{-1} are

$$\alpha_\lambda^{-1} = \frac{1}{1 + \frac{4\Delta t}{(\Delta x)^2} \sin^2(\frac{\lambda\pi}{2N})},$$

then the implicit method is unconditionally stable since the modules of the eigenvalues are always less than 1.

Stability of Crank-Nicolson method

The stability analysis requires rewriting the algorithm (3.13) in a matrix notation:

$$\phi^{n+1} = \frac{1}{1 + \frac{1}{2}\mu\delta^2}[(1 - \frac{1}{2}\mu\delta^2)\phi^n + S^n\Delta t],$$

the eigenvalues are now

$$\alpha_\lambda = \frac{1 - \frac{1}{2}\Delta t\epsilon_\lambda}{1 + \frac{1}{2}\Delta t\epsilon_\lambda}.$$

Since $\Delta t\epsilon_\lambda > 0$ then $|\alpha_\lambda| < 1$ so we deduce the algorithm unconditional stability.

Appendix C

Error order analysis

Error order analysis in explicit method

Let us have a quick refresher on numerical integration of diffusion equation, if we use discrete values of time and space ($t_n = n\Delta t$ and $x_m = m\Delta x$, $m = 0, ..., N-1$) then the discretized derivatives:

$$\frac{\partial \phi}{\partial \phi} = \frac{\phi(t_{n+1}, x_m) - \phi(t_n), x_m}{\Delta t},$$

$$\frac{\partial^2 \phi}{\partial x^2} = \frac{\phi(t_n, x_{m+1}) + \phi(t_n, x_{m-1}) - 2\phi(t_n, x_m)}{\Delta x^2}$$

as we have done in the explicit method chapter.

Taylor series expansion of $\phi(t + \Delta t, x) - \phi(t, x)$ for the explicit algorithm (with a zero source function) gives

$$\phi(t + \Delta t, x) - \phi(t, x) = \frac{\Delta t}{(\Delta x)^2}\delta^2\phi(t, x) = \frac{\Delta t}{(\Delta x)^2}(\phi(t, x + \Delta x) + \phi(t, x - \Delta x) - 2\phi(t, x)),$$

making use of the diffusion equation we get

$$\frac{\Delta t}{(\Delta x)^2}((\Delta x)^2\frac{\partial^2}{\partial x^2}\phi(t, x) + \frac{(\Delta x)^4}{4!}\frac{\partial^4}{\partial x^4}\phi(t, x) + ...)$$

$$= \Delta t\frac{\partial^2}{\partial x^2}\phi(t, x) + \frac{(\Delta t \Delta x)^2}{4!}\frac{\partial^4}{\partial x^4}\phi(t, x) + ...$$

Errors obtained are proportional to the time step and the square of space step:

$$\Delta\phi = O(\Delta t) + O(\Delta x^2).$$

Error order analysis in implicit method

Applying the same conventions as the previous paragraph we could find the order error of the implicit method:

$$\begin{aligned}
& \phi(t+\Delta t,x)-\phi(t,x)=\frac{\Delta t}{(\Delta x)^2}\delta^2\phi(t+\Delta t,x) \\
= & \frac{\Delta t}{(\Delta x)^2}(\phi(t+\Delta t,x+\Delta x)+\phi(t+\Delta t,x-\Delta x)-2\phi(t+\Delta t,x)) \\
= & \frac{\Delta t}{(\Delta x)^2}((\Delta x)^2\frac{\partial^2}{\partial x^2}\phi(t,x)+\frac{(\Delta x)^4}{4!}\frac{\partial^4}{\partial x^4}\phi(t,x)+...) \\
+ & \frac{(\Delta t)^2}{(\Delta x)^2}((\Delta x)^2\frac{\partial}{\partial t}\frac{\partial^2}{\partial x^2}\phi(t,x)+\frac{(\Delta x)^4}{4!}\frac{\partial^4}{\partial x^4}\phi(t,x)+...) \\
= & \Delta t\phi(t,x)+\Delta t^2\frac{\partial}{\partial t}\phi(x,t)+\frac{\delta t\delta x^2}{4!}(\frac{\partial^4}{\partial x^4}\phi(x,t)+\Delta t\frac{\partial^4}{\partial x^4}\phi(x,t))+...
\end{aligned}$$

The error is of the same order as before:

$$\Delta\phi=O(\Delta t)+O(\Delta x^2).$$

Error order analysis in Crank-Nicolson method
As we have done before

$$\phi(t+\Delta t,x)-\phi(t,x)=\frac{\Delta t}{(\Delta x)^2}\delta^2\frac{1}{2}(\phi(t+\Delta t,x)+\phi(t,x)),$$

and from a comparison with the exact Taylor series the error order appears to be a second order in time:

$$\Delta\phi=O(\Delta t^2)+O(\Delta x^2).$$

Appendix D

Fortran code

The FORTRAN code to find the solutions of time-dependent Schrödinger equation.

```
C
C TIME DEPENDENT SCHROEDINGER EQUATION FOR AN ELECTRON
C
C
      IMPLICIT REAL*8(A–B,D–H,O–Z)
      IMPLICIT COMPLEX*16(C)
C
      PARAMETER (NDIM=1000)
C
      DIMENSION CPHI(NDIM)
      DIMENSION EXACT(NDIM)
C
      CALL INPUT
      CALL PAKET(CPHI)
      CALL POTENTIAL
      CALL ALPHAGAMMA
C
      CALL SOLVING(CPHI)
C
      STOP
      END
C———————————————————————————————
      SUBROUTINE INPUT
C———————————————————————————————
C
      IMPLICIT REAL*8(A–B,D–H,O–Z)
```

```
      IMPLICIT COMPLEX*16 (C)
C
      PARAMETER (NDIM=1000)
C
      COMMON/RDATA/DR,R(NDIM),NPOINT
      COMMON/TDAT/DT,NTIME,IPRINT
      COMMON/PAK/APAK,BPAK,IPAK
      COMMON/PT/APOT,BPOT,V(NDIM),KPOT
      COMMON/CNST/PI,HBC,ELMASS,ZERO,UNO,CI
C
       !READ(5,99)
       !READ(5,99)
      READ(5,*) RMAX,DR
      WRITE(6,1000) RMAX,DR
      NPOINT=INT(RMAX/DR+0.0001) + 1
      DO I=1,NPOINT
        R(I)=FLOAT(I-1)*DR
      ENDDO
C
       !READ(5,99)
      READ(5,*) DT,NTIME,IPRINT
      WRITE(6,1001) DT,NTIME,IPRINT
C
       !READ(5,99)
      READ(5,*) IPAK
      WRITE(6,1002) IPAK
C
       !READ(5,99)
      READ(5,*) APAK,BPAK
      WRITE(6,1000) APAK,BPAK
C
       !READ(5,99)
      READ(5,*) KPOT
      WRITE(6,1002) KPOT
C
       !READ(5,99)
      READ(5,*) APOT,BPOT
      WRITE(6,1003) APOT,BPOT
C
C CONSTANTS
C
```

```
      PI=4.0*ATAN(1.0)
      HBC=197.327053                  ! MeV fm
      HBC=HBC*10.                     ! eV ANGSTROM
      ELMASS=0.51099895 * 10**6       ! eV
      ZERO=0.0
      UNO=1.0
      CI=CMPLX(ZERO,UNO)
C
      RETURN
 99   FORMAT(1X)
 1000 FORMAT(2X,4(F10.5,2X))
 1001 FORMAT(2X,1PE12.5,2(2X,I6))
 1002 FORMAT(2X,I6)
 1003 FORMAT(2X,1PE12.5,1x,F10.5)
C
      END
C-------------------------------------
      SUBROUTINE PAKET(CPHI)
C-------------------------------------
C
C THE INITIAL WAVE PACKET
C
C
      IMPLICIT REAL*8(A-B,D-H,O-Z)
      IMPLICIT COMPLEX*16 (C)
C
      PARAMETER(NDIM=1000)
C
      COMMON/RDATA/DR,R(NDIM),NPOINT
      COMMON/PAK/APAK,BPAK,IPAK
      COMMON/CNST/PI,HBC,ELMASS,ZERO,UNO,CI
C
      DIMENSION CPHI(NDIM)
C
C DIRCHLET BOUNDARY CONDITIONS
C
      CPHI(1)=CMPLX(ZERO,ZERO)
      CPHI(NPOINT)=CPHI(1)
C
C GAUSSIAN PACKET
C
```

```
      IF (IPAK.EQ.1) THEN
       DO I=2,NPOINT-1
        RPART=EXP(-APAK*(R(I)-BPAK)**2)
        CPHI(I)=CMPLX(RPART,ZERO)
       ENDDO
      ENDIF
C
C  LORENTZIAN PACKET
C
       IF (IPAK.EQ.2) THEN
        DO I=2,NPOINT-1
         RPART=(0.5*APAK)/(PI*((R(I)-BPAK)**2+(0.5*APAK)**2))
         CPHI(I)=CMPLX(RPART,ZERO)
        ENDDO
       ENDIF

      CALL NORM(CPHI)
C
      WRITE(6,1000)
      DO I=1,NPOINT
       PHI=SQRT( CPHI(I)*CONJG(CPHI(I)) )
       WRITE(6,1001) R(I),PHI
      ENDDO
C
      RETURN
 1000 FORMAT(2X,'STARTING PACKET',
     ^/,7X,'R',10X,'|PHI|',10X,'|EXACT|')
 1001 FORMAT(2X,F10.5,3(2X,1PE12.5))
C
      END
C-----------------------------------------------------------
      SUBROUTINE NORM(CPHI)
C-----------------------------------------------------------
C
C NORMALISING THE SOLUTION
C
      IMPLICIT REAL*8(A-B,D-H,O-Z)
      IMPLICIT COMPLEX*16 (C)
C
      PARAMETER (NDIM=1000)
C
```

```
      COMMON/RDATA/DR,R(NDIM),NPOINT
      DIMENSION CPHI(NDIM),WINT(NDIM)
C
      XNORM=0.0
      DO I=1,NPOINT
        WINT(I)=CPHI(I)*CONJG(CPHI(I))
      ENDDO
      XNORM=SIMP1(1,NPOINT,DR,WINT)
      XNORM=SQRT(FLOAT(NPOINT)/XNORM)
      DO I=1,NPOINT
        CPHI(I)=XNORM*CPHI(I)
      ENDDO
C
      RETURN
      END
C-------------------------------------------
      SUBROUTINE POTENTIAL
C-------------------------------------------
C
C POTENTIAL(S) VALUES OF IPRINT
C
C
      IMPLICIT REAL*8(A-B,D-H,O-Z)
      IMPLICIT COMPLEX*16 (C)
C
      PARAMETER(NDIM=1000)
C
      COMMON/TDAT/DT,NTIME,IPRINT
      COMMON/RDATA/DR,R(NDIM),NPOINT
      COMMON/PAK/APAK,BPAK,IPAK
      COMMON/CNST/PI,HBC,ELMASS,ZERO,UNO,CI
      COMMON/PT/APOT,BPOT,V(NDIM),KPOT
C
        IF(KPOT.EQ.0) THEN          !CONSTANT
        DO I=1,NPOINT
          V(I)=APOT
        ENDDO
        ELSEIF(KPOT.EQ.1) THEN      !WELL
        BOX=BPOT*R(NPOINT)
        DO I=NPOINT,BOX,-1
          V(I)=APOT
```

```
          IF((R(I).LE.BOX)) V(I)=0.0
         ENDDO
        ELSEIF(KPOT.EQ.2) THEN     !PARABOLIC
         BOX=BPOT*R(NPOINT)
         DO I=1,NPOINT
          V(I)=APOT*((R(I)-BOX)**2)    !-APOT
         ENDDO
        ELSEIF(KPOT.EQ.3) THEN     !GAUSSIAN
         DO I=1,NPOINT
          BOX=BPOT*R(NPOINT)
          V(I)=APOT*EXP(-(R(I)-BOX)**2)  !- APOT
         ENDDO
        ENDIF
C
        RETURN
 1000 FORMAT(2X,'CONSTANT POTENTIAL')
 1001 FORMAT(2X,'SQUARE WELL')
        END
C-------------------------------------
        SUBROUTINE ALPHAGAMMA
C-------------------------------------
C
C CALCULATING ALPHA AND GAMMA ONCE FOREVER
C
        IMPLICIT REAL*8(A-B,D-H,O-Z)
        IMPLICIT COMPLEX*16 (C)
C
        PARAMETER(NDIM=1000)
C
        COMMON/TDAT/DT,NTIME,IPRINT
        COMMON/RDATA/DR,R(NDIM),NPOINT
        COMMON/CNST/PI,HBC,ELMASS,ZERO,UNO,CI
        COMMON/PT/APOT,BPOT,V(NDIM),KPOT
        COMMON/AG/CALPHA(NDIM),CGAMMA(NDIM)
C
        UDTH=(DR*DR)/DT
        UDTH=UDTH*2.0*ELMASS/(HBC*HBC)
C
        CAP=1.0
        CAM=CAP
        CAZ= 2.0*CI*UDTH -2.0 - V(NPOINT)*UDTH*DT
```

```
C
      CALPHA(NPOINT)=ZERO
      CGAMMA(NPOINT)=0  !-1.0/CAZ
      DO I=NPOINT,2,-1
        CALPHA(I-1)=CGAMMA(I)*CAM
        CAZ= 2.0*CI*UDHT -2.0 - V(I-1)*UDTH*DT
        CGAMMA(I-1)=-1.0/(CAZ+CAP*CALPHA(I-1))
      ENDDO
C
      RETURN
      END
C-------------------------------------------
      SUBROUTINE SOLVING(CPHI)
C-------------------------------------------
C
C ROUTINE TO SOLVE THE EQUATIONS
C
      IMPLICIT REAL*8(A-B,D-H,O-Z)
      IMPLICIT COMPLEX*16 (C)
C
      PARAMETER(NDIM=1000)
C
      COMMON/CNST/PI,HBC,ELMASS,ZERO,UNO,CI
      COMMON/RDATA/DR,R(NDIM),NPOINT
      COMMON/TDAT/DT,NTIME,IPRINT
      COMMON/PAK/APAK,BPAK,IPAK
      COMMON/PT/APOT,BPOT,V(NDIM),KPOT
      COMMON/AG/CALPHA(NDIM),CGAMMA(NDIM)
C
      DIMENSION CPHI(NDIM),CBETA(NDIM),CHI(NDIM),EXACT(NDIM)
C
      UDTH=(DR*DR)/DT
      UDTH=UDTH*2.0*ELMASS/(HBC*HBC)
      CFACT=4.0*CI*UDTH
      CVU=0.5*CI*DT

C
C  LOOP OVER THE TIME
C
      DO ITIME=1,NTIME
C
```

```
C FIND BETA
C
      CBETA(NPOINT)=CMPLX(ZERO,ZERO)
      DO I=NPOINT,2,-1
        CBETA(I-1)=CGAMMA(I)*(CBETA(I)-CFACT*CPHI(I))
      ENDDO
C
C FIND CHI
C
      CHI(1)=CMPLX(ZERO,ZERO)
      CHI(NPOINT)=CHI(1)
      DO I=1,NPOINT-1
        CHI(I+1)=CALPHA(I)*CHI(I)+CBETA(I)
      ENDDO
C
C FIND PHI
C
      CPHI(1)=CMPLX(ZERO,ZERO)
      CPHI(NPOINT)=CPHI(1)
      DO I=2,NPOINT-1
        CPHI(I) = 0.5*CHI(I)
     ^          + (CHI(I+1) + CHI(I-1) - 2.0*CHI(I))/CFACT
     ^          + 0.5*CVU*V(I)*CHI(I)
      ENDDO

      CALL ENERGY(CPHI)

C FIND EXACT
      RPART=(2*SQRT(PI)/SQRT(1/APAK))*EXP(-APAK*((R(I)-BPAK)**2))
      CPHI(I)=CMPLX(RPART,ZERO)
      CALL NORM(CPHI)
      EXACT(1)=CMPLX(ZERO,ZERO)
      EXACT(NPOINT)=EXACT(1)
      DO I=2,NPOINT-1
      EXACT(I)=CPHI(I)*EXP(-CI*E*DT*ITIME/HBC)
      ENDDO
C
      CALL NORM(CPHI)
      CALL NORM(EXACT)
C
      T=DT*FLOAT(ITIME)
```

```
          CALL GETOUT(CPHI,T,ITIME,EXACT)
C
        ENDDO
C
        RETURN
        END
C-------------------------------------
        SUBROUTINE GETOUT(CPHI,T,ITIME,EXACT)
C-------------------------------------
C THIS ROUTINE IS ONLY FOR THE OUTPUT
C PRINT EVERY IPRINT TIMES
C
        IMPLICIT REAL*8(A-B,D-H,O-Z)
        IMPLICIT COMPLEX*16 (C)
C
        PARAMETER(NDIM=1000)
C
        COMMON/TDAT/DT,NTIME,IPRINT
        COMMON/RDATA/DR,R(NDIM),NPOINT
        COMMON/PAK/APAK,BPAK,IPAK
        COMMON/PT/APOT,BPOT,V(NDIM),KPOT
C
        DIMENSION CPHI(NDIM),EXACT(NDIM)
C
        ITEST=(ITIME/IPRINT)*IPRINT
        IF(ITEST.EQ.ITIME) THEN
C
         WRITE(6,1000) ITIME,T
C


C
          DO I=1,NPOINT
            PHI=SQRT( CPHI(I)*CONJG(CPHI(I)) )
            REXACT=SQRT( EXACT(I)*CONJG(EXACT(I)) )
            WRITE(6,1001) R(I),PHI,REXACT
          ENDDO
C
        ENDIF
C
        RETURN
 1000 FORMAT(2X,'ITER'1X,I5,2X,'TIME / HBAR',2X,F10.5,
```

```
     ^/,7X,'R',10X,'|PHI|',10X,'|EXACT|')
 1001 FORMAT(2X,F10.5,3(2X,1PE12.5))
      END
C*************************************************************
      FUNCTION SIMP1(IU,IO,DR,WF)
C*************************************************************
C
C     BETTER SIMPSON INTEGRATION
C     ABRAMOWITZ 25.4.6
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION WF(1)
C
      S=WF(IU)+WF(IO)
      DO J1=IU+1,IO-1,2
        S=S + WF(J1)*4.0
      ENDDO
      DO J1=IU+2,IO-1,2
        S=S + WF(J1)*2.0
      ENDDO
      SIMP1=S*DR/3.0
      RETURN
      END
C-------------------------------------------------------------------
       SUBROUTINE ENERGY(CPHI)
C-------------------------------------------------------------------
C
      IMPLICIT REAL*8(A-B,D-H,O-Z)
      IMPLICIT COMPLEX*16 (C)
C
      PARAMETER(NDIM=1000)
C
      COMMON/CNST/PI,HBC,ELMASS,ZERO,UNO,CI
      COMMON/RDATA/DR,R(NDIM),NPOINT
      COMMON/TDAT/DT,NTIME,IPRINT
      COMMON/PT/APOT,BPOT,V(NDIM),KPOT
      COMMON/AG/CALPHA(NDIM),CGAMMA(NDIM)

C
      DIMENSION CPHI(NDIM)
C
```

```
FACT=HBC*HBC/(2.0*ELMASS)
PE=0
KE=0
DO I=1,NPOINT-1          !INTEGRATE USING TRAPEZOIDAL RULE
  PE=PE+V(I)*(CPHI(I)**2)
  KE=KE+CONJG(CPHI(I))*(CPHI(I-1)-2*CPHI(I)+CPHI(I+1))
ENDDO
  KE=-FACT*KE/(DR)**2
  E=PE+KE
  WRITE(6,*)  '------>',E
RETURN
END
```

Appendix E

Insights into chapter 4

Alternative method for $\mathcal{U}$ operator

From 4.10 it could be defined $U = [\frac{1-\frac{i}{2}\mathcal{H}_j\Delta T}{1+\frac{i}{2}\mathcal{H}_j\Delta T}]$, the so-called propagator operator.

If $\mathcal{H}$ (nor V) is not time-dependent then the exact solution of the time-dependent Schroedinger equation is

$$\phi(x,t) = U(t)\phi(x,0) = e^{-\frac{i\mathcal{H}t}{\hbar}}\phi(x,0),$$

since U (propagator operator) is unitary: $UU^\dagger = 1$ then it conserves the norm of the wave function $|\phi(x,t)|^2 = |\phi(x,0)|^2$.

After applying the discretization to the equation above, we obtain

$$\phi_j^{n+1} = e^{-\frac{i\mathcal{H}\Delta T}{2}} e^{\frac{i\mathcal{H}\Delta T}{2}} \phi_j^n,$$

multiplying the equation by $U^\dagger(\frac{\Delta T}{2})$ we obtain

$$e^{\frac{i\mathcal{H}\Delta T}{2}}\phi_j^{n+1} = e^{-\frac{i\mathcal{H}\Delta T}{2}}\phi_j^n,$$

and by expanding the exponential functions into a taylor series and cutting off after the second order term

$$\exp A = 1 + A + \frac{A^2}{2!} + \frac{A^3}{3!} + ...$$

where A is any matrix function, we obtain

$$[1 + \frac{i}{2}\mathcal{H}\Delta T]\phi_j^{n+1} = [1 - \frac{i}{2}\mathcal{H}\Delta T]\phi_j^n.$$

Then we have found a unitary method to approximate U (known as Cayley's form)

$$U = e^{-i\mathcal{H}\Delta T} \approx \frac{1 - \frac{i}{2}\mathcal{H}\Delta T}{1 + \frac{i}{2}\mathcal{H}\Delta T}.$$

[Pre93] [Cra47]

Evolution of a Gaussian wavepacket in a zero potential

The analytical solution of the evolution of $\phi(x,t)$ will be of great use to test the accuracy of the numerical treatment in Chapter 4.

Starting from a superposition of plane waves at certain time $t = 0$

$$\phi(x,0) = C\int_{-\infty}^{+\infty} g(k)e^{ik(x-x_0)}dk,$$

where $C \in \mathbb{R}$ is a normalisation constant and

$$g(k) = e^{-\frac{\alpha^2}{4}(k-k_0)^2},$$

$\alpha \in \mathbb{R}$.

The initial wavefunction is the Fourier transform of a complex function $g(k)$. Solutions for the integrals above are well-known

$$\phi(x,0) = C\frac{2\sqrt{\pi}}{\alpha}e^{ik_0x}e^{-\frac{(x-x_0)^2}{\alpha^2}},$$

so imposing the normalization condition

$$\int_{-\infty}^{+\infty}|\phi(x,0)|^2 = 1 \Rightarrow C = \frac{\sqrt{\alpha}}{(2\pi)^{\frac{3}{4}}}.$$

The function of starting wawepacket is so obtained

$$\phi(x,0) = \left(\frac{2}{\pi\alpha^2}\right)^{\frac{1}{4}}e^{ik_0x}e^{-\frac{(x-x_0)^2}{\alpha^2}} \tag{E.1}$$

and the probability distribution is given by

$$|\phi(x,0)|^2 = \sqrt{\frac{2}{\pi\alpha^2}}e^{-\frac{2(x-x_0)^2}{\alpha^2}}. \tag{E.2}$$

Time evolution of (E.1) in a zero potential (particular case of constant potential) is given by (4.23) in section 4.3

$$\phi(x,t) = e^{-\frac{it}{\hbar}\mathcal{H}}\phi(x,0) \tag{E.3}$$

then

$$\phi(x,t) = \frac{\sqrt{\alpha}}{2^{\frac{3}{4}}\pi^{\frac{1}{4}}}\frac{\exp\left(-\frac{\alpha^2k_0^2}{4}\right)}{\sqrt{\frac{\alpha^2}{4}+\frac{i\hbar t}{2m}}}\exp\left[\frac{(\frac{\alpha^2}{2}k_0+i(x-x_0))^2}{\alpha^2+\frac{2i\hbar t}{m}}\right]. \tag{E.4}$$

$k_0 = 0$ has been imposed in the examples.

Trapezoidal rule

The trapezoidal rule works by approximating the region under the graph of the integrating function as a trapezoid and calculating its area as it follows

$$\int a^b f(x)dx \approx (b-a) * \frac{1}{2}(f(a)+f(b)).$$

The integral could be better approximated by partitioning the interval domain and applying the above rule to each subinterval

$$\int a^b f(x)dx \approx \sum_{k=1}^{N} \frac{f(x_{k-1})+f(x_k)}{2}\Delta x_k.$$

Extended Simpson's rule for integration

Simpson's rules consist of several approximations for definite integrals, the most basic of these rules, called Simpson's 1/3 rule

$$\int_a^b f(x)dx \approx \frac{b-a}{6}[f(a)+4f(\frac{a+b}{2})+f(b)],$$

If the 1/3 rule is applied to n equal subdivisions of the integration range [a, b], one obtains the extended Simpson's rule, particularly useful in situation of high oscillations or lack of derivatives.

The extended rule deals with the breaking of the interval [a,b] into $n > 2$ small subintervals, Simpson's rule is then applied to each subinterval, with the results being summed to produce an approximation for the integral over the entire interval

$$\int_a^b f(x)dx = \frac{h}{3}[f(a)+4\sum_{k=1}^{n/2} f(x_{2k-1})+2\sum_{k=1}^{n/2-1} f(x_{2k})+f(b)] - \frac{nh^5}{90}f^{(4)}(\chi),$$

where $h=(a-b)/n$ is the step length and χ is a number between a and b. [Abr64]

Bibliography

[Abr64] M. Abramowitz, I.A. Stegun, Handbook of mathematical functions, Dover publications Inc., New York (1964).

[Coh73] C. Cohen-Tannoudji, B. Diu, F. Laloe, Quantum Mechanics vol.1-2, Wiley-VCH Verlag, Weinheim (1973).

[Cou28] R. Courant, K. Friedrichs, H. Lewy, On the partial difference equations of mathematical physics, Mathematische Annalen Springer, Berlin (1928).

[Cra47] J. Crank, P. Nicolson A practical method for numerical evaluation of solutions of partial differential equations of the heat conduction type, Mathematical Proceedings of the Cambridge Philosophical Society, Cambridge University Press, Cambridge (1947).

[Cra75] J. Crank, The mathemtics of diffusion, Clarendon Press, Oxford (1975).

[Fox68] L. Fox, D.F. Mayers, Computing Methods for Scientists and Engineers, Clarendon Press, Oxford (1968).

[Fox62] L. Fox, Numerical solution of ordinary and partial differential equations, Pergamo Press, Oxford (1962).

[Gol67] A. Goldberg, H. M. Schey, and J. L. Schwartz, Computer-Generated Motion Pictures of One-Dimensional Quantum-Mechanical Transmission and Reflection Phenomena American Journal of Physics 35, (1967).

[Hig02] N. J. Highman, Accuracy and stability of numerical algorithms (second edition), Society for Industrial and Applied Mathematics, Philadephia (2002).

[Koo86] S. E. Koonin, D. C. Meredith, Computational Physics: Fortran Version, CRC Press, Baton Rouge (1986).

[Lan17] H. P. Langtangen, S. Linge, Finite Difference Computing with PDEs: A Modern Software Approach, Springer International Publishing, Berlin (2017).

[Lan16] H. P. Langtangen, Finite Difference Computing with Exponential Decay Models, Springer International Publishing, Berlin (2016).

[Lee03] H. J. Lee, W. E. Schiesser, Ordinary and Partial Differential Equation Routines in C, C++, Fortran, Java, Maple, and MATLAB, Chapman & Hall CRC, Boca Raton (2003).

[Mit69] A.R. Mitchell, Computational methods in partial differential equations, Wiley, New York (1969).

[Nos13] S. Noschese, L. Pasquini, L. Reichel, Tridiagonal Toeplitz matrices: Properties and novel applications, John Wiley & Sons, New Jersey (2013).

[Ols16] L. Olsen-Kettle, Numerical Solution of Partial Differential Equations, lecture notes (2016) unpublished, *https* : $//espace.library.uq.edu.au/data/UQ_2$39427/$Lectures_Book.pdf$.

[Pre93] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, Numerical Recipes in Fortran 77 The Art of Scientific Computing, Cambridge University Press, Cambridge (1993).

[Sch10] P.O.J. Scherer, Computational physics simulation of classical and quantum systems, Springer, Berlin (2010)

[Smi65] G.D. Smith, Numerical solution of partial differential equations, Oxford University Press, Oxford (1965).

[Tho95] J.W. Thomas, Numerical Partial Differential Equations: Finite Difference Methods, Springer-Verlag, Berlin (1995).

[Var62] R. Varga, Matrix iterative analysis, Prentice-Hall, Englewood Cliffs (1962).

[Vol13] J. Volker, Numerical Methods for Partial Differential Equations, lecture notes (2013) unpublished, *https* : $//www.wias - berlin.de/people/john/LEHRE/NUM.PDE.FUB/num - pde - fub.pdf$.

[Wil61] J.H. Wilkinson, Error Analysis of Direct Methods of Matrix Inversion, Journal of the ACM, New York (1961).