Numerical Simulation of the Time-Dependent Schrödinger Equation: Crank–Nicolson Method

Alexis F. Espinoza Q.

University of Talca

20 de julio de 2025

Professor: Américo Cuchillo Flores

Contents

Schrödinger Equation

The time-dependent Schrödinger equation for a free particle ($V = 0$ or $V =$ constant) is given by:

$$i\hbar\frac{\partial\psi}{\partial t} = -\frac{\hbar^2}{2m}\nabla^2\psi$$

A Gaussian wave packet is proposed as the initial condition, confined within a two-dimensional domain (2D quantum box), to numerically solve the Schrödinger equation using the Crank–Nicolson method.

Schrödinger Equation in 2D

Developing the Laplacian operator:

$$i\hbar\frac{\partial\psi(x,y,t)}{\partial t} = -\frac{\hbar^2}{2m}\left(\frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial y^2}\right)$$

We work with boundary conditions (infinite potential walls):

$$\psi(0,y,t) = \psi(L,y,t) = \psi(x,0,t) = \psi(x,L,t) = 0$$

Initial condition (Gaussian wave packet with momentum $\vec{p} = \hbar\vec{k}$):

$$\psi(x,y,0) = e^{-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}} \cdot e^{i(k_{0x}x+k_{0y}y)}$$

Spatial and Temporal Discretization

A rectangular grid of $(N_x + 2) \times (N_y + 2)$ points is defined, including boundaries.
Spatial steps:

$$\Delta x = \frac{L}{N_x + 1}, \quad \Delta y = \frac{L}{N_y + 1}$$

with

$$x_i = i\Delta x, \quad y_j = j\Delta y, \quad i = 1, ..., N_x, \quad j = 1, ..., N_y$$

2D Laplacian using centered finite differences:

$$\nabla^2 \psi_{i,j} \approx \frac{\psi_{i+1,j} - 2\psi_{i,j} + \psi_{i-1,j}}{\Delta x^2} + \frac{\psi_{i,j+1} - 2\psi_{i,j} + \psi_{i,j-1}}{\Delta y^2}$$

Temporal Discretization - Crank-Nicolson

Temporal discretization with step Δt. Use of differentials: $dt \approx \Delta t$, $dx \approx \Delta x$, $dy \approx \Delta y$.
Crank-Nicolson (implicit, unitary):

$$\frac{\psi^{n+1} - \psi^n}{\Delta t} = -\frac{i\hbar}{2m}\left(\nabla^2\psi^{n+1} + \nabla^2\psi^n\right)$$

Multiply by Δt, and rearrange as:

$$\left(I + s\nabla^2\right)\psi^{n+1} = \left(I - s\nabla^2\right)\psi^n \quad \text{, with} \quad s = \frac{i\hbar\Delta t}{4m\Delta x^2}$$

2D Laplacian via Kronecker Product

We define:

$$D_x = \text{tridiagonal matrix of centered finite differences in } x$$
$$D_y = \text{tridiagonal matrix of centered finite differences in } y$$
$$I_x = \text{identity matrix of size } N_x \times N_x$$
$$I_y = \text{identity matrix of size } N_y \times N_y$$

Full Laplacian:

$$L = I_y \otimes D_x + D_y \otimes I_x$$

The Crank–Nicolson evolution matrices are given by:

$$A = I + sL, \quad B = I - sL$$

The linear system to be solved at each time step is:

$$A\psi^{n+1} = B\psi^n$$

Tridiagonal Matrices D_x and D_y

The matrices D_x and D_y represent the discretization of the second-order spatial derivative in the x and y directions, respectively. They are symmetric and tridiagonal:

$$D = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ 0 & -1 & 2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix}$$

This structure applies to both D_x and D_y, adjusted to the sizes $N_x \times N_x$ and $N_y \times N_y$, respectively.

In Python, this is built using: ‘scipy.sparse.diags([-1, 2, -1], [-1, 0, 1])‘

2D Laplacian via Kronecker Product

The full 2D Laplacian is constructed as:

$$L = I_y \otimes D_x + D_y \otimes I_x$$

This generates a large sparse matrix representing the coupling between all interior grid points in the 2D domain.

- I_x, I_y: identity matrices of sizes N_x and N_y
- $\otimes$: Kronecker product (tensor product)
- L: matrix of size $(N_x N_y) \times (N_x N_y)$, acting on the flattened wavefunction vector

In Python, this is implemented as: ‘Laplacian = kron(Iy, Dx) + kron(Dy, Ix)‘

Evolution Matrices A and B

In the Crank–Nicolson method, time evolution requires solving the system at each time step:

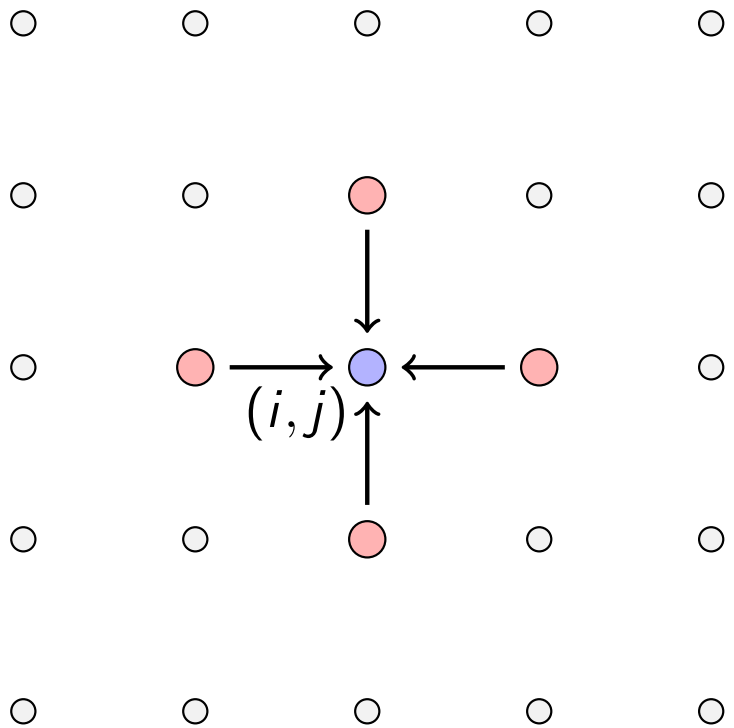
$$A\psi^{n+1} = B\psi^n \quad \text{with} \quad A = I + sL, \quad B = I - sL$$

Where:

- I: identity matrix of size $N_x N_y \times N_x N_y$
- L: full Laplacian matrix (via Kronecker product)
- $s = \dfrac{i\hbar\Delta t}{4m\Delta x^2}$

Both A and B are sparse matrices. The matrix A is LU-factorized once to speed up the simulation.

5-Point Stencil – Grid and Coupling



Laplacian Operator:

$$\nabla^2 \psi_{i,j} \approx \frac{\psi_{i+1,j} + \psi_{i-1,j} + \psi_{i,j+1} + \psi_{i,j-1} - 4\psi_{i,j}}{\Delta x^2}$$

Each point (i, j) is coupled with its immediate neighbors using the 5-point stencil scheme to approximate the Laplacian.

General Algorithm

1. Define the grid: $\Delta x, \Delta y, \Delta t, N$
2. Set initial conditions and box dimensions: $x_0, y_0, k_{0x}, k_{0y}, L$.
   - Gaussian wave packet centered at (x_0, y_0)
   - Packet with width σ and initial phase $e^{i(k_{0x}x + k_{0y}y)}$
3. Construct matrix D, then formulate the 2D Laplacian L using Kronecker products:

$$L = I_y \otimes D_x + D_y \otimes I_x$$

General Algorithm

4. Compute implicit evolution matrices:

$$A = I + sL, \quad B = I - sL \quad \text{with} \quad s = \frac{i\hbar\Delta t}{4m\Delta x^2}$$

and factorize A via LU decomposition for efficient solving; this is done only once in the code, since L, A, B remain constant.

5. Flatten the wavefunction $\psi(x, y)$ into a column vector (initially declared as a matrix on the grid).

General Algorithm

6. For each time step:
   - Compute right-hand side: $b = B\psi^n$
   - Solve the linear system: $A\psi^{n+1} = b$
   - Reshape ψ^{n+1} back to 2D matrix form and apply boundary conditions
   - Flatten ψ again into column vector for the next time step
   - Every 10 time steps:
     - Compute $|\psi(x, y, t)|^2$ (probability density)
     - Store the frame for animation
7. Visualize through animation:
   - A video is generated showing the evolution of $|\psi(x, y, t)|^2$
   - Uses an `inferno` color scale with colorbar

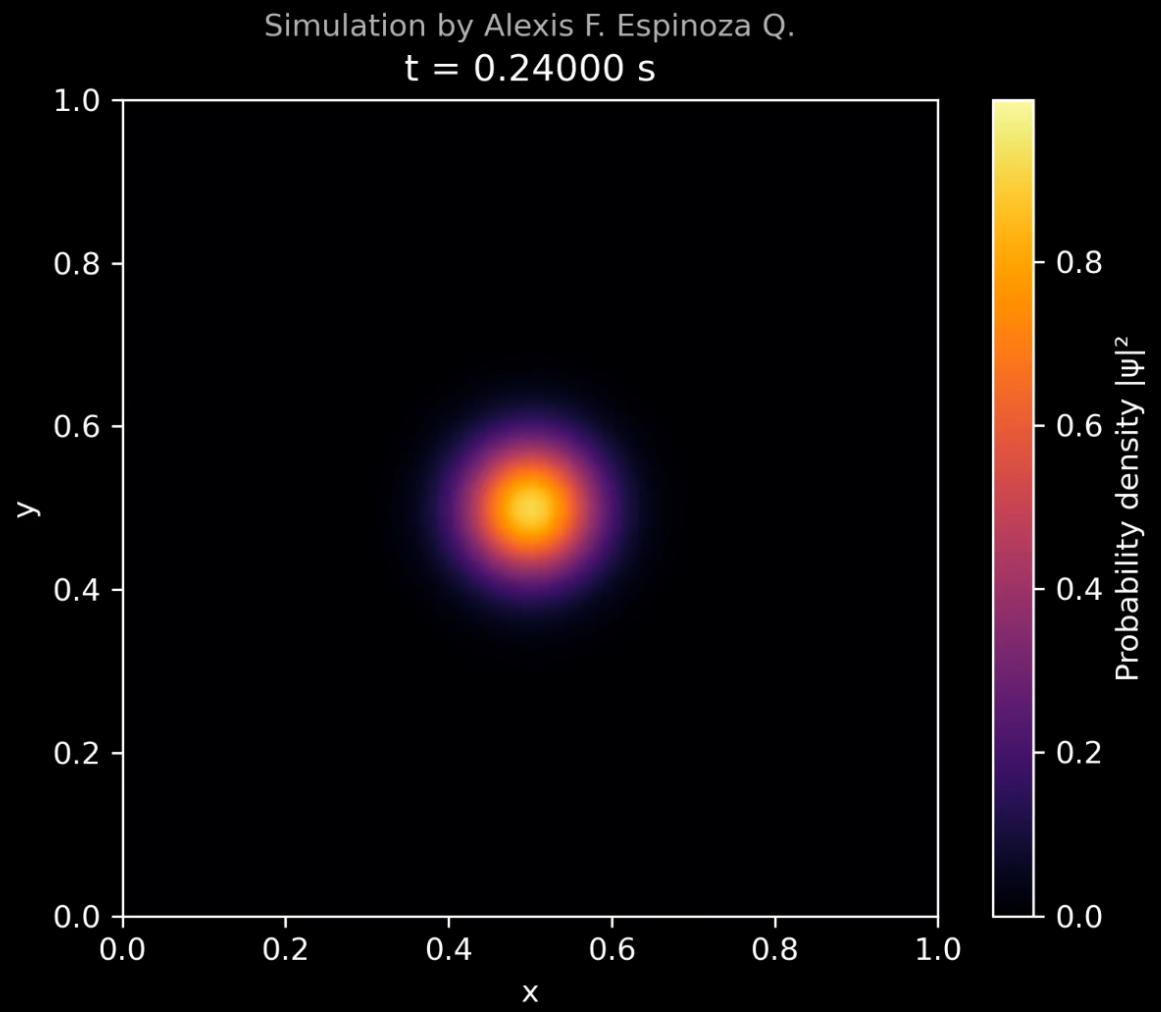


Figure: Evolution of the 2D quantum wave packet. Propagation with initial momentum $p_0 = \hbar k_0$. Unitary simulation using Crank–Nicolson.

Simulations with Diverse Initial Parameters

Case	x_0	y_0	k_0	θ	σ
1	$0.3L$	$0.3L$	12.0	0°	0.05
2	$0.2L$	$0.2L$	15.0	45°	0.15
3	$0.5L$	$0.5L$	0.0	—	0.08
4	$0.5L$	$0.5L$	0.0	—	0.4
5	$0.25L$	$0.25L$	5.0	90°	0.13
6	$0.5L$	$0.5L$	3.0	135°	0.2
7	$0.9L$	$0.1L$	8.0	180°	0.4
8	$0.05L$	$0.95L$	25.0	270°	0.07
9	$0.2L$	$0.8L$	10.0	30°	0.1
10	$0.45L$	$0.55L$	7.5	120°	0.06

→ Videos ...

Physical Analysis of Cases 1–5

- **Case 1:** Horizontal propagation with $k_0 = 12$, very narrow wave packet $\sigma = 0.05$, strong collision with the right wall, clear reflection.
- **Case 2:** Diagonal motion from the lower-left corner. With $\theta = 45^\circ$ and $k_0 = 15$, produces multiple reflections and complex interference patterns.
- **Case 3:** No initial momentum, localized wave with $\sigma = 0.15$. Centered, symmetric packet. Clearly defined and periodic oscillatory modes; ideal case to analyze norm conservation.
- **Case 4:** Similar to Case 3 ($k_0 = 0$), but with $\sigma = 0.4$: significantly wider packet. Approximates an extended state across the region.
- **Case 5:** Pure vertical motion, $k_0 = 5$, reflection at the upper wall. Packet with $\sigma = 0.13$. Clean pattern.

Physical Analysis of Cases 6–10

- **Case 6:** Diagonal trajectory toward the upper-left corner with $k_0 = 3$. Wide packet $\sigma = 0.2$, smooth motion, more diffuse patterns.
- **Case 7:** From the right edge toward the left with $k_0 = 8$, fast collision. Visible reflection and crossed interference patterns.
- **Case 8:** Top-to-bottom trajectory with $k_0 = 25$, highly collimated ($\sigma = 0.07$). Violent rebound and long straight-line path.
- **Case 9:** Smooth trajectory $\theta = 30^\circ$, medium-energy collision. Internal interference is visible at each bounce.
- **Case 10:** From the center with $\theta = 120^\circ$. Packet width $\sigma = 0.06$. Wave modes form that periodically expand and refocus the wave packet at the center of the region.

Problem Parameters and Grid

```
# Square domain of size L
L = 0.6

# Spatial resolution: Nx x Ny interior points
Nx = Ny = 320
dx = L / (Nx + 1)      # uniform spatial step

# Spatial coordinates
x = np.linspace(0, L, Nx + 2)
y = np.linspace(0, L, Ny + 2)
X, Y = np.meshgrid(x, y)  # 2D spatial mesh/grid
```

Temporal Discretization

```
# Total simulation time
T = 0.8

# Small time step for stability
dt = 1e-4
Nt = int(T / dt)

# Dimensionless factor: s = i hbar  t  / (2m  x **2)
hbar = 1.0
m = 1.0
s = 1j * hbar * dt / (4 * m * dx**2)
```

Initial Condition: Gaussian Wave Packet - 1

```
# Parameters of the Gaussian packet
x0, y0 = 0.3, 0.3       # initial position
sigma = 0.1             # packet width
k0 = 10.0               # initial momentum
theta = 0               # direction (in radians)

# Components of the wave vector
k0x = k0 * np.cos(theta)
k0y = k0 * np.sin(theta)
```

Initial Condition: Gaussian Wave Packet - 2

```
# Initial phase according to momentum
fase = np.exp(1j * (k0x * X + k0y * Y))
# Definition of the initial wave function
psi0 = np.exp(-((X - x0)**2 + (Y - y0)**2) / (2 *
    sigma**2)) * fase
# Boundary conditions: zero values at the edges
psi0[0, :] = psi0[-1, :] = 0.0
psi0[:, 0] = psi0[:, -1] = 0.0
# Extract the interior part and flatten into a column
    vector
psi = psi0[1:-1, 1:-1].flatten()
```

Construction of the 2D Laplacian

```
# Construction of tridiagonal matrices
Ix = identity(Nx)
Iy = identity(Ny)
Dx = diags([-1, 2, -1], [-1, 0, 1], shape=(Nx, Nx))
Dy = diags([-1, 2, -1], [-1, 0, 1], shape=(Ny, Ny))

# 2D Laplacian = Iy  Dx + Dy  Ix
Laplacian = kron(Iy, Dx) + kron(Dy, Ix)
```

Crank-Nicolson Matrices and LU Factorization

```
# Evolution matrices: A psi n+1 = B psi n
A = identity(Nx * Ny) + s * Laplacian
B = identity(Nx * Ny) - s * Laplacian

# LU factorization of A (constant over time)
LU = splu(A.tocsc())
```

Temporal Evolution of the System

```
# List to store the evolution
data = []

# Time iteration
for n in range(Nt):
    b = B.dot(psi)              # right hand side: B psi n
    psi = LU.solve(b)           # solution: psi n+1 = A**-1
        b

    if n % 10 == 0:             # save every 10 steps
        psi_grid = np.zeros((Ny + 2, Nx + 2), dtype=
            complex)
        psi_grid[1:-1, 1:-1] = psi.reshape((Ny, Nx))
        data.append(np.abs(psi_grid)**2)
```

Visualization and Animation - 1

```
# Graphic configuration
fig, ax = plt.subplots()
im = ax.imshow(data[0], extent=[0, L, 0, L], origin='
    lower',
                   cmap='inferno', vmin=0, vmax=np.max(
                       data[0]))

# Axes and title
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_title('|  (x, y, t)|   evolution')
```

Visualization and Animation - 2

```
# Colorbar
cbar = fig.colorbar(im, ax=ax)
cbar.set_label('Probability density |  |  ')

# Update function for the animation
def update(frame):
    im.set_data(data[frame])
    ax.set_title(f't = {frame * dt * 10:.5f} s')
    return [im]

# Create and save animation
ani = animation.FuncAnimation(fig, update, frames=len(
    data), blit=True)
ani.save("schrodinger2DBox.mp4", writer='ffmpeg', fps
    =30, dpi=300)
```

Simulation and Animation

Animation produced with matplotlib.animation.FuncAnimation, saved as: `schrodinger2DBox.mp4`
Includes:

- Colormap of type `inferno`
- Scale for probability density
- Dynamic time label (time counter)

Normal Modes and Eigenfunctions in the Quantum Box

- A Gaussian wave packet inside a quantum box is a superposition of multiple eigenstates of the system. If the packet has no initial momentum ($k_0 = 0$), primarily low-energy modes centered symmetrically are excited.
- Each of these modes is an **eigenfunction of the Hamiltonian** with infinite well boundary conditions:

$$\phi_{n,m}(x,y) = \sin\left(\frac{n\pi x}{L}\right)\sin\left(\frac{m\pi y}{L}\right), \quad n,m \in \mathbb{N}$$

- These functions satisfy:

$$\hat{H}\phi_{n,m} = E_{n,m}\phi_{n,m}, \quad \text{with} \quad E_{n,m} = \frac{\hbar^2\pi^2}{2mL^2}(n^2+m^2)$$

Normal Modes and Eigenfunctions in the Quantum Box

- During temporal evolution, each mode oscillates independently as:

$$\psi_{n,m}(x, y, t) = \phi_{n,m}(x, y)e^{-iE_{n,m}t/\hbar}$$

- The interference among multiple modes generates **quasiperiodic oscillatory patterns** in the probability density. In the simulations, these patterns appear as "beats" or modulations, especially in cases c2 and c3.

What happens if $k_0 \neq 0$?

- If the initial Gaussian wave packet has nonzero momentum, it is still a superposition of the system's eigenfunctions:

$$\psi(x, y, 0) = \sum_{n,m} c_{n,m}\, \phi_{n,m}(x, y)$$

- Each coefficient $c_{n,m}$ depends on the characteristics of the packet (center, width, and wave vector $\vec{k}_0$), but the eigenfunctions $\phi_{n,m}$ remain the same.
- During the evolution, the wave function is:

$$\psi(x, y, t) = \sum_{n,m} c_{n,m}\, \phi_{n,m}(x, y)\, e^{-iE_{n,m}t/\hbar}$$

What happens if $k_0 \neq 0$?

- Although the packet moves, bounces, and changes shape due to interference, after an initial transient a quasiperiodic dynamics with regular interferences or "beats" may also emerge; this is observed in cases c8, c9, and c10.
- The probability density does not "collapse" into individual eigenstates, but it can stabilize into a recurrent structure.

Conclusion:
Even with $\vec{k}_0 \neq \vec{0}$, quasiperiodic beating patterns can be observed, although with different shapes and frequencies compared to the case with $\vec{k}_0 = 0$.

Energy Conservation in the Quantum System

- The quantum evolution is governed by the Schrödinger equation:

$$i\hbar\frac{\partial}{\partial t}\psi(x,y,t) = \hat{H}\psi(x,y,t)$$

- If the Hamiltonian $\hat{H}$ is time-independent (as in this system):

$$\frac{d}{dt}\langle\hat{H}\rangle = 0 \quad \Rightarrow \quad \text{the total energy is conserved.}$$

- The Hamiltonian operator in the simulation is:

$$\hat{H} = -\frac{\hbar^2}{2m}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)$$

- The numerical evolution uses Crank–Nicolson, which is:
  - **Unitary:** conserving the norm $\int |\psi|^2\, dx\, dy$
- Therefore, in all simulation cases (with $k_0 = 0$ or $k_0 \neq 0$), the expected energy is conserved:

$$\langle\psi(t)|\hat{H}|\psi(t)\rangle = \text{constant}$$

Conclusions

- The Crank-Nicolson method is numerically stable and unitary, fulfilling the necessary conditions to maintain the norm of the wave function ψ at each iteration. Its temporal and spatial discretization allows simulating time evolution, with the possibility of extension to external potentials, barriers, or more complex potential well geometries.
- In the case of zero initial momentum, cases c2 and c3, the wave packet does not propagate in any preferred direction. However, it does not remain static: spatial pattern oscillations are observed, resulting from interference among multiple stationary modes of the system. These oscillations correspond to quantum normal modes. The total energy is conserved, but its spatial distribution varies periodically in time.

Thank you for your attention

Any questions?