Simulación numérica ecuación de Schrödinger dependiente del tiempo : Método de Crank-Nicolson

Alexis F. Espinoza Q.

Universidad de Talca

20 de julio de 2025

Profesor: Américo Cuchillo Flores

Contenido

1. Introducción
2. Formulación del Problema
3. Discretización
4. Formulación Matricial
5. Algoritmo - Paso a paso
6. Algoritmo - Código
7. Conclusiones

Ecuación de Schrödinger

La ecuación de Schrödinger dependiente del tiempo para una partícula libre ($V = 0$) es:

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

Se propone encerrar un paquete de onda gaussiano como condición inicial, dentro de un espacio bidimensional (caja cuántica 2D), para resolver numéricamente la ecuación de Schrödinger con el método de Crank-Nicolson.

Ecuación de Schrödinger en 2D

Desarrollando el operador laplaciano

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

Se trabaja con las condiciones de borde (paredes potencial infinito):

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

Condición inicial (paquete gaussiano con momento $\vec{p} = \hbar\vec{k}$):

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

Discretización Espacial y Temporal

Se define una malla rectangular de $(Nx + 2) \times (Ny + 2)$ puntos, incluyendo bordes.
Pasos espaciales:

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

con

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

Laplaciano 2D con diferencias finitas centradas:

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

Discretización Temporal - Crank-Nicolson

Discretización temporal con paso Δt. Uso de diferenciales: $dt \approx \Delta t$, $dx \approx \Delta x$, $dy \approx \Delta y$.
Crank-Nicolson (implícito, unitario):

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

Se multiplica por Δt, y se reordena como:

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

Laplaciano 2D con Producto de Kronecker

Se define:

$$D_x = \text{matriz tridiagonal de diferencias centradas en } x$$
$$D_y = \text{matriz tridiagonal de diferencias centradas en } y$$
$$I_x = \text{matriz identidad } N_x \times N_x$$
$$I_y = \text{matriz identidad } N_y \times N_y$$

Laplaciano completo:

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

Matrices de evolución del método son:

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

La ecuación lineal a resolver en cada paso de tiempo queda como:

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

Matrices Tridiagonales D_x y D_y

Las matrices D_x y D_y representan la discretización de la segunda derivada espacial en las direcciones x e y, respectivamente. Son tridiagonales y simétricas:

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

Esta forma se repite tanto para D_x como para D_y, adaptándose al tamaño $N_x \times N_x$ o $N_y \times N_y$.

En Python, se construyó con ‘scipy.sparse.diags([-1, 2, -1], [-1, 0, 1])‘.

Laplaciano 2D con Producto de Kronecker

El Laplaciano completo en 2D se construye como:

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

Esto genera una gran matriz dispersa (sparse), que representa el acoplamiento entre todos los puntos interiores de la malla 2D.

- I_x, I_y: matrices identidad de tamaños N_x, N_y.
- $\otimes$: producto de Kronecker (producto tensorial).
- L: matriz de tamaño $(N_x N_y) \times (N_x N_y)$, aplicada sobre el vector aplanado de la función de onda.

En Python se logra con: ‘Laplacian = kron(Iy, Dx) + kron(Dy, Ix)‘

Matrices de Evolución A y B

En el método de Crank-Nicolson, la evolución en el tiempo requiere resolver en cada paso:

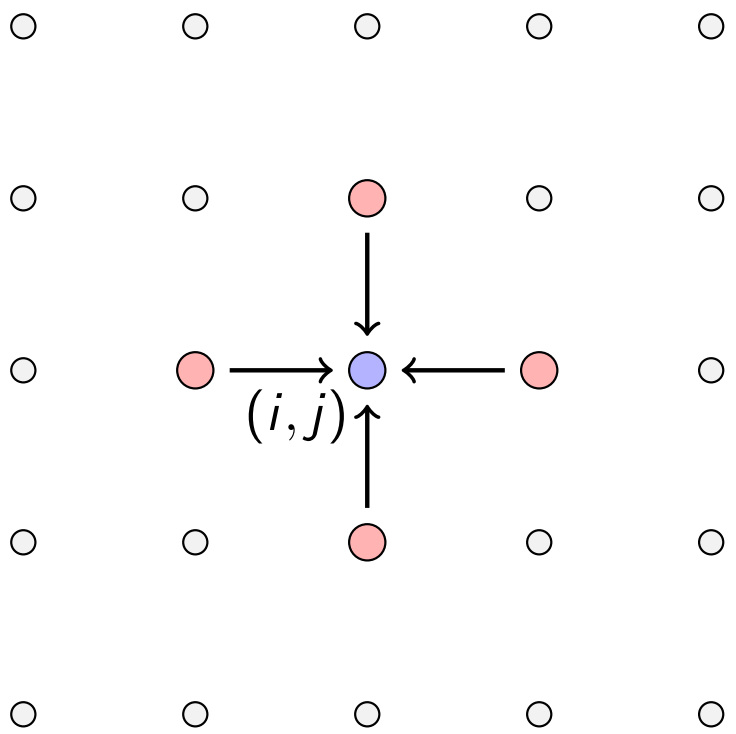
$$A\psi^{n+1} = B\psi^n \quad \text{con} \quad A = I + sL, \quad B = I - sL$$

Donde:

- I: matriz identidad de tamaño $N_x N_y \times N_x N_y$
- L: Laplaciano completo (producto de Kronecker)
- $s = \dfrac{i\hbar\Delta t}{4m\Delta x^2}$

Las matrices A y B también son dispersas, pero A se factoriza una sola vez (LU) para acelerar la simulación.

Stencil de 5 puntos - Mallado y Acoplamiento



Operador Laplaciano:

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

Cada punto (i, j) se acopla con sus vecinos inmediatos usando el esquema de 5 puntos para aproximar el Laplaciano.

Algoritmo General

1. Definir malla: $\Delta x, \Delta y, \Delta t, N$
2. Establecer condiciones iniciales y dimensión de la caja: $x_0, y_0, k_{0x}, k_{0y}, L$.
   - Paquete gaussiano centrado en (x_0, y_0).
   - Paquete de ancho σ y fase inicial $e^{i(k_{0x}x + k_{0y}y)}$
3. Construir matriz D, entonces, se formula el Laplaciano 2D L mediante productos de Kronecker

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

Algoritmo General

4. Calcular matrices de evolución implícitas:

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

y descomponer A mediante factorización LU para resolver eficientemente; esto sucede solo una vez en el código, dado que L, A, B son constantes.

5. Aplanar la función de onda $\psi(x, y)$ a un vector columna (Al inicio es declarada como matriz en el mallado).

Algoritmo General

4. Para cada paso de tiempo:
   - Calcular el término del lado derecho: $b = B\psi^n$
   - Resolver el sistema lineal: $A\psi^{n+1} = b$
   - Convertir ψ^{n+1} a forma matricial 2D y aplicarle condiciones de borde
   - Volver a convertir ψ a vector columna para el siguiente paso
   - Cada 10 pasos de tiempo:
     - Calcular $|\psi(x, y, t)|^2$ (densidad de probabilidad)
     - Almacenar el frame para animación
5. Visualizar mediante animación:
   - Se genera un video de la evolución de $|\psi(x, y, t)|^2$
   - Se utiliza escala de colores tipo `inferno` y barra de color

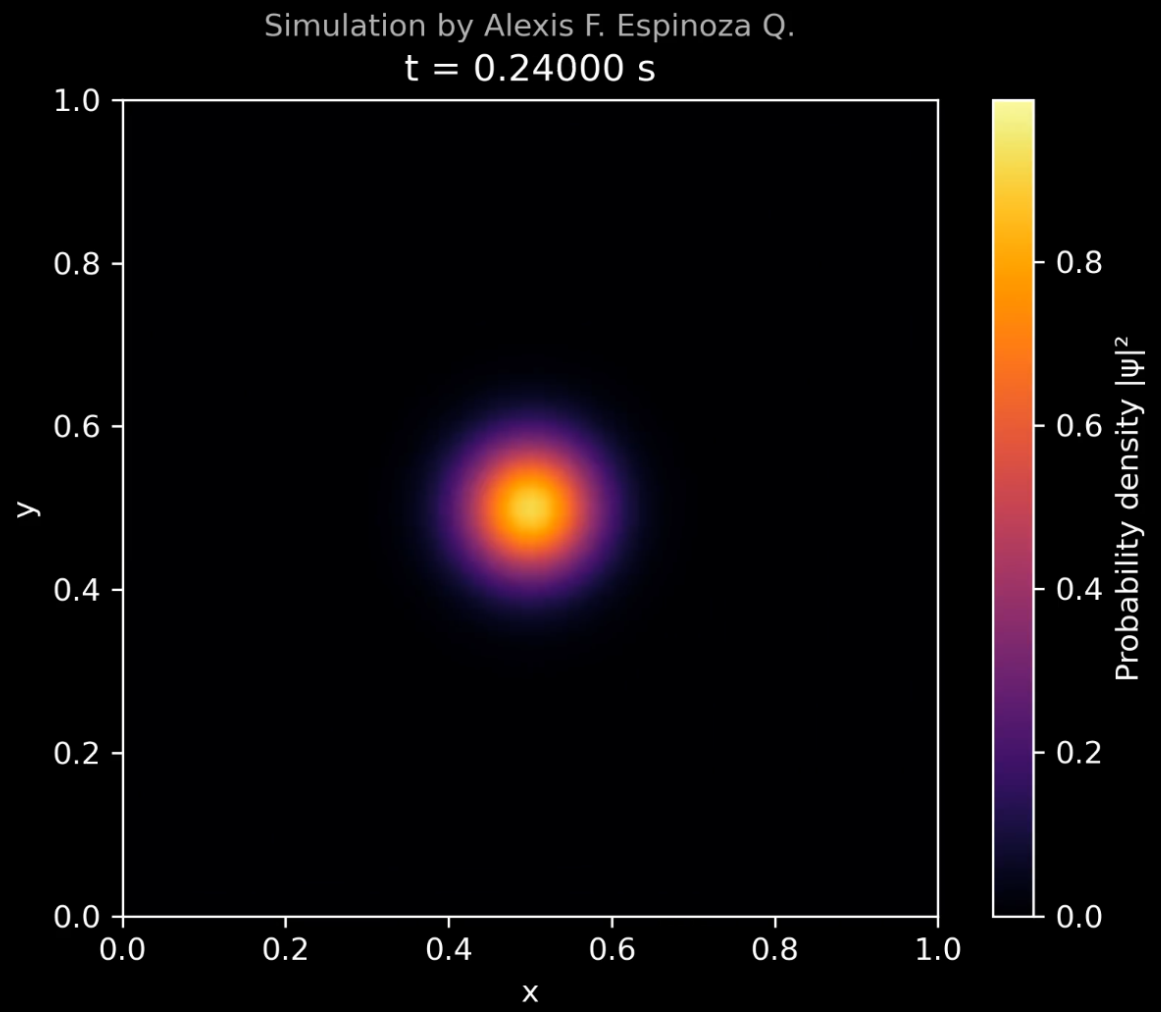


Figura: Evolución del paquete de onda cuántico en 2D. Propagación con momento inicial $p_0 = \hbar k_0$. Simulación conservativa con Crank-Nicolson.

Simulaciones con Parámetros Iniciales Diversos

Caso	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

$\rightarrow$ **Videos ...**

Análisis Físico de Casos 1–5

- **Caso 1:** Propagación horizontal con $k_0 = 12$, paquete muy estrecho $\sigma = 0.05$, colisión fuerte con pared derecha, reflexión clara.
- **Caso 2:** Movimiento diagonal desde esquina inferior izquierda. Con $\theta = 45^\circ$ y $k_0 = 15$, genera múltiples rebotes y patrones de interferencia complejos.
- **Caso 3:** Sin momento inicial, onda localizada con $\sigma = 0.15$. Paquete centrado, simétrico. Onda con diversos modos de oscilación bien marcados y periódicos; caso favorable para estudiar la conservación de la norma.
- **Caso 4:** Similar a caso 3 ($k_0 = 0$), pero con $\sigma = 0.4$: paquete mucho más ancho. Aproximación a un estado extendido en la región.
- **Caso 5:** Movimiento vertical puro, $k_0 = 5$, rebote en la pared superior. Paquete con $\sigma = 0.13$. Patrón limpio.

Análisis Físico de Casos 6–10

- **Caso 6:** Diagonal hacia esquina superior izquierda con $k_0 = 3$. Paquete ancho $\sigma = 0.2$, movimiento suave, patrones más difusos.
- **Caso 7:** Desde borde derecho hacia la izquierda con $k_0 = 8$, colisión rápida. Reflexión visible y patrones cruzados.
- **Caso 8:** Desde arriba hacia abajo con $k_0 = 25$, muy colimado ($\sigma = 0.07$). Rebote violento y larga trayectoria recta.
- **Caso 9:** Trayectoria suave $\theta = 30°$, colisión de energía media. Interferencias internas visibles en cada rebote.
- **Caso 10:** Desde el centro con $\theta = 120°$. Con anchura de $\sigma = 0.06$. Se forman modos que tienden a expandir el paquete de onda y luego localizarlo nuevamente en la mitad de la región de forma periódica.

Parámetros del problema y malla

```
# Dominio cuadrado de tamano L
L = 0.6

# Resolucion espacial: Nx x Ny puntos interiores
Nx = Ny = 320
dx = L / (Nx + 1)      # paso espacial uniforme

# Coordenadas espaciales
x = np.linspace(0, L, Nx + 2)
y = np.linspace(0, L, Ny + 2)
X, Y = np.meshgrid(x, y)  # malla/grilla espacial 2D
```

Discretización temporal

```
# Tiempo total de simulacion
T = 0.8

# Paso temporal pequeno para estabilidad
dt = 1e-4
Nt = int(T / dt)

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

Condición inicial: paquete gaussiano - 1

```
# Parametros del paquete gaussiano
x0, y0 = 0.3, 0.3        # posicion inicial
sigma = 0.1              # ancho del paquete
k0 = 10.0                # momento inicial
theta = 0                # direccion (en radianes)

# Componentes del vector de onda
k0x = k0 * np.cos(theta)
k0y = k0 * np.sin(theta)
```

Condición inicial: paquete gaussiano - 2

```
# Fase inicial segun momento
fase = np.exp(1j * (k0x * X + k0y * Y))
# Definicion de la funcion de onda inicial
psi0 = np.exp(-((X - x0)**2 + (Y - y0)**2) / (2 *
    sigma**2)) * fase
# Condiciones de frontera: valores nulos en los bordes
psi0[0, :] = psi0[-1, :] = 0.0
psi0[:, 0] = psi0[:, -1] = 0.0
# Se extrae parte interior y se aplana en vector
    columna
psi = psi0[1:-1, 1:-1].flatten()
```

Construcción del Laplaciano 2D

```
# Construccion de matrices tridiagonales
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))

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

Matrices Crank-Nicolson y factorización LU

```
# Matrices de evolucion: A psi n+1 = B psi n
A = identity(Nx * Ny) + s * Laplacian
B = identity(Nx * Ny) - s * Laplacian

# Factorizaci n LU de A (constante en el tiempo)
LU = splu(A.tocsc())
```

Evolución temporal del sistema

```
# Lista para guardar la evolucion
data = []

# Iteracion temporal
for n in range(Nt):
    b = B.dot(psi)          # miembro derecho: B psi n
    psi = LU.solve(b)       # resolucion: psi n+1 = A
        **-1 b

    if n % 10 == 0:         # guardar cada 10 pasos
        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)
```

Visualización y animación - 1

```
# Configuracion grafica
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]))

# Ejes y titulo
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_title('|  (x, y, t)|    evolution')
```

Visualización y animación - 2

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

# Funcion de actualizaci n para la animacion
def update(frame):
    im.set_data(data[frame])
    ax.set_title(f't = {frame * dt * 10:.5f} s')
    return [im]

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

Simulación y Animación

Animación producida con ‘matplotlib.animation.FuncAnimation‘, guardada como: `schrodinger2DBox.mp4`
Incluye:

- Colormap tipo `inferno`
- Escala para la densidad de probabilidad
- Etiqueta temporal dinámica (contador de tiempo)

Modos Normales y Eigenfunciones en la Caja Cuántica

- Un paquete de onda gaussiano en una caja cuántica es una superposición de múltiples eigenestados del sistema. Si el paquete no posee momento inicial ($k_0 = 0$), se excitan principalmente modos de baja energía centrados simétricamente.
- Cada uno de estos modos es una **eigenfunción del Hamiltoniano** con condiciones de frontera tipo pozo infinito:

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

- Estas funciones satisfacen:

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

Modos Normales y Eigenfunciones en la Caja Cuántica

- En la evolución temporal, cada modo oscila de forma independiente como:

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

- La interferencia entre múltiples modos genera **patrones oscilatorios cuasiperiódicos** en la densidad de probabilidad. En las simulaciones se observan estos patrones como "latidos" o modulaciones, especialmente en los casos c2 y c3.

¿Qué ocurre si $k_0 \neq 0$?

- Si el paquete gaussiano inicial tiene momento distinto de cero, sigue siendo una superposición de autofunciones del sistema:

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

- Cada coeficiente $c_{n,m}$ depende de características del paquete (centro, anchura, y vector de onda $\vec{k}_0$), pero las autofunciones $\phi_{n,m}$ son las mismas.
- Durante la evolución, la función de onda es:

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

¿Qué ocurre si $k_0 \neq 0$?

- Aunque el paquete se desplaza, rebota y cambia de forma por interferencia, tras un transitorio inicial también puede emerger una dinámica cuasiperiódica con interferencias regulares o "latidos", esto se observa en el c8, c9 y c10.
- La densidad de probabilidad no "colapsa" en eigenestados individuales, pero sí puede estabilizarse en una estructura recurrente.

Conclusión:
Incluso con $\vec{k_0} \neq \vec{0}$, pueden observarse patrones de latido cuasiperiódico, aunque con distinta forma y frecuencia respecto al caso con $\vec{k_0} = 0$.

Conservación de la Energía en el Sistema Cuántico

- La evolución cuántica está gobernada por la ecuación de Schrödinger:

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

- Si el Hamiltoniano $\hat{H}$ es independiente del tiempo (este sistema):

$$\frac{d}{dt}\langle\hat{H}\rangle = 0 \quad \Rightarrow \quad \text{la energía total se conserva.}$$

- El operador Hamiltoniano en la simulación es:

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

- La evolución numérica usa Crank–Nicolson, que es:
  - **Unitario:** conserva la norma $\int |\psi|^2\, dx\, dy$
- Entonces, en todos los casos de simulación (con $k_0 = 0$ o $k_0 \neq 0$), se conserva la energía esperada:

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

Conclusiones

- Método Crank-Nicolson es estable (numéricamente) y unitario, cumpliendo lo necesario para mantener la norma de la función de onda ψ en cada iteración. La discretización temporal y espacial que ofrece el método, permite simular la evolución temporal, con posibilidad de extender a potenciales externos, barreras o geometrías de pozos potenciales más complejos.
- En el caso de momento inicial nulo, c2 y c3, el paquete de onda no se propaga en ninguna dirección preferente. Sin embargo, no permanece estático: se observa una oscilación en patrones espaciales, resultado de la interferencia entre múltiples modos estacionarios del sistema. Estas oscilaciones corresponden a modos normales cuánticos. La energía total se conserva, pero su distribución espacial varía periódicamente con el tiempo.

Gracias por su atención

¿Preguntas?