Evolución temporal de paquete gaussiano

Curso de programación en Mathematica

1. Método numérico

A continuación se muestra un método numérico para calcular la evolución temporal de un paquete gaussiano. El código en Mathematica es una reimplementación del que se muestra en `http://jakevdp.github.io/blog/2012/09/05/quantum-python/`.

Para obtener la evolución temporal del paquete gaussiano se utilizará un método numérico. Partiendo de la ecuación de Schrödinger,

$$i\hbar\frac{\partial\psi}{\partial t} = -\frac{\hbar^2}{2m}\frac{\partial^2\psi}{\partial x^2} + V\psi.$$

con la transformada de Fourier

$$\tilde{\psi}(k,t) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\psi(x,t)e^{ikx}\,dx,$$

se llega a la representación de la ecuación de Scrödinger en el espacio de Fourier

$$i\hbar\frac{\partial\tilde{\psi}}{\partial t} = \frac{\hbar^2k^2}{2m}\tilde{\psi} + V\left(i\frac{\partial}{\partial k}\right)\tilde{\psi}.$$

La aproximación consiste en resolver ambas ecuaciones para intervalos temporales pequeños sin considerar los términos que depende de la derivada de x y k. Entonces queda

$$i\hbar\frac{\partial\psi}{\partial t} = V(x)\psi$$

cuya solución es

$$\psi(x,t+\Delta t) = \psi(x,t)e^{-\frac{iV(x)\Delta t}{\hbar}},$$

y de la ecuación de Scrödinger en el espacio de Fourier

$$i\hbar\frac{\partial\tilde{\psi}}{\partial t} = \frac{\hbar^2k^2}{2m}\tilde{\psi},$$

cuya solución es

$$\tilde{\psi}(k,t+\Delta t) = \tilde{\psi}(k,t)e^{-\frac{i\hbar k^2\Delta t}{2m}}.$$

Para resolver el problema numéricamente conviene discretizarlo, de manera que se usará una transformada de Fourier discreta,

$$\widetilde{F_m} = \sum_{n=0}^{N-1}F_ne^{-2\pi inm/N},$$

y su inversa

$$F_n = \frac{1}{N}\sum_{m=0}^{N-1}\widetilde{F_m}e^{2\pi inm/N}.$$

Para llegar a esta forma discreta primero se hará la consideración de que el potencial tiende a infinito para los valores $x \leq a$ y $x \geq b$. Entonces

$$\widetilde{\psi}(k,t) = \frac{1}{\sqrt{2\pi}}\int_a^b\psi(x,t)e^{-ikx}dx.$$

Al aproximar la integral por una suma de Riemman de N términos, con $\Delta x = (b-a)/N$ y $x_n = a + n\Delta x$ queda

$$\widetilde{\psi}(k,t) \simeq \frac{1}{\sqrt{2\pi}} \sum_{n=0}^{N-1} \psi(x_n,t) e^{-ikx_n} \Delta x.$$

Por último se hace $k_m = k_0 + m\Delta k$, con $\Delta k = 2\pi/(N\Delta x)$, y queda

$$\widetilde{\psi}(k_m,t) \simeq \frac{1}{\sqrt{2\pi}} \sum_{n=0}^{N-1} \psi(x_n,t) e^{-ik_m x_n} \Delta x.$$

Sustituyendo esto en la expresión para la transformada de Fourier discreta se llega a

$$\left[\widetilde{\psi}(k_m,t) e^{imx_0\Delta k}\right] \simeq \sum_{n=0}^{N-1} \left[\frac{\Delta x}{\sqrt{2\pi}} \psi(x_n,t) e^{-ik_0 x_n}\right] e^{-2\pi imn/N}$$

y para la transformada de Fourier inversa

$$\left[\frac{\Delta x}{\sqrt{2\pi}} \psi(x_n,t) e^{-ik_0 x_n}\right] \simeq \frac{1}{N} \sum_{m=0}^{N-1} \left[\widetilde{\psi}(k_m,t) e^{-imx_0\Delta k}\right] e^{2\pi imn/N}.$$

Este resultado nos indica que el par

$$\psi(x,t) \Longleftrightarrow \widetilde{\psi}(k,t)$$

en el caso continuo corresponde al par

$$\frac{\Delta x}{\sqrt{2\pi}} \psi(x_n,t) e^{-ik_0 x_n} \Longleftrightarrow \widetilde{\psi}(k_m,t) e^{-imx_0\Delta k}$$

en el caso discreto con las aproximaciones. Con esto se puede construir un algoritmo para evaluar la evolución temporal de la función de onda.

1. Se escogen a, b, N y k_0 suficientes para representar el estado inicial de la función de onda $\psi(x)$. Con esto se puede calcular $\Delta x = (b-a)/N$ y $\Delta k = 2\pi/(b-a)$. Se definen también los arrays $x_n = a + n\Delta x$ y $k_m = k_0 + m\Delta k$.
2. Se discretiza la función de onda en el array. Sea $\psi_n(t) = \psi(x_n,t)$, $V_n = V(x_n)$, y $\widetilde{\psi}_m = \widetilde{\psi}(k_m,t)$.
3. Para avanzar el sistema en intervalo temporal Δt se hace lo siguiente. Se computa un medio paso en x,
$$\psi_n \longleftarrow \psi_n \exp[-i(\Delta t/2)(V_n/\hbar)]$$
4. Se calcula $\widetilde{\psi}_m$ de ψ_n usando la transformada de Fourier discreta.
5. Se computa un paso completo en k,
$$\widetilde{\psi}_m \longleftarrow \widetilde{\psi}_m \exp[-i\hbar(k\cdot k)\Delta t/(2m)].$$
6. Se calcula ψ_n de $\widetilde{\psi}_m$ usando la transformada de Fourier discreta inversa.
7. Se computa un segundo medio paso en x,
$$\psi_n \longleftarrow \psi_n \exp[-i(\Delta t/2)(V_n/\hbar)]$$
8. Se repite desde el paso 3 hasta que se avance al tiempo deseado.

El código de Mathematica que implementa este algoritmo es el siguiente

```
(* Parametros *)
dt = 0.005;
ℏ = 1;
m = 1.9;
V0 = 0.21;
dx = 0.1;
xmin = -102.4;
xmax = 102.4;
Nn = (xmax - xmin)/dx; (* Tamaño de los arrays *)

(* Array de espacios *)
   arreglo
xarray = Table[ i, {i, -1/2 Nn dx, 1/2 Nn dx, dx}];
         tabla
dk = (2 π)/(Nn dx);
k0 = -1/2 Nn dk;
karray = Table[ k0 + i, {i, 0, Nn dk, dk}];
         tabla

(* Potencial *)
L = ℏ/Sqrt[2 m];
ac = 3 L;
V[x_, a_, V0_] := (2 V0/3) (UnitStep[x + 2 a] - UnitStep[x + a]) +
                            función paso unidad    función paso unidad
   (V0/2) (UnitStep[x + a] - UnitStep[x]) + V0 (UnitStep[x] - UnitStep[x - a]) +
           función paso unidad  función paso unidad  función paso unidad función paso unidad
   (V0/2) (UnitStep[x - a] - UnitStep[x - 2 a]) +
           función paso unidad  función paso unidad
   (V0/8) (UnitStep[x - 2 a] - UnitStep[x - 4 a]);
           función paso unidad    función paso unidad
varray = V[xarray, ac, V0];
zeroindex = 45;
Do[varray[[i]] = 10^20, {i, 1, zeroindex }]
repite
Do[varray[[i]] = 10^20, {i, Length[varray] - zeroindex , Length[varray]}]
                            longitud                        longitud

(* Array de funciones de onda *)
   arreglo
x0c = -60 L;
```

$$p0c = \sqrt{2\,m\,(0.2)};$$

$$dp2 = \frac{p0c^2}{80};$$

$$d = \frac{\hbar}{\sqrt{2\,dp2}};$$

$$k0c = \frac{p0c}{\hbar};$$

$$\psi 0x[x\_,\ a\_,\ x0\_,\ k0\_] := \frac{1}{\sqrt{a\sqrt{\pi}}}\, e^{-\frac{1}{2}\left(\frac{x-x0}{a}\right)^2 + i\,x\,k0};$$

```
ψ0xarray = ψ0x[xarray, d, x0c, k0c];

(* Para graficar *)
```

$$scalegauss = \frac{1}{\sqrt{d\sqrt{\pi}}};$$

```
scalev0 = V0;
scale = 0;
If[scalegauss > scalev0, scale = scalegauss, scale = scalev0];
si

(* Funciones *)
CF[ψxarray_] := Module[{ψtmp},
                módulo
    ψtmp = ψxarray;
    Do[ψtmp[[i]] = 0.0, {i, 1, zeroindex}];
    repite
    Do[ψtmp[[i]] = 0.0, {i, Length[ψtmp] - zeroindex , Length[ψtmp]}];
    repite                 longitud                   longitud
    Return[ψtmp];
    retorna
  ];
```

$$Discretize[\psi xarray\_] := \psi xarray\,\frac{dx}{\sqrt{2\pi}}\, e^{-i\,k0\,xarray};$$

$$Undiscretize[\psi xarray\_] := \psi xarray\,\frac{\sqrt{2\pi}}{dx}\, e^{i\,k0\,xarray};$$

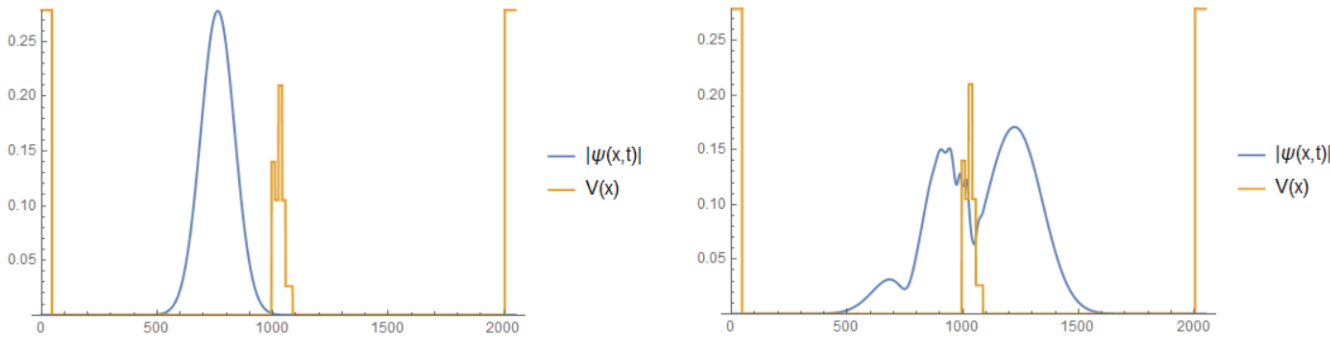
$$Step\psi x\,[\psi xarray\_,\ dt\_] := \psi xarray\, e^{-\frac{i}{2}\frac{varray}{\hbar}dt};$$

$$Step\psi k\,[\psi karray\_,\ dt\_] := \psi karray\, e^{-\frac{i}{2}\frac{\hbar\,karray^2}{m}dt};$$

```
Stepψ[ψxarray_] := Module[{ψmx, ψmk},
                   módulo
  ψmx = CF[ψxarray];
  ψmx = Stepψx[ψmx, dt];
```

```
    ψmk = Fourier[ψmx];
          transformada de Fourier discreta
    ψmk = Stepψk[ψmk, dt];
    ψmx = InverseFourier[ψmk];
          transformada de Fourier discreta inversa
    ψmx = Stepψx[ψmx, dt];
    Return[ψmx];
    retorna
  ]
StepψForList[ψxarray_] := Module[{ψmx, ψmk},
                          módulo
    ψmx = CF[ψxarray];
    ψmx = Stepψx[Discretize[ψmx], dt];
    ψmk = Fourier[ψmx];
          transformada de Fourier discreta
    ψmk = Stepψk[ψmk, dt];
    ψmx = InverseFourier[ψmk];
          transformada de Fourier discreta inversa
    ψmx = Stepψx[ψmx, dt];
    Return[Undiscretize[ψmx]];
    retorna
  ]
Evolve[ψxarray_, iter_] := Module[{},
                           módulo
    Print["t = " <> ToString[iter * dt]];
                    convierte a cadena de caracteres
    Return[Undiscretize[Nest[Stepψ, Discretize[ψxarray], iter]]];
                        anida
  ]
EvolveList[ψxarray_, iter_] := Module[{},
                               módulo
    Print["tmax = " <> ToString[iter * dt]];
                       convierte a cadena de caracteres
    Return[NestList[StepψForList, ψxarray, iter]];
           lista de resultados anidados
  ]
ListLinePlot[{Abs[Evolve[ψ0xarray, 2000]], varray},
gráfico de líne···   valor absoluto
  PlotRange → scale, PlotLegends → {"|ψ(x,t)|", "V(x)"}]
                     leyendas de representación
```

Las gráficas que se generan son



(a) $t = 10$.

(b) $t = 100$.

Figura 1: Gráficas de la evolución temporal.