```
(* 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 *)
xarray = Table[ i, {i, -1/2 Nn dx, 1/2 Nn dx, dx}];

dk = 2 π/(Nn dx);
k0 = -1/2 Nn dk;
karray = Table[ k0 + i, {i, 0, Nn dk, dk}];

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

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

```
p0c = √(2 m (0.2)) ;
dp2 = p0c^2 / 80;
d = ħ / √(2 dp2);
k0c = p0c / ħ;

ψ0x[x_, a_, x0_, k0_] := 1 / √(a √π) e^(-1/2 ((x-x0)/a)^2 + i x k0);
ψ0xarray = ψ0x[xarray, d, x0c, k0c];

(* Para graficar *)
scalegauss = 1 / √(d √π);
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[ψxarray_] := ψxarray dx / √(2 π) e^(-i k0 xarray);
Undiscretize[ψxarray_] := ψxarray √(2 π) / dx e^(i k0 xarray);

Stepψx [ψxarray_, dt_] := ψxarray e^(-i/2 varray/ħ dt);
Stepψk [ψkarray_, dt_] := ψkarray e^(-i/2 ħ 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["t_max = " <> 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
```