Paquet d'onde d'une particule matérielle

1. Introduction

Ce document montre comment calculer des paquets d'onde par combinaison linéaire d'ondes de de Broglie. On utilise un système d'unités où $\hbar = 1$ et $m = 1$.

2. Définition des paquets d'onde

Une onde de de Broglie pour une particule d'impulsion p et d'énergie $E = p^2/2$ s'écrit :

$$\psi(x,t) = Ae^{i(px-Et)} \tag{1}$$

On considère la superposition suivante :

$$\psi(x,t) = \sum_j A(p_j)e^{i(p_j x-E_j t)} \tag{2}$$

Le spectre en impulsion de cette onde comporte des raies d'impulsion p_j et d'énergie $E_j = p_j^2/2$.

Pour définir un paquet d'onde gaussien, on doit définir un spectre gaussien centré sur une impulsion p_0. Pour la raie d'impulsion $p = p_0 + n$, l'amplitude est :

$$A(p) = e^{-\frac{(p-p_0)^2}{4(\Delta p)^2}} \tag{3}$$

La densité de probabilité en impulsion est alors :

$$|A(p)|^2 = e^{-\frac{(p-p_0)^2}{2(\Delta p)^2}} \tag{4}$$

Δp est l'écart-type en impulsion. Il est lié à l'écart-type en position Δx par l'inégalité d'Heisenberg :

$$\Delta x \Delta p \geq \frac{1}{2} \tag{5}$$

Pour le paquet d'onde gaussien, l'égalité est réalisée à $t = 0$. Pour $t > 0$, la largeur Δx du paquet augmente.

Le nombre de raies de part et d'autre de la raie centrale doit être choisi pour que la dernière raie soit au bord de la gaussienne. Le choix suivant convient :

$$n_{raies} = 4\Delta p \tag{6}$$

L'onde résultante comporte une succession périodique de paquets d'ondes. Pour suivre un paquet d'onde en particulier, celui qui se trouve au voisinage de $x = 0$ à l'instant $t = 0$, on doit utiliser la vitesse de groupe :

$$V_g = \frac{dE}{dp} = p_0 \tag{7}$$

La fonction d'onde calculée n'est pas normalisée.

3. Fonctions Python

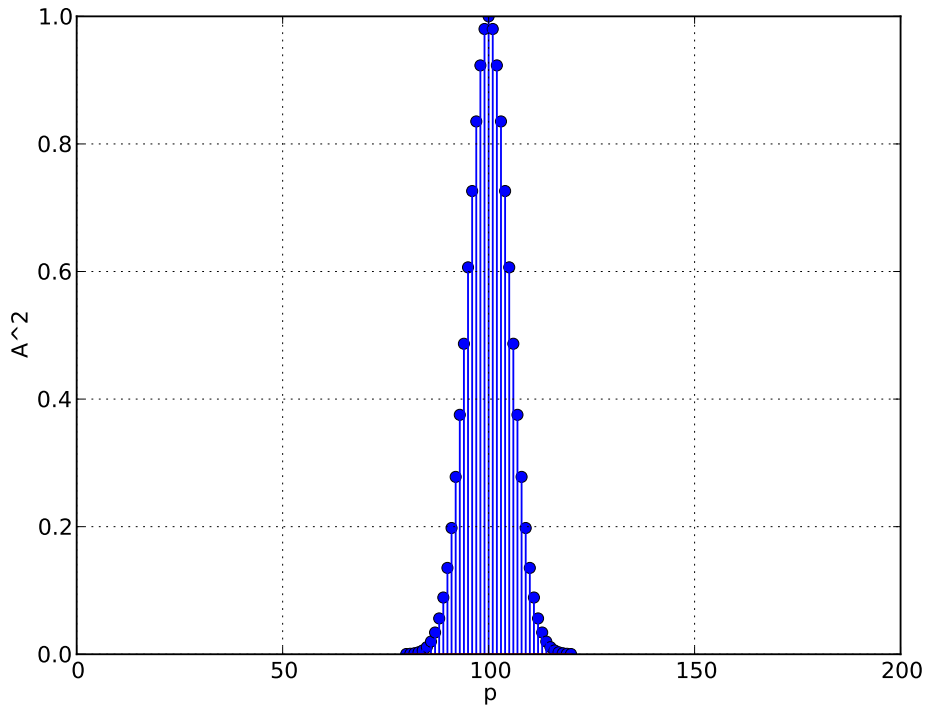
PaquetOndeParticule.py

```
import numpy


class PaquetOndeParticule:
    def __init__(self,suivi):
        self.suivi = suivi
        self.p = []
        self.E = []
        self.A = []

    def spectre_gaussien(self,p0,delta_p):
        self.p = []
        self.E = []
        self.A = []
        delta_p=delta_p*1.0
        n_raies = int(4*delta_p)
        def amplitude(n):
            return numpy.exp(-n*n/(4*delta_p**2))
        for n in range(-n_raies,n_raies+1):
            p = p0+n
            self.p.append(p)
            self.E.append(p*p/2)
            self.A.append(amplitude(n))
        self.N_raies = n_raies*2+1
        self.v_groupe = p0
        return (numpy.array(self.p),numpy.array(self.A))

    def echantillons(self,xmin,xmax,t,N):
        x = numpy.linspace(xmin,xmax,N)
        psi = numpy.zeros(x.size,dtype=numpy.complex)
        for i in range(self.N_raies):
            p = self.p[i]
            E = self.E[i]
            phase = p*x+(p*self.v_groupe*self.suivi-E)*t
            psi += self.A[i]*numpy.exp(1j*phase)
        proba = numpy.real(numpy.conj(psi)*psi)
        return (x,numpy.real(psi),proba)
```

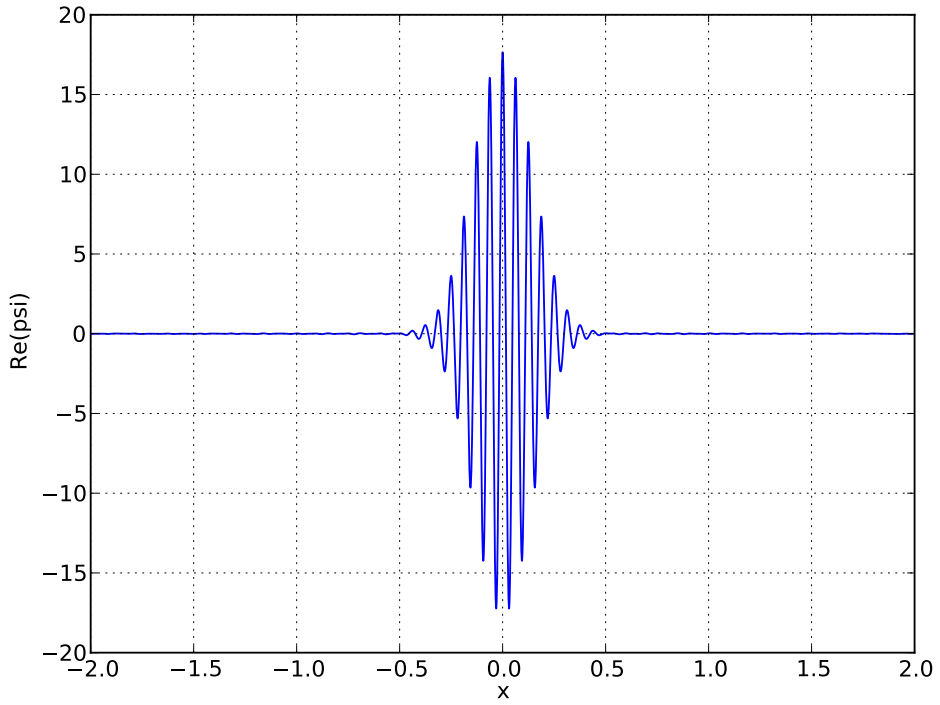
4. Exemple

```
import numpy
from matplotlib.pyplot import *
```

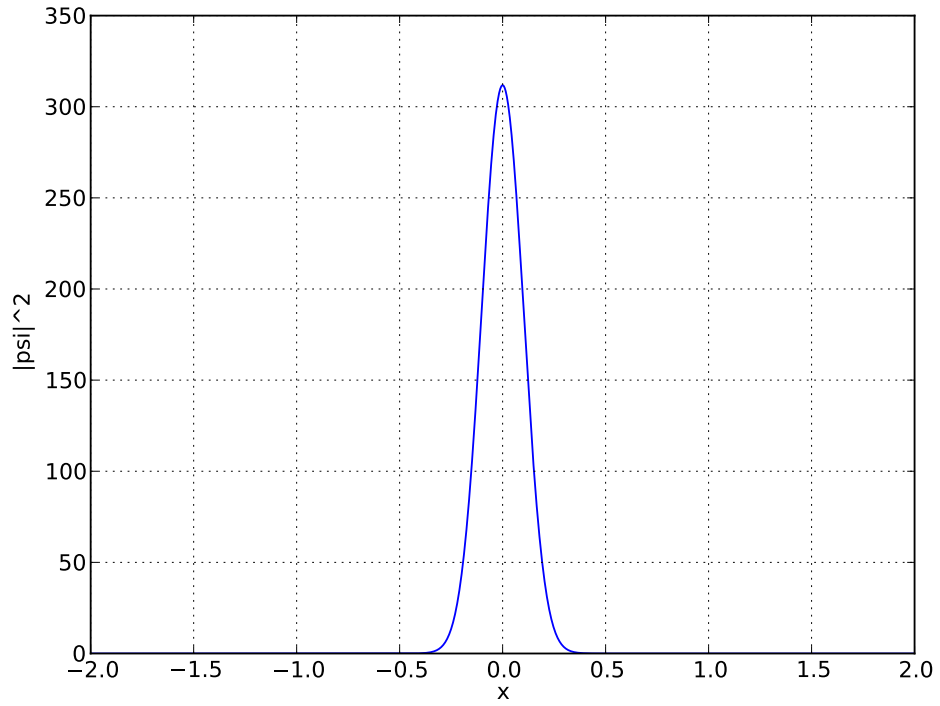
```
from PaquetOndeParticule import *

paquet = PaquetOndeParticule(True)
p0=100
delta_p = 5
(p,A)=paquet.spectre_gaussien(p0,delta_p)
figure()
stem(p,A*A)
xlabel("p")
ylabel("A^2")
axis([0,200,0,1])
grid()
```




```
figure()
xmin=-2
xmax=2
N=5000
(x,re_psi,proba) = paquet.echantillons(xmin,xmax,0.0,N)
plot(x,re_psi)
xlabel("x")
ylabel("Re(psi)")
grid()
```




```
figure()
plot(x,proba)
xlabel("x")
ylabel("|psi|^2")
grid()
```



5. Animation

```
import matplotlib.animation as animation

fig, ax = subplots()
line, = ax.plot(x,proba)
ax.grid()
temps = 0.0
dt = 1.0e-4

def animate(i):
    global temps,xmin,xmax,N
    temps += dt
    (x,re_psi,proba) = paquet.echantillons(xmin,xmax,temps,N)
    line.set_xdata(x)
    line.set_ydata(proba)
    return line,

ani = animation.FuncAnimation(fig,animate,1000,interval=40)

show()
```