mesure_faible_electrique_temporel

June 23, 2023

Calcul analytique de la partie réel et imaginaire de la valeur faible d'un systèm d'optique quantique.

Le système contient un laser pulsé avec un profil gaussien temporel qui est préparé avec une lame demi onde et quart d'onde. L'impulsion subit une mesure faible sur sa partie horizontale de l'état de polarisation presenté par un décalage temporel. Ensuite, l'impulsion est projecter avec l'état de polairsation diagonale.

[ ]:
```python
#sympy pour effectuer les calcules
import sympy as smp
from IPython.display import display, Math
smp.init_session()
```

```
IPython console for SymPy 1.11.1 (Python 3.11.3-64-bit) (ground types: python)

These commands were executed:
>>> from sympy import *
>>> x, y, z, t = symbols('x y z t')
>>> k, m, n = symbols('k m n', integer=True)
>>> f, g, h = symbols('f g h', cls=Function)
>>> init_printing()

Documentation can be found at https://docs.sympy.org/1.11.1/
```

[ ]:
```python
a = smp.symbols('a', real=False, complex = True)
t = smp.symbols('t', real=True)
o = smp.symbols(r'\sigma', real=True, positive = True)
b = smp.symbols('b', real=False, complex = True)
d = smp.symbols(r'\delta', real=True)
z = smp.symbols('z', real=True)
k = smp.symbols('k', real=True, positive=True)
w = smp.symbols(r'\omega', real=True, positive=True)
tau = smp.symbols(r'\tau', real=True)
c = smp.symbols('c', real=True, positive=True, constant=True)
```

[ ]:
```python
#Fonction gaussien
A = (smp.sqrt(1/((smp.sqrt(2*smp.pi))*o)))*smp.exp(-((t-z/c)**2)/(4*o**2))
display(Math('A(t) = '+smp.latex(A)))
```

```python
xi = A*smp.exp(smp.I*(k*z - w*t))
display(Math(r'|\xi(z,t)> = '+smp.latex(xi)))

#intiallement
phi = a + b
H = smp.symbols('|H>')
V = smp.symbols('|V>')
display(Math(r'|\varphi(\theta, \phi)> = ' +smp.latex(a*H + b*V)))
display(Math(r'|E(z,t)> = |\varphi(\theta, \phi)> \otimes |\xi(z,t)>'))
```

$$A(t) = \frac{2^{\frac{3}{4}} e^{- \frac{\left(t - \frac{z}{c}\right)^{2}}{4 \sigma^{2}}}}{2 \sqrt[4]{\pi} \sqrt{\sigma}}$$

$$|\xi(z,t) >= \frac{2^{\frac{3}{4}} e^{i \left(- \omega t + k z\right)} e^{- \frac{\left(t - \frac{z}{c}\right)^{2}}{4 \sigma^{2}}}}{2 \sqrt[4]{\pi} \sqrt{\sigma}}$$

$$|\varphi(\theta, \phi) >= a|H > +b|V >$$

$$|E(z,t) >= |\varphi(\theta, \phi) > \otimes |\xi(z,t) >$$

[ ]:
```python
#decalage (mesure faible) sur H
A_f = (smp.sqrt(1/((smp.sqrt(2*smp.pi))*o)))*smp.exp(-((t+tau-z/c)**2)/(4*o**2))
xi_f = A_f*smp.exp(smp.I*(k*z - w*(t+tau)))
display(Math(r'|\xi_{f}(z,t+\tau)> = ' +smp.latex(xi_f)))
```

$$|\xi_{f}(z,t+\tau) >= \frac{2^{\frac{3}{4}} e^{i \left(- \omega \left(\tau + t\right) + k z\right)} e^{- \frac{\left(\tau + t - \frac{z}{c}\right)^{2}}{4 \sigma^{2}}}}{2 \sqrt[4]{\pi} \sqrt{\sigma}}$$

Ici nous effectuons notre procédure de caractérsation faible et pusique trouver l'expression du opérateur d'intéraction

[ ]:
```python
#champ total initial
E_i = xi*phi
display(Math(r'|E_{i}(z,t)> = '+smp.latex(E_i)))

#PBS
#effectue une mesure faible sur la partie horizontale de la polarisation
E_1 = xi_f*a #celui faible
E_2 = xi*b
display(Math(r'|E_{1}(z,t+\tau)> = '+smp.latex(E_1)))
display(Math(r'|E_{2}(z,t)> = '+smp.latex(E_2)))

#résourd l'opérateur d'interaction
U = smp.symbols(r'\hat{U}')
eq1 = smp.Eq(U*xi*a, xi_f*a)
display(Math(smp.latex(eq1)))
eq2 = smp.solve(eq1, U)
U = eq2[0].simplify()
```

```
display(Math(r'\hat{U} = ' + smp.latex(U)))

#postselection sur D = 1/sqrt(2)*(H_faible + V)
E_w = ((1/smp.sqrt(2))*(E_1 + E_2))
#mesure faible sur H
display(Math(r'|E_{f}(z,t)> = <D|\hat{U}|E_{i}(z,t)> = ' + smp.latex(E_w)))
```

$$|E_i(z,t)>=\frac{2^{\frac{3}{4}}\left(a+b\right)e^{i(-\omega t+kz)}e^{-\frac{\left(t-\frac{z}{c}\right)^2}{4\sigma^2}}}{2\sqrt[4]{\pi}\sqrt{\sigma}}$$

$$|E_1(z,t+\tau)>=\frac{2^{\frac{3}{4}}ae^{i(-\omega(\tau+t)+kz)}e^{-\frac{\left(\tau+t-\frac{z}{c}\right)^2}{4\sigma^2}}}{2\sqrt[4]{\pi}\sqrt{\sigma}}$$

$$|E_2(z,t)>=\frac{2^{\frac{3}{4}}be^{i(-\omega t+kz)}e^{-\frac{\left(t-\frac{z}{c}\right)^2}{4\sigma^2}}}{2\sqrt[4]{\pi}\sqrt{\sigma}}$$

$$\frac{2^{\frac{3}{4}}\hat{U}ae^{i(-\omega t+kz)}e^{-\frac{\left(t-\frac{z}{c}\right)^2}{4\sigma^2}}}{2\sqrt[4]{\pi}\sqrt{\sigma}}=\frac{2^{\frac{3}{4}}ae^{i(-\omega(\tau+t)+kz)}e^{-\frac{\left(\tau+t-\frac{z}{c}\right)^2}{4\sigma^2}}}{2\sqrt[4]{\pi}\sqrt{\sigma}}$$

$$\hat{U}=e^{\frac{-4i\omega\sigma^2\tau c^2+(ct-z)^2-(c(\tau+t)-z)^2}{4\sigma^2c^2}}$$

$$|E_f(z,t)>=<D|\hat{U}|E_i(z,t)>=\frac{\sqrt{2}\cdot\left(\frac{2^{\frac{3}{4}}ae^{i(-\omega(\tau+t)+kz)}e^{-\frac{\left(\tau+t-\frac{z}{c}\right)^2}{4\sigma^2}}}{2\sqrt[4]{\pi}\sqrt{\sigma}}+\frac{2^{\frac{3}{4}}be^{i(-\omega t+kz)}e^{-\frac{\left(t-\frac{z}{c}\right)^2}{4\sigma^2}}}{2\sqrt[4]{\pi}\sqrt{\sigma}}\right)}{2}$$

[ ]:
```
#T = smp.symbols('T', real=True)
#I_i = smp.re(smp.integrate(smp.conjugate(E_i)*E_i, (t, -T, T)))
#I = smp.Abs((1/smp.sqrt(2))*E_i)**2
#I_i = smp.re(smp.integrate(I, (t, -tau/2, tau/2)))
#display(Math('I_{i}(z,t) = ' + smp.latex(I_i)))

I_11 = smp.conjugate(E_1)*E_1
I_22 = smp.conjugate(E_2)*E_2
I_33 = (1/smp.sqrt(2))*(I_11+I_22)
display(Math('I(z,t) = ' + smp.latex((smp.re(I_33)).simplify())))



#I_f = smp.re(smp.integrate(smp.conjugate(E_w)*E_w, (t, -t, t)))
#I_w = (1/smp.sqrt(2))*smp.conjugate(E_i)*E_w
#I_f = smp.re(smp.integrate(I_w, (t, -smp.oo, smp.oo)))
#display(Math('I_{f}(z,t) = ' + smp.latex(I_f.simplify())))

#I_limit = I_f.subs(smp.exp(-(tau**2)/(8*o**2)), 1)
#display(Math(r'\lim_{e^{\frac{-\delta^{2}}{8\sigma^{2}}} \to 1} \langle
↪\hat{I} \rangle= ' +smp.latex(I_limit.simplify())))
```

$$I(z,t) = \frac{e^{\frac{-c^2t^2+2ctz-z^2}{2\sigma^2c^2}}\left|b\right|^2 + e^{\frac{-c^2\left(\tau^2+2\tau t+t^2\right)+2cz(\tau+t)-z^2}{2\sigma^2c^2}}\left|a\right|^2}{2\sqrt{\pi}\sigma}$$

Trouvons la partie réel de la valeur faible

[ ]:
```
t_moy = (smp.integrate(smp.conjugate(E_w)*t*E_w, (t, -smp.oo, smp.oo))).
↪simplify()
display(Math(r'\langle \hat{T} \rangle= <E_{f}(z,t)|\hat{t}|E_{f}(z,t)> =' +smp.
↪latex(t_moy)))

t_moy_limit = t_moy.subs(smp.exp((-tau**2)/(8*o**2)), 1)
display(Math(r'\lim_{e^{\frac{-\delta^{2}}{8\sigma^{2}}} \to 1} \langle \hat{T}␣
↪\rangle= ' +smp.latex(t_moy_limit.simplify())))
```

$$\langle \hat{T} \rangle = < E_f(z,t)|\hat{t}|E_f(z,t) > = -\frac{\tau a e^{-i\omega\tau - \frac{\tau^2}{8\sigma^2}}\overline{b}}{4} - \frac{\tau a\overline{a}}{2} - \frac{\tau b e^{i\omega\tau - \frac{\tau^2}{8\sigma^2}}\overline{a}}{4} + \frac{aze^{-i\omega\tau - \frac{\tau^2}{8\sigma^2}}\overline{b}}{2c} + \frac{az\overline{a}}{2c} + \frac{bze^{i\omega\tau - \frac{\tau^2}{8\sigma^2}}\overline{a}}{2c} + \frac{bz\overline{b}}{2c}$$

$$\lim_{e^{\frac{-\delta^2}{8\sigma^2}} \to 1} \langle \hat{T} \rangle = \frac{\left(-\tau ac\overline{b} - \tau c\left(2a + be^{i\omega\tau}\right)e^{i\omega\tau}\overline{a} + 2az\overline{b} + 2z\left(a\overline{a} + be^{i\omega\tau}\overline{a} + b\overline{b}\right)e^{i\omega\tau}\right)e^{-i\omega\tau}}{4c}$$

Trouvons le G_1

[ ]:
```
G_1_TAU = (smp.integrate(smp.conjugate(E_2)*E_1, (t, -smp.oo, smp.oo))).
↪simplify()
display(Math(r'G(\tau) = ' + smp.latex(G_1_TAU)))
G_1_ZERO = smp.integrate(smp.conjugate(E_2)*E_2, (t, -smp.oo, smp.oo))
display(Math(r'G(0) = ' + smp.latex(G_1_ZERO.simplify())))

g_1 = (G_1_TAU/G_1_ZERO).simplify()
display(Math(r'g^{(1)}(\tau) = ' + smp.latex(g_1)))
```

$$G(\tau) = ae^{-\tau\left(i\omega + \frac{\tau}{8\sigma^2}\right)}\overline{b}$$

$$G(0) = b\overline{b}$$

$$g^{(1)}(\tau) = \frac{ae^{-\tau\left(i\omega + \frac{\tau}{8\sigma^2}\right)}}{b}$$

La temps de cohérence

[ ]:
```
tau_c = smp.integrate((smp.conjugate(g_1)*g_1), (tau, -smp.oo, smp.oo))
display(Math(r'\tau_c = ' + smp.latex(tau_c)))
```

$$\tau_c = \frac{2\sqrt{\pi}\sigma a\overline{a}}{b\overline{b}}$$

Ici nous allons trouvons la partie imaginaire de la valeur faible avec le power spectrum et la fonction de transformation de fourier.

[ ]:
```
f = smp.symbols('f', real=True, constante=True, positive=True)
G_1_TAU=G_1_TAU.subs(w, 2*smp.pi*f)
#using the autocorrelation function
S = (smp.integrate(G_1_TAU*smp.exp(-smp.I*2*smp.pi*f*tau), (tau, -smp.oo, smp.
↪oo))).simplify()
display(Math(r'S(f) = ' + smp.latex(S)))
```

$$S(f) = 2\sqrt{2}\sqrt{\pi}\sigma a e^{-32\pi^2\sigma^2 f^2}\overline{b}$$

[ ]:
```
df = (((smp.integrate(S, (f, 0, smp.oo)))**2)/(smp.integrate(S**2, (f, 0, smp.
↪oo)))).simplify()
display(Math(r'\Delta f = ' + smp.latex(df.simplify())))
```

$$\Delta f = \frac{1}{8\sqrt{\pi}\sigma}$$

[ ]:
```
I_s = smp.re(smp.integrate(S, (f, 0, smp.oo)))
display(Math(r'I_s = ' + smp.latex(I_s.simplify())))
```

$$I_s = \frac{\operatorname{re}\left(a\overline{b}\right)}{4}$$

[ ]:
```
w_moy = (smp.integrate(smp.conjugate(S)*f*S, (f, 0, smp.oo))).simplify()
display(Math(r'<\hat{\Omega}> = <S(\omega)|\hat{\omega}|S(\omega)> =' +smp.
↪latex(w_moy)))
```

$$<\hat{\Omega}>=<S(\omega)|\hat{\omega}|S(\omega)>= \frac{ab\overline{a}\overline{b}}{16\pi}$$

[ ]:
```
F_w = (1/(2*smp.pi)*smp.integrate(g_1*smp.exp(smp.I*w*tau), (tau, -smp.oo, smp.
↪oo))).simplify()
display(Math(r'F(\omega) = ' + smp.latex(F_w)))
```

$$F(\omega) = \frac{\sqrt{2}\sigma a}{\sqrt{\pi} b}$$

[ ]:
```
w_moy = (smp.integrate(smp.conjugate(F_w)*w*F_w, (w, 0, smp.oo))).simplify()
display(Math(r'<\hat{\Omega}> = <F(\omega)|\hat{\omega}|F(\omega)> =' +smp.
↪latex(w_moy)))
```

$$<\hat{\Omega}>=<F(\omega)|\hat{\omega}|F(\omega)>= \begin{cases} \text{NaN} & \text{for}\ \frac{a\overline{a}}{b\overline{b}} = 0 \\ \frac{\infty a\overline{a}}{b\overline{b}\left|\frac{a\overline{a}}{b\overline{b}}\right|} & \text{otherwise} \end{cases}$$

[ ]:
```
```