

MECÁNICA CUÁNTICA I

TAREA #10 (4,3; 4,7)

SOLUCIÓN

PROBLEMA #43

a) $I \cap [Z] ; \omega_0 \equiv \gamma B_0$

b) Podemos usar la fórmula para spin en dirección arbitraria. En este caso tendremos $\theta = \pi/4$, $\varphi = 0$.

$$| \chi_n^+ \rangle = \begin{pmatrix} \cos \pi/8 \\ \sin \pi/8 \end{pmatrix} ; | \chi_n^- \rangle = \begin{pmatrix} \sin \pi/8 \\ -\cos \pi/8 \end{pmatrix}$$

\downarrow v.p. $S_n = +\hbar/2$ \downarrow $S_n = -\hbar/2$.

c) Expandimos $| \rightarrow \rangle$ en la base $\{ | \chi_n^\pm \rangle \}$:

$$| \rightarrow \rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = b \begin{pmatrix} \cos \pi/8 \\ \sin \pi/8 \end{pmatrix} + a \begin{pmatrix} \sin \pi/8 \\ -\cos \pi/8 \end{pmatrix}$$

$$\begin{aligned} b \cos \pi/8 + a \sin \pi/8 &= 0 \\ b \sin \pi/8 - a \cos \pi/8 &= 1 \end{aligned} \Rightarrow \begin{aligned} b &= \sin \pi/8 \\ a &= -\cos \pi/8 \end{aligned}$$

$$\Rightarrow P(\hbar \omega_0/2) = |b|^2 \approx 0,15$$

$$P(-\hbar \omega_0/2) = |a|^2 \approx 0,85$$

$$c) | \psi(t) \rangle = e^{-i\omega_0 t} \sin \pi/8 \begin{pmatrix} \cos \pi/8 \\ \sin \pi/8 \end{pmatrix} - \cos \pi/8 \begin{pmatrix} \sin \pi/8 \\ -\cos \pi/8 \end{pmatrix} e^{+i\omega_0 t}$$

$$\langle S_x \rangle = \langle \psi(t) | S_x | \psi(t) \rangle = -\frac{\hbar}{2} \sin^2 \left\{ \frac{\omega_0 t}{2} \right\}$$

El valor esperado oscila, de manera que se puede interpretar como una precesión alrededor de \hat{x} .

PROBLEMA #7

a) En tal base, la matriz será $S_{1y} = S_y \otimes id$

$$S_{1y} = \frac{\hbar}{2} \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \frac{\hbar}{2} \begin{pmatrix} 0 & -i \cdot id \\ i \cdot id & 0 \end{pmatrix}$$

Los vec. prop. son

$$S_{1y} = -\frac{\hbar}{2} : \frac{1}{\sqrt{2}} (i, 0, 1, 0)^T ; \frac{1}{\sqrt{2}} (0, i, 0, 1)^T$$

$$S_{1y} = +\frac{\hbar}{2} : \frac{1}{\sqrt{2}} (-i, 0, 1, 0)^T ; \frac{1}{\sqrt{2}} (0, -i, 0, 1)^T$$

b) Tenemos 2 bases: una es la canónica $E = \{|++\rangle, |+-\rangle, |-+\rangle, |--\rangle\}$ y la otra es $B = \{|+_x+_y\rangle, |+_x-_-y\rangle, |-_x+_y\rangle, |-_x-_-y\rangle\}$. Construimos la matriz $T = [|+_x+_y\rangle \ |+_x-_-y\rangle \ |-_x+_y\rangle \ |-_x-_-y\rangle]$ de tal forma que

$$T^\dagger |\psi\rangle = \begin{pmatrix} \langle+_x+_y|\psi\rangle \\ \langle+_x-_-y|\psi\rangle \\ \langle-_-x+_y|\psi\rangle \\ \langle-_-x-_-y|\psi\rangle \end{pmatrix}$$

las amplitudes de probabilidad de obtener mediciones de S_{1x} S_{2y}

$$\begin{matrix} +\hbar/2 & +\hbar/2 \\ +\hbar/2 & -\hbar/2 \\ -\hbar/2 & +\hbar/2 \\ -\hbar/2 & -\hbar/2 \end{matrix}$$

Los valores se muestran en In[11], con las amplitudes podemos calcular la probabilidad.

Si $|\psi\rangle = (l|+\rangle + m|-\rangle) \otimes (n|+\rangle + o|-\rangle)$ tenemos que las amplitudes deben ser $\alpha = ln$, $\beta = lo$, $\delta = mo$, $\gamma = mn$.

c) Con el mismo procedimiento anterior, se obtienen los valores mostrados en In[13]. Si $|\psi\rangle$ es prod. tensorial, los coeficientes deben relacionarse de igual manera que en (b). (In[14])

d) En In[15] se calculan las probabilidades. Note que éstas son iguales, como se esperaba.

$$P_{S_{1-}, b} = |\langle+_x-_-y|\psi\rangle|^2 + |\langle-_-x-_-y|\psi\rangle|^2$$

$$P_{S_{1-}, c} = |\langle+_y-_-y|\psi\rangle|^2 + |\langle-_-y-_-y|\psi\rangle|^2$$

Sol_HW_10

May 13, 2018

```
In [1]: from sympy import *
init_printing()
from IPython.display import display
from sympy.physics.matrices import msigma
from sympy.physics.quantum.dagger import Dagger
from sympy.physics.quantum import Ket, Bra
from sympy.physics.quantum.state import Wavefunction
from sympy.physics.quantum import TensorProduct as TP
```

1 Problema 3

```
In [2]: h, B0, w0 = symbols('hbar B_0 omega_0', positive=True, real=True)
```

```
H = w0/sqrt(2) * msigma(1)*h/2 + w0/sqrt(2) * msigma(3)*h/2
display(H)
#0.25
```

$$\begin{bmatrix} \frac{\hbar\omega_0}{4}\sqrt{2} & \frac{\hbar\omega_0}{4}\sqrt{2} \\ \frac{\hbar\omega_0}{4}\sqrt{2} & -\frac{\hbar\omega_0}{4}\sqrt{2} \end{bmatrix}$$

```
In [3]: display(H.eigenvects())
vec1 = simplify(H.eigenvects()[0][2][0])
vec2 = simplify(H.eigenvects()[1][2][0])
val1 = simplify(H.eigenvects()[0][0])
val2 = simplify(H.eigenvects()[1][0])
Nsq=(Dagger(vec1)*vec1)[0]
vec1/=sqrt(Nsq)
display('sn_dn',simplify(vec1.evalf()))
Nsq=(Dagger(vec2)*vec2)[0]
vec2/=sqrt(Nsq)
display('sn_up',simplify(vec2.evalf()))
#0.25
```

$$\left[\left(-\frac{\hbar\omega_0}{2}, \quad 1, \quad \left[\left[-\frac{\sqrt{2}\hbar\omega_0}{4\left(\frac{\hbar\omega_0}{4}\sqrt{2}+\frac{\hbar\omega_0}{2}\right)} \right] \right] \right), \quad \left(\frac{\hbar\omega_0}{2}, \quad 1, \quad \left[\left[-\frac{2\sqrt{2}}{\hbar\omega_0} \left(-\frac{\hbar\omega_0}{2} - \frac{\hbar\omega_0}{4}\sqrt{2} \right) \right] \right] \right) \right]$$

'sn_dn'

$$\begin{bmatrix} -0.38268343236509 \\ 0.923879532511287 \end{bmatrix}$$

'sn_up'

$$\begin{bmatrix} 0.923879532511287 \\ 0.38268343236509 \end{bmatrix}$$

```
In [4]: th, phi, w = symbols('theta phi omega', real=True)
chi_n_up = Matrix([[cos(th/2)], [sin(th/2)*exp(I*phi)]])
chi_n_dn = Matrix([[sin(th/2)], [-cos(th/2)*exp(I*phi)]])
sn_up = chi_n_up.subs(th, pi/4).subs(phi, 0)
sn_dn = chi_n_dn.subs(th, pi/4).subs(phi, 0)
display('sn_up', sn_up.evalf())
display('sn_dn', sn_dn.evalf())
```

'sn_up'

$$\begin{bmatrix} 0.923879532511287 \\ 0.38268343236509 \end{bmatrix}$$

'sn_dn'

$$\begin{bmatrix} 0.38268343236509 \\ -0.923879532511287 \end{bmatrix}$$

```
In [5]: asq, bsq, t = symbols('a^2 b^2 t', positive=True, real=True)
minus = Matrix([[0], [1]])
solve(Eq(minus[0], asq*vec1[0] + bsq*vec2[0]), (asq, bsq))[0][asq]
solve(Eq(minus[1], asq*vec1[1] + bsq*vec2[1]), (asq, bsq))[0][asq]
b = solve(Eq(solve(Eq(minus[0], asq*vec1[0] + bsq*vec2[0]), \
                    (asq, bsq))[0][asq], solve(Eq(minus[1], asq*vec1[1] + bsq*vec2[1]), (asq, bsq))[0][asq]), \
          asq*vec1[1] + bsq*vec2[1]), (asq, bsq))[0][asq].subs(bsq, b)
display('P_sn_dn=a**2', (a**2).evalf(), 'P_sn_up=b**2', (b**2).evalf())
#Se pueden obtener -hw/2 con probabilidad a**2 y hw/2 con probabilidad b**2
```

'P_sn_dn=a**2'

0.853553390593274

```
'P_sn_up=b**2'
```

0.146446609406726

```
In [6]: minus_t = exp(-I*val1*t/h)*a*vec1 + exp(-I*val2*t/h)*b*vec2
display('psi_t',trigsimp(minus_t.evalf()))
```

```
'psi_t'
```

$$\begin{bmatrix} -0.353553390593274e^{\frac{i\omega_0}{2}t} + 0.353553390593274e^{-\frac{i\omega_0}{2}t} \\ 0.853553390593274e^{\frac{i\omega_0}{2}t} + 0.146446609406726e^{-\frac{i\omega_0}{2}t} \end{bmatrix}$$

```
In [7]: avg_Sx = h/2 * Dagger(minus_t)*msigma(1)*minus_t
display(simplify(avg_Sx[0].rewrite(sin)))
```

$$-\frac{\hbar}{2} \sin^2\left(\frac{\omega_0 t}{2}\right)$$

2 Problema 7

```
In [8]: from sympy.physics.quantum import TensorProduct as TP
S1y = h/2 * TP(msigma(2), eye(2))
display('S1y',S1y)
```

```
'S1y'
```

$$\begin{bmatrix} 0 & 0 & -\frac{i\hbar}{2} & 0 \\ 0 & 0 & 0 & -\frac{i\hbar}{2} \\ \frac{i\hbar}{2} & 0 & 0 & 0 \\ 0 & \frac{i\hbar}{2} & 0 & 0 \end{bmatrix}$$

```
In [9]: S1y.eigenvecs()
```

Out[9]:

$$\left[\left(-\frac{\hbar}{2}, 2, \begin{bmatrix} i \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ i \\ 0 \\ 1 \end{bmatrix} \right), \left(\frac{\hbar}{2}, 2, \begin{bmatrix} -i \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -i \\ 0 \\ 1 \end{bmatrix} \right) \right]$$

```
In [10]: a, b, g, d = symbols('alpha beta gamma delta', real=False)
Ket_p = Matrix([[1],[0]])
Ket_m = Matrix([[0],[1]])
Ket_pp = TP(Ket_p,Ket_p)
Ket_pm = TP(Ket_p,Ket_m)
Ket_mp = TP(Ket_m,Ket_p)
Ket_mm = TP(Ket_m,Ket_m)
init = a*Ket_pp + b*Ket_pm + g*Ket_mp + d*Ket_mm
init
```

Out[10]:

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{bmatrix}$$

```
In [11]: sx = [chi_n_up.subs(th,pi/2).subs(phi,0),chi_n_dn.subs(th,pi/2).subs(phi,0)]
sy = [chi_n_up.subs(th,pi/2).subs(phi,pi/2),chi_n_dn.subs(th,pi/2).subs(phi,pi/2)]
Ket_pxpy = TP(sx[0],sy[0])
Ket_pxmy = TP(sx[0],sy[1])
Ket_mxpy = TP(sx[1],sy[0])
Ket_mxmy = TP(sx[1],sy[1])
Base_change=Matrix([Ket_pxpy,Ket_pxmy,Ket_mxpy, Ket_mxmy]).reshape(4,4).transpose()
init_sxsy = Dagger(Base_change)*init
display(init_sxsy)
```

$$\begin{bmatrix} \frac{\alpha}{2} - \frac{i\beta}{2} - \frac{i\delta}{2} + \frac{\gamma}{2} \\ \frac{\alpha}{2} + \frac{i\beta}{2} + \frac{i\delta}{2} + \frac{\gamma}{2} \\ \frac{\alpha}{2} - \frac{i\beta}{2} + \frac{i\delta}{2} - \frac{\gamma}{2} \\ \frac{\alpha}{2} + \frac{i\beta}{2} - \frac{i\delta}{2} - \frac{\gamma}{2} \end{bmatrix}$$

```
In [12]: l, m, n, o = symbols('l m n o', real=False)
Ket_z1 = l*Ket_p + m*Ket_m
Ket_z2 = n*Ket_p + o*Ket_m
state= expand(TP(Ket_z1,Ket_z2))
state_sxsy = simplify(Dagger(Base_change)*state)
display(state_sxsy)
solve(init_sxsy-state_sxsy)
```

$$\begin{bmatrix} \frac{ln}{2} - \frac{il}{2}o + \frac{mn}{2} - \frac{im}{2}o \\ \frac{ln}{2} + \frac{il}{2}o + \frac{mn}{2} + \frac{im}{2}o \\ \frac{ln}{2} - \frac{il}{2}o - \frac{mn}{2} + \frac{im}{2}o \\ \frac{ln}{2} + \frac{il}{2}o - \frac{mn}{2} - \frac{im}{2}o \end{bmatrix}$$

Out[12]:

$$[\{\alpha : ln, \quad \beta : lo, \quad \delta : mo, \quad \gamma : mn\}]$$

```
In [13]: Ket_pypy = TP(sy[0],sy[0])
Ket_pymy = TP(sy[0],sy[1])
Ket_mypy = TP(sy[1],sy[0])
Ket_mymy = TP(sy[1],sy[1])
Base_change=Matrix([Ket_pypy,Ket_pymy,Ket_mypy, Ket_mymy]).reshape(4,4).transpose()
init_sysy = Dagger(Base_change)*init
display(init_sysy)
```

$$\begin{bmatrix} \frac{\alpha}{2} - \frac{i\beta}{2} - \frac{\delta}{2} - \frac{i\gamma}{2} \\ \frac{\alpha}{2} + \frac{i\beta}{2} + \frac{\delta}{2} - \frac{i\gamma}{2} \\ \frac{\alpha}{2} - \frac{i\beta}{2} + \frac{\delta}{2} + \frac{i\gamma}{2} \\ \frac{\alpha}{2} + \frac{i\beta}{2} - \frac{\delta}{2} + \frac{i\gamma}{2} \end{bmatrix}$$

```
In [14]: state_sysy = simplify(Dagger(Base_change)*state)
display(state_sysy)
solve(init_sysy-state_sysy)
```

$$\begin{bmatrix} \frac{ln}{2} - \frac{il}{2}o - \frac{im}{2}n - \frac{mo}{2} \\ \frac{ln}{2} + \frac{il}{2}o - \frac{im}{2}n + \frac{mo}{2} \\ \frac{ln}{2} - \frac{il}{2}o + \frac{im}{2}n + \frac{mo}{2} \\ \frac{ln}{2} + \frac{il}{2}o + \frac{im}{2}n - \frac{mo}{2} \end{bmatrix}$$

Out[14]:

$$[\{\alpha:ln, \quad \beta:lo, \quad \delta:mo, \quad \gamma:mn\}]$$

```
In [15]: #De b:
Pb = init_sxsys.conjugate().multiply_elementwise(init_sxsys)
display('Psy-, b', factor(Pb[1]+Pb[3]))
#De c:
Pc = init_sysy.conjugate().multiply_elementwise(init_sysy)
display('Psy-,c', factor(Pc[1]+Pc[3]))
```

'Psy-, b'

$$-\frac{1}{2}(-\alpha\bar{\alpha} + i\alpha\bar{\beta} - i\beta\bar{\alpha} - \beta\bar{\beta} - \delta\bar{\delta} - i\delta\bar{\gamma} + i\gamma\bar{\delta} - \gamma\bar{\gamma})$$

'Psy-,c'

$$-\frac{1}{2}(-\alpha\bar{\alpha} + i\alpha\bar{\beta} - i\beta\bar{\alpha} - \beta\bar{\beta} - \delta\bar{\delta} - i\delta\bar{\gamma} + i\gamma\bar{\delta} - \gamma\bar{\gamma})$$