énergie telle que, l'électron initialement dans l'état $|0\rangle$ se trouve à mi-chemin entre $|0\rangle$ et $|1\rangle$, dans l'état

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle). \tag{1.1.5}$$

On retient que

*La différence entre le bit classique et le bit quantique ou qubit se situe au niveau de la description d'un système physique (ses propriétés), étant entendu que l'**état** d'un système est l'ensemble des propriétés que le système possède. Si on pose par exemple la question **est-il possible de trouver la propriété** P **du système lors d'une mesure ?** La théorie classique répond **NON** alors que la théorie quantique répond **OUI mais avec une probabilité** $|\alpha|^2$ par exemple.*

1.1.2 Représentation de la sphère de Bloch avec QuTiP

En utilisant le logiciel QuTiP[2], on peut représenter sur la sphère de Bloch un qubit.

Les commandes suivantes représentent respectivement les états $|0\rangle$, $|1\rangle$ et $(|0\rangle + |1\rangle)/\sqrt{2}$. Le symbole # introduit un commentaire.

```
from qutip import *
B=Bloch() # crée une sphère de Bloch vide
ket0 = basis(2,0) # définit le |0⟩
ket1 = basis(2,1) # définit le |1⟩
B.add_states(ket0) # représente le |0⟩ sur B
B.add_states(ket1) # représente le |1⟩ sur B
B.add_states((ket0+ket1)/sqrt(2)) # représente l'état (|0⟩+|1⟩)/√2 sur B
B.show() # permet de visualiser la sphère B avec les trois états ajoutés.
```

L'image obtenue peut être sauvegardé pour utilisation ultérieur.

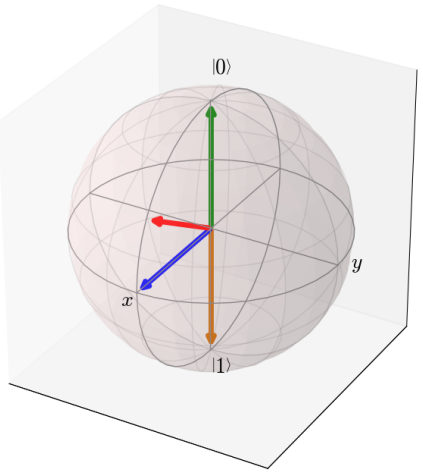

Figure 1.1.4 – *Représentation avec QuTiP de la sphère de Bloch avec les états* $|0\rangle$, $|1\rangle$ *et* $(|0\rangle + |1\rangle)/\sqrt{2}$.

[2]Voir l'Annexe A pour l'installation de QuTiP.

1. Appliquer le principe de superposition pour exprimer l'amplitude de probabilité $|\psi\rangle = \langle X\,|S\rangle$ du photon détecté au point X. Expliciter cette expression à l'aide du principe de factorisation séquentielle. En déduire l'expression de la probabilité $\mathcal{P}(X \leftarrow S)$ pour qu'un photon émis par la source S arrive en X en passant par l'une ou l'autre fente en faisant apparaître explicitement les termes d'*interférences quantiques*, en supposant que les amplitudes de probabilité de transition de la source S aux fentes F_1 et F_2 sont les mêmes.

2. Soient t_0 le temps de parcours de la source à l'une des fentes, t_1 le temps de parcours de la fente F_1 au point X, t_2 le temps de parcours de la fente F_2 au point X. Exprimer, en vertu de l'Eq. (1.4.1), chaque amplitude de probabilité de transition intermédiaire en fonction r_i et t_i et en déduire l'expression de $|\psi\rangle$.

3. Aux grandes distances, c'est-à-dire lorsque r_1 et r_2 sont beaucoup plus grands que la distance entre des fentes les amplitudes $\varphi(r_1)$ et $\varphi(r_2)$ sont à peu près égales et on peut écrire $\varphi(r_1) = \varphi(r_2) = C'$. Montrer que la probabilité pour qu'un photon émis par la source S arrive en X en passant par l'une ou l'autre fente est de la forme

$$\mathcal{P}(X \leftarrow S) = 2|C|^2 \left[1 + \cos\frac{\omega}{c}(r_2 - r_1)\right]. \tag{1.4.3}$$

4. Pour quelles valeurs de la différence de marche $r_2 - r_1$, en fonction de λ, a-t-on les franges brillantes et des franges sombres (voir la figure (1.4.3)) ?

1.4.3 Chat de Schrödinger

Nous reprenons dans cette exercice, la célèbre expérience de pensée de Schrödinger en considérant un chat enfermé dans une boîte avec un poison (substance radioactive) qui va déclencher la mort du chat à un moment ou un autre. Classiquement, le chat est soit mort, soit vivant. Quantiquement, on dit que le chat est dans une *superposition d'états mort ou vivant.*

1. Quelle base conceptuelle de la physique classique se trouve remise en cause par cette expérience ?

2. On se place dans la situation quantique. On désigne l'état mort et l'état vivant par les kets respectifs $|m\rangle$ et $|v\rangle$.

   (a) Exprimer dans la base $\{|m\rangle, |v\rangle\}$ un état quelconque $|\psi\rangle$ du chat.

   (b) Quelle condition doit être remplie pour que cet état traduise une onde de probabilité ?

   (c) Quels sont les deux états *limites* ? Quand les obtient-on ?

   (d) Soit t_1 le temps au bout duquel le chat a une chance sur deux d'être vivant et t_2 celui au bout duquel il a une chance sur quatre d'être vivant. Exprimer les états $|\psi_1\rangle$ et $|\psi_2\rangle$ du chat correspondant à ces instants.

3. En utilisant QuTiP, représenter sur une sphère de Bloch les états $|m\rangle$, $|v\rangle$, $|\psi_1\rangle$ et $|\psi_2\rangle$.

ANNEXE A

INSTALLATION DE QUTIP

QuTiP (**Qu**antum **T**oolbox **i**n **P**ython) est un logiciel *libre ou open-source* de calculs d'optique quantique avec des applications en information quantique. C'est un excellent outil d'appropriation et de simulation des concepts fondamentaux de la théorie quantique. Grâce à ce logiciel dont la maîtrise est aisée, l'étudiant peut facilement représenter un état quantique ou un opérateur, calculer une valeur moyenne, simuler l'évolution d'un système, implémenter des algorithmes de l'information quantique.

QuTiP étant rédiger en Python, il est nécessaire, pour utilisant optimale, d'avoir quelques notions de base du langage de programmation Python. Pour cela nous vous conseillons le cours gratuit *Apprenez à programmer en Python* disponible sur le site du zéro `http://uploads.siteduzero.com/pdf/223267-apprenez-a-programmer-en-python.pdf`. Il est important de souligner que Python est un langage de programmation **interprété**, c'est-à-dire que les instructions que vous lui envoyez sont transcrites en langage machine au fur et à mesure de leur lecture. Les langages comme le C / C++ ou le fortran sont appelés **langages compilés** car, avant de pouvoir les exécuter, un logiciel spécialisé se charge de transformer le code du programme en langage machine par la *compilation*. À chaque modification du code, il faut rappeler une étape de compilation.

Nous présentons ici les étapes à suivre pour l'installation de QuTiP sous le système d'exploitation libre **Linux**. Il est à noter que QuTiP fonctionne aussi sous les systèmes d'exploitation **Mac OS X** et **Windows**.

A.1 Installation de QuTiP pour Ubuntu 12.04 et plus récente

Pour commencer, il faut déjà un ordinateur sur lequel est installé une variante la distribution Linux Ubuntu (Ubuntu, Xubuntu, Kubuntu, Lubuntu, etc.). Se connecter à Internet pour télécharger les *paquets* nécessaires[1]

[1]Il faut aussi préciser qu'il y a deux possibilités d'installer QuTiP : manuellement et automatiquement. Nous allons présenter seulement l'installation automatique pour sa simplicité. Pour l'installation manuel et l'installation sur d'autres systèmes d'exploitation veuillez consulter le manuel de QuTiP *QuTiP : The Quantum Toolbox in Python release 2.2.0* disponible sur le site `http://qutip.googlecode.com/files/QuTiP-2.2.0-DOC.pdf`.

Aller sur le terminal et entrez la commande suivante, pour ajouter le dépôt de QuTiP à la *sourcelist* :
`sudo add-apt-repository ppa:jrjohansson/qutip-releases`
Mettre ensuite cette liste de dépôt grâce à la commande
`sudo apt-get update`
et installer QuTiP avec la commande
`sudo apt-get install python-qutip`

Il est recommandé d'installer d'autres paquets afin de compléter l'installation. Pour cela dans votre terminal, tapez les commandes suivantes :
`sudo apt-get install texlive-latex-extra`
`sudo apt-get install python-nose`

La dernière étape de la procédure consiste à installer une console Python ou interpréteur. A cet effet nous suggérons `IPython` ou `bpython` qui permet une bonne complétion. Taper sur le terminal
`sudo apt-get install ipython`
ou
`sudo apt-get install bpython`

A.2 Vérification de l'installation

Il est possible de vérifier si votre installation de QuTiP s'est bien dérouler. Le temps de cette vérification est fonction de la puissance de votre ordinateur. Pour vérifier, il faut taper sur terminal les commandes :

```
ipython # ou bpython si vous l'avez installé
import qutip.testing as qt
qt.run()
```

Une fois la vérification faite vous pouvez utiliser QuTiP.

ANNEXE E

CORRECTION DES EXERCICES

E.1 Qubits et états quantiques

E.1.1 Chat de Shrödinger

Les états $|\psi_1\rangle$ et $|\psi_2\rangle$ sont donnes par :

$$\begin{aligned} |\psi_1\rangle &= \frac{1}{\sqrt{2}}(|m\rangle + |v\rangle) \\ |\psi_2\rangle &= \frac{1}{2}(\sqrt{3}\,|m\rangle + |v\rangle). \end{aligned} \tag{E.1.1}$$

Pour les représenter sur la sphère de Bloch, on procède comme suit :

```
#Chat de Shrodinger

from qutip import *
from pylab import *

#Definition des etats morts et vivants : ces etats representent les etats 0 et 1
mort = basis(2,0)
vivant = basis(2,1)
#Definition des etats Psi_1 et Psi_2
Psi_1 = (mort+vivant)/sqrt(2)
Psi_2 = (sqrt(3)*mort+vivant)/2

B=Bloch()
B.add_states([mort,vivant,Psi_1,Psi_2])
B.show()
```

E.2 Mesure et opérateurs linéaires

E.2.1 Représentation matricielle

En utilisant QuTiP, répondre aux questions de l'exercice 2.5.1.

En QuTiP, le programme est le suivant :

```
In [65]: from qutip import *

In [66]: from pylab import *

In [67]:

In [67]: Phi_1 = basis(3,0)

In [68]: Phi_2 = basis(3,1)

In [69]: Phi_3 = basis(3,2)

In [70]:

In [70]: Psi_0 = Phi_1/sqrt(2)+1j*Phi_2/2+Phi_3/2

In [71]: Psi_1 = (Phi_1+1j*Phi_3)/sqrt(3)

In [72]:

In [72]: # Verification de la norme

In [73]: Psi_0.norm()
Out[73]: 1.0

In [74]: Psi_1.norm()
Out[74]: 0.81649658092772615

In [75]:

In [75]: # Calcul de P0 et P1

In [76]: P0 = Psi_0*Psi_0.dag()

In [77]: P1 = Psi_1*Psi_1.dag()

In [78]:

In [78]: # Hermiticite

In [79]: P0.isherm
Out[79]: True
```

```
In [80]: P1.isherm
Out[80]: True

In [81]:

In [81]: # Definir Y

In [82]: Y = sigmay()

In [83]:

In [83]: #Est-elle hermitienne?

In [84]: Y.isherm
Out[84]: True

In [85]:

In [85]: #Valeurs propres et vecteurs propres

In [86]: Val, Vec = Y.eigenstates()

In [87]:

In [87]: Vec1 = Vec[0]

In [88]: Vec2 = Vec[1]

In [89]:

In [89]: # Calcul des projecteurs sur ces etats

In [90]: P_1 = Vec1*Vec1.dag()

In [91]: P_2 = Vec2*Vec2.dag()

In [92]:

In [92]: # Relation de fermeture et d'orthogonalite

In [93]: P_1+P_2 # Ce resultat doit etre Identite
Out[93]:
Quantum object: dims = [[2], [2]], shape = [2, 2], type = oper, isherm = True
Qobj data =
[[ 1.  0.]
 [ 0.  1.]]
```

```
In [94]: P_1*P_2 # Celui-ci doit etre une matrice nulle
Out[94]:
Quantum object: dims = [[2], [2]], shape = [2, 2], type = oper, isherm = True
Qobj data =
[[ 0.  0.]
 [ 0.  0.]]
```

E.2.2 ECOC avec QuTiP

Cet exercice est lié à l'exercice (**??**). Écrire un programme en utilisant QuTiP permettant de répondre aux questions suivantes.

1. Définir les états $|u1\rangle$, $|u2\rangle$, $|u3\rangle$ ainsi que le hamiltonien H.
2. Calculer les énergies $E1$, $E2$, $E3$ ainsi que les vecteurs propres $|E_1\rangle$, $|E_2\rangle$, $|E_3\rangle$.
3. QuTiP permet de résoudre l'équation de Schrödinger et de calculer les états $|\psi(t)\rangle$ pour les valeurs de temps données. Il permet en même temps de calculer les valeurs moyennes voulues. Calculer pour $t \in [0, 10]$, les états les états $|\psi(t)\rangle$ lorsque l'état initial du système est $|u1\rangle$.
4. Calculer la déviation standard de H pour $t = 10s$.

E.2.3 ECOC Avec QuTiP

```
# ECOC

from qutip import *
from pylab import *

#Definition des etats de base.
u1 = basis(3,0)
u2 = basis(3,1)
u3 = basis(3,2)

#Definition de l'operateur H.
H = [[2, -3*sqrt(2), 3*sqrt(2)],[-3*sqrt(2), -1, -3],[3*sqrt(2), -3, -1]]
H = Qobj(H) # Rendre H un objet Quantique

#Calcul des energies et vecteurs propres de H
[E1, E2,E3], [ketE1, ketE2, ketE3] = H.eigenstates()

#Resolution de l'equation de Schrodinger
T = linspace(0,10,11) #Partitionner l'interval de temps
data = mesolve(H,u1,T,[],[H,H*H]) #Resolution de l'equation avec mesolve
Etat = data.states #Extraire les etats pour t dans [0,10]
MoyenH = data.expect[0] #Moyenne de H pour t dans [0,10]
MoyenHH = data.expect[1] # Moyenne de H*H pour t dans [0,10]
DeltaH = sqrt(MoyenHH[10]-MoyenH[10]**2)
```

E.3 Postulats et évolution

E.3.1 Évolution d'un état de spin 1/2

Le programme est le suivant :

```
from qutip import *
from pylab import *

#Hamiltonien H et ses etats propres
H = 0.25*sigmaz()
zp = basis(2,0)
zm = basis(2,1)
#Projecteurs sur les etats x+ et x-
xp = (zp+zm)/sqrt(2)
xm = (-zp+zm)/sqrt(2)
Pp = xp*xp.dag()
Pm = xm*xm.dag()
#Resolution de l' equation  de schrodinger
T = linspace(0,30,100)
data = mesolve(H, xp, T, [], [Pp,Pm,H,H*H])
P1 = data.expect[0]
P2 = data.expect[1]
MoyH = data.expect[2]
MoyHH = data.expect[3]
#Representation
plot(T, P1, T, P2)
xlabel('Temps $t$')
ylabel('Probabilite')
legend(("P+","P-"))
title(("Probabilite de transition"))
show()
#Calcul de la deviation standard
DeltaH = sqrt(MoyHH[99]-MoyH[99]**2)
```

E.4 Calculs quantiques

E.4.1 Circuit intraportation avec QuTiP

Un des programmes en QuTiP qui implémente le circuit intraportation est le suivant

```
# Teleportation d'une paire EPR sans bruit

from qutip import *
from pylab import *

# Definitions des operateurs intervenants dans le programme
X = sigmax()
Z = sigmaz()
```

```
Y = sigmay()
W = (X+Z)/sqrt(2)
I = qeye(2)

CX_12 = tensor(ket0*ket0.dag(), I, I) + tensor(ket1*ket1.dag(), X, I)
W_1 = tensor(W, I, I)
CX_23 = tensor(I,ket0*ket0.dag(), I) + tensor(I,ket1*ket1.dag(), X)
CZ_13 = tensor(ket0*ket0.dag(), I, I) + tensor(ket1*ket1.dag(), I, Z)

# Definition de B
W_2 = tensor(W,I)
CX_23_1 = tensor(ket0*ket0.dag(), I) + tensor(ket1*ket1.dag(), X)
B = CX_23_1*W_2
U = CZ_13*CX_23*W_1*CX_12 # Operateur d'evolution du circuit de la teleportation

#Generer l'etat EPR
ket00 = tensor(ket0,ket0)
EPR = B*ket00

# Definition de l'etat d'entree
ket0 = basis(2,0)
ket1 = basis(2,1)
Psi = (ket0+ket1)/sqrt(2)
Psi_in = tensor(Psi, EPR)

# Calcul de l'etat de sortie (Psi_out et Rho_out)
Psi_out = U*Psi_in
Rho_Bob = Psi_out.ptrace(2)
Prob = Psi.dag()*Rho_Bob*Psi
```

E.4.2 Téléportation d'une paire EPR

Une façon d'écrire ce programme en QuTiP est le suivant

```
# Teleportation d'une paire EPR

from qutip import *
from pylab import *

#Definitions des etats
ket0 = basis(2,0)
ket1 = basis(2,1)
Psi = (ket0+ket1)/sqrt(2)

# Definition des operateurs
X = sigmax()
Y = sigmay()
Z = sigmaz()
W = (X+Z)/sqrt(2)
```

```
I = qeye(2)

W_2 = tensor(I,W,I,I,I)
W_3 = tensor(W,I,I)
W_4 = tensor(I,I,I,W,I)
W_5 = tensor(I,I,I,I,W)

CX_12_1 = tensor(ket0*ket0.dag(), I) + tensor(ket1*ket1.dag(), X)
CX_12_2 = tensor(ket0*ket0.dag(), I, I) + tensor(ket1*ket1.dag(), X, I)
CX_13 = tensor(ket0*ket0.dag(), I, I) + tensor(ket1*ket1.dag(), I, X)
CX_14 = tensor(ket0*ket0.dag(),I,I,I,I) + tensor(ket1*ket1.dag(),I,I,X,I)
CX_23 = tensor(I,ket0*ket0.dag(),I,I,I) + tensor(I,ket1*ket1.dag(),X,I,I)
CX_24 = tensor(I,ket0*ket0.dag(),I,I,I) + tensor(I,ket1*ket1.dag(),I,X,I)
CX_34 = tensor(I,I,ket0*ket0.dag(),I,I) + tensor(I,I,ket1*ket1.dag(),X,I)
CX_35 = tensor(I,I,ket0*ket0.dag(),I,I) + tensor(I,I,ket1*ket1.dag(),I,X)
CX_54 = tensor(I,I,I,I,ket0*ket0.dag()) + tensor(I,I,I,X,ket1*ket1.dag())
CX_53 = tensor(I,I,I,I,ket0*ket0.dag()) + tensor(I,I,X,I,ket1*ket1.dag())
CX_51 = tensor(I,I,I,I,ket0*ket0.dag()) + tensor(X,I,I,I,ket1*ket1.dag())

#Definitions des etats
ket0 = basis(2,0)
ket1 = basis(2,1)
Psi = (ket0+ket1)/sqrt(2)
# Generation des etats EPR et GHZ
EPR = CX_12_1*tensor(Psi, ket1)
GHZ = CX_13*CX_12_2*W_3*tensor(ket0, ket0, ket0)
#etat d'entree
Psi_in = tensor(EPR,GHZ)

# Operateur d'evolution du circuit
U = CX_51*W_5*CX_14*CX_54*CX_35*W_4*CX_24*W_4*CX_34*W_2*CX_23*W_2

# Calcul de l'etat de sortie du circuit
Psi_out = U*Psi_in
Rho_5 = Psi_out.ptrace(4)

# Comparaison : calcul de la probabilite
Proba = Psi.dag()*Rho_5*Psi
```

