Homework #1 - ECEN 4005 - Sanjay Kumar Keshava

In [1]:
```
import numpy as np
import qutip as qt
from qutip.qip.operations import x_gate, y_gate, z_gate, s_gate, t_gate, snot, rx, ry, rz, swap ,iswap, swapalpha,cnot, cz_gate, globa

%matplotlib notebook
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

In [2]:
```
si = qt.qeye(2); sx = qt.sigmax(); sy = qt.sigmay(); sz = qt.sigmaz();
```

Problem 1(a) - Hermitian Operators

In [3]:
```
si
```

Out[3]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = True

$$\begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$$

In [4]:
```
sx
```

Out[4]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = True

$$\begin{pmatrix} 0.0 & 1.0 \\ 1.0 & 0.0 \end{pmatrix}$$

In [5]:
```
sy
```

Out[5]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = True

$$\begin{pmatrix} 0.0 & -1.0j \\ 1.0j & 0.0 \end{pmatrix}$$

In [6]:
```
sz
```

Out[6]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = True

$$\begin{pmatrix} 1.0 & 0.0 \\ 0.0 & -1.0 \end{pmatrix}$$

In [7]:
```
si.dag()
```

Out[7]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = True

$$\begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$$

In [8]:
```
sx.dag()
```

Out[8]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = True

$$\begin{pmatrix} 0.0 & 1.0 \\ 1.0 & 0.0 \end{pmatrix}$$

In [9]:
```
sy.dag()
```

Out[9]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = True

$$\begin{pmatrix} 0.0 & -1.0j \\ 1.0j & 0.0 \end{pmatrix}$$

In [10]:
```
sz.dag()
```

Out[10]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = True

$$\begin{pmatrix} 1.0 & 0.0 \\ 0.0 & -1.0 \end{pmatrix}$$

In [11]:
```
si.dag()-si
```

Out[11]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = True

$$\begin{pmatrix} 0.0 & 0.0 \\ 0.0 & 0.0 \end{pmatrix}$$

In [12]:
```
sx.dag()-sx
```

Out[12]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = True

$$\begin{pmatrix} 0.0 & 0.0 \\ 0.0 & 0.0 \end{pmatrix}$$

In [13]:
```
sy.dag()-sy
```

Out[13]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = True

$$\begin{pmatrix} 0.0 & 0.0 \\ 0.0 & 0.0 \end{pmatrix}$$

In [14]:
```
sz.dag()-sz
```

Out[14]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = True

$$\begin{pmatrix} 0.0 & 0.0 \\ 0.0 & 0.0 \end{pmatrix}$$

Problem 1(a) - Unitary Operators

In [15]:
```
si.dag()*si
```

Out[15]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = True

$$\begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$$

In [16]:
```
sx.dag()*sx
```

Out[16]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = True

$$\begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$$

In [17]:
```
sy.dag()*sy
```

Out[17]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = True

$$\begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$$

In [18]:
```
sz.dag()*sz
```

Out[18]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = True

$$\begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$$

Problem 1(a) - Eigenvalues and Eigenvectors

In [19]:
```
si.eigenstates()
```

Out[19]:
```
(array([1., 1.]),
 array([Quantum object: dims = [[2], [1]], shape = (2, 1), type = ket
        Qobj data =
        [[-1.]
         [ 0.]]                                                       ,
        Quantum object: dims = [[2], [1]], shape = (2, 1), type = ket
        Qobj data =
        [[0.]
         [1.]]                                                        ],
       dtype=object))
```

In [20]:
```
sx.eigenstates()
```

Out[20]:
```
(array([-1.,  1.]),
 array([Quantum object: dims = [[2], [1]], shape = (2, 1), type = ket
        Qobj data =
        [[-0.70710678]
         [ 0.70710678]]                                               ,
        Quantum object: dims = [[2], [1]], shape = (2, 1), type = ket
        Qobj data =
        [[0.70710678]
         [0.70710678]]                                                ],
       dtype=object))
```

In [21]:
```
sy.eigenstates()
```

Out[21]:
```
(array([-1.,  1.]),
 array([Quantum object: dims = [[2], [1]], shape = (2, 1), type = ket
        Qobj data =
        [[-0.70710678+0.j        ]
         [ 0.        +0.70710678j]]                                   ,
        Quantum object: dims = [[2], [1]], shape = (2, 1), type = ket
        Qobj data =
        [[-0.70710678+0.j        ]
         [ 0.        -0.70710678j]]                                   ],
       dtype=object))
```

In [22]:
```
sz.eigenstates()
```

Out[22]:
```
(array([-1.,  1.]),
 array([Quantum object: dims = [[2], [1]], shape = (2, 1), type = ket
        Qobj data =
        [[ 0.]
         [-1.]]                                                       ,
        Quantum object: dims = [[2], [1]], shape = (2, 1), type = ket
        Qobj data =
        [[-1.]
         [ 0.]]                                                       ],
       dtype=object))
```

Problem 1(b) - Commutation Relations

In [23]:
```
qt.commutator(sx, sy)
```
Out[23]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = False

$$\begin{pmatrix} 2.0j & 0.0 \\ 0.0 & -2.0j \end{pmatrix}$$

In [24]:
```
2*1j*sz
```
Out[24]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = False

$$\begin{pmatrix} 2.0j & 0.0 \\ 0.0 & -2.0j \end{pmatrix}$$

In [25]:
```
qt.commutator(sy, sz)
```
Out[25]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = False

$$\begin{pmatrix} 0.0 & 2.0j \\ 2.0j & 0.0 \end{pmatrix}$$

In [26]:
```
2*1j*sx
```
Out[26]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = False

$$\begin{pmatrix} 0.0 & 2.0j \\ 2.0j & 0.0 \end{pmatrix}$$

In [27]:
```
qt.commutator(sz, sx)
```
Out[27]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = False

$$\begin{pmatrix} 0.0 & 2.0 \\ -2.0 & 0.0 \end{pmatrix}$$

In [28]:
```
2*1j*sy
```
Out[28]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = False

$$\begin{pmatrix} 0.0 & 2.0 \\ -2.0 & 0.0 \end{pmatrix}$$

Problem 1(c) - Uncertainity Relation

In [29]:
```
comm_sx_sy = 2*1j*sz; comm_sx_sy
```
Out[29]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = False

$$\begin{pmatrix} 2.0j & 0.0 \\ 0.0 & -2.0j \end{pmatrix}$$

In [30]:
```
zero = qt.basis(2,0); zero
```
Out[30]: Quantum object: dims = [[2], [1]], shape = (2, 1), type = ket

$$\begin{pmatrix} 1.0 \\ 0.0 \end{pmatrix}$$

In [31]:
```
min_uncertainity = abs(((zero.dag()*comm_sx_sy*zero)/2)[0,0])
```

In [32]:
```
min_uncertainity
```
Out[32]:
```
1.0
```

Problem 4(a) - Equivalent Gates

In [33]:
```
sx - snot()*sz*snot()
```
Out[33]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = True

$$\begin{pmatrix} 0.0 & 0.0 \\ 0.0 & 0.0 \end{pmatrix}$$

Problem 4(b) - Equivalent Gates

In [34]:
```
snot()
```
Out[34]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = True

$$\begin{pmatrix} 0.707 & 0.707 \\ 0.707 & -0.707 \end{pmatrix}$$

In [35]:
```
rx(np.pi)*ry(np.pi/2)
```
Out[35]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = False

$$\begin{pmatrix} -0.707j & -0.707j \\ -0.707j & 0.707j \end{pmatrix}$$

In [36]:
```
snot() - (1j)*rx(np.pi)*ry(np.pi/2)
```

Out[36]: Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = True

$$\begin{pmatrix} 0.0 & 0.0 \\ 0.0 & 0.0 \end{pmatrix}$$

Problem 4(c) - Equivalent Gates

In [37]:
```
cnot() - qt.tensor(si, ry(np.pi/2))*cz_gate()*qt.tensor(si, ry(np.pi/(-2)))
```

Out[37]: Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = True

$$\begin{pmatrix} 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix}$$

Problem 5(a) - SWAP and iSWAP

In [38]:
```
plus = (qt.basis(2,0) + qt.basis(2,1)).unit(); minus = (qt.basis(2,0) - qt.basis(2,1)).unit();
plusminus = qt.tensor(plus,minus); plusminus
```

Out[38]: Quantum object: dims = [[2, 2], [1, 1]], shape = (4, 1), type = ket

$$\begin{pmatrix} 0.500 \\ -0.500 \\ 0.500 \\ -0.500 \end{pmatrix}$$

In [39]:
```
swap()*plusminus
```

Out[39]: Quantum object: dims = [[2, 2], [1, 1]], shape = (4, 1), type = ket

$$\begin{pmatrix} 0.500 \\ 0.500 \\ -0.500 \\ -0.500 \end{pmatrix}$$

In [40]:
```
qt.concurrence(swap()*plusminus)
```

Out[40]: 0

In [41]:
```
iswap()*plusminus
```

Out[41]: Quantum object: dims = [[2, 2], [1, 1]], shape = (4, 1), type = ket

$$\begin{pmatrix} 0.500 \\ 0.500j \\ -0.500j \\ -0.500 \end{pmatrix}$$

In [42]:
```
qt.concurrence(iswap()*plusminus)
```

Out[42]: 0.9999999864330515

Problem 5(b) - SWAP and iSWAP

In [43]:
```
onezero = qt.tensor(qt.basis(2,1),qt.basis(2,0)); onezero
```

Out[43]: Quantum object: dims = [[2, 2], [1, 1]], shape = (4, 1), type = ket

$$\begin{pmatrix} 0.0 \\ 0.0 \\ 1.0 \\ 0.0 \end{pmatrix}$$

In [44]:
```
iswap()*onezero
```

Out[44]: Quantum object: dims = [[2, 2], [1, 1]], shape = (4, 1), type = ket

$$\begin{pmatrix} 0.0 \\ 1.0j \\ 0.0 \\ 0.0 \end{pmatrix}$$

In [45]:
```
qt.concurrence(iswap()*onezero)
```

Out[45]: 0