Listing 6.22
Quantum Entangled States

Chrysanthi Tsiara
AM 4404

- Hamiltonian matrix
- Eigenvalues
- Eigenvectors
- Quantum states

```python
import numpy as np

nmax=4
H=np.zeros((nmax,nmax),float)
XAXB=np.array([[0,0,0,1],[0,0,1,0],[0,1,0,0],[1,0,0,0]])
YAYB=np.array([[0,0,0,-1],[0,0,1,0],[0,1,0,0],[-1,0,0,0]])
ZAZB=np.array([[1,0,0,0],[0,-1,0,0],[0,0,-1,0],[0,0,0,1]])
SASB=XAXB+YAYB+ZAZB-3*ZAZB
print('\n Hamiltonian without mu^2/r^3 factor \n',SASB,'\n')
```

$$H = \frac{\mu^2}{r^3}(X_A \otimes X_B Y_A \otimes Y_B + Z_A \otimes Z_B - 3Z_A \otimes Z_B).$$

```
Hamiltonian without mu^2/r^3 factor
[[-2  0  0  0]
 [ 0  2  2  0]
 [ 0  2  2  0]
 [ 0  0  0 -2]]
```

```python
es,ev=np.linalg.eig(SASB)
print ('Eigenvalues \n', np.round(es,2),'\n')
print('Eigenvalues (in columns) \n',ev,'\n')

phi1=ev[:,0]
phi4=ev[:,1]
phi3=ev[:,2]
phi2=ev[:,3]
```

```
Eigenvalues
 [ 4.  0. -2. -2.]

Eigenvalues (in columns)
 [[ 0.          0.          1.          0.        ]
 [ 0.70710678  0.70710678  0.          0.        ]
 [ 0.70710678 -0.70710678  0.          0.        ]
 [ 0.          0.          0.          1.        ]]
```

```python
basis=np.array([phi1,phi2,phi3,phi4])

for i in range(0,nmax):
    for j in range(0,nmax):
        term=np.dot(SASB,basis[i])
        H[i,j]=np.round(np.dot(basis[j],term),2)
print('Hamiltonian in Eigenvector Basis \n',H)
```

$$H = \begin{pmatrix} \langle\phi_1|H|\phi_1\rangle & \langle\phi_1|H|\phi_2\rangle & \langle\phi_1|H|\phi_3\rangle & \langle\phi_1|H|\phi_4\rangle \\ \langle\phi_2|H|\phi_1\rangle & \langle\phi_2|H|\phi_2\rangle & \langle\phi_2|H|\phi_3\rangle & \langle\phi_2|H|\phi_4\rangle \\ \langle\phi_3|H|\phi_1\rangle & \langle\phi_3|H|\phi_2\rangle & \langle\phi_3|H|\phi_3\rangle & \langle\phi_3|H|\phi_4\rangle \\ \langle\phi_4|H|\phi_1\rangle & \langle\phi_4|H|\phi_2\rangle & \langle\phi_4|H|\phi_3\rangle & \langle\phi_4|H|\phi_4\rangle \end{pmatrix}$$

```
Hamiltonian in Eigenvector Basis
 [[ 4.  0.  0.  0.]
 [ 0. -2.  0.  0.]
 [ 0.  0. -2.  0.]
 [ 0.  0.  0.  0.]]
```

Listing 6.23
SU3 Matrix Manipulations

- SU3 group of generators
- Basis vectors
- Raising and lowering operators

```python
import numpy as np

L1=np.array([[0,1,0],[1,0,0],[0,0,0]])
L2=np.array([[0,-1j,0],[1j,0,0],[0,0,0]])
L3=np.array([[1,0,0],[0,-1,0],[0,0,0]])
L4=np.array([[0,0,1],[0,0,0],[1,0,0]])
L5=np.array([[0,0,-1j],[0,0,0],[1j,0,0]])
L6=np.array([[0,0,0],[0,0,1],[0,1,0]])
L7=np.array([[0,0,0],[0,0,-1j],[0,1j,0]])
L8=np.array([[1,0,0],[0,1,0],[0,0,-2]])*1/np.sqrt(3)
```

$$\lambda_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \lambda_2 = \begin{bmatrix} 0 & -i & 0 \\ i & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \lambda_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$\lambda_4 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad \lambda_5 = \begin{bmatrix} 0 & 0 & -i \\ 0 & 0 & 0 \\ i & 0 & 0 \end{bmatrix}, \quad \lambda_6 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix},$$

$$\lambda_7 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -i \\ 0 & 1 & 0 \end{bmatrix}, \quad \lambda_8 = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -2 \end{bmatrix}.$$

```
u=np.array([1,0,0])
d=np.array([0,1,0])
s=np.array([0,0,1])
```

$$|\mathrm{u}\rangle = \begin{bmatrix}1\\0\\0\end{bmatrix}, \quad |\mathrm{d}\rangle = \begin{bmatrix}0\\1\\0\end{bmatrix}, \quad |\mathrm{s}\rangle = \begin{bmatrix}0\\0\\1\end{bmatrix}.$$

```
Ip=0.5*(L1+1j*L2)
Up=0.5*(L6+1j*L7)
Vp=0.5*(L4+1j*L5)
Im=0.5*(L1-1j*L2)
Um=0.5*(L6-1j*L7)
Vm=0.5*(L4-1j*L5)
```

$$I_\pm = \frac{1}{2}(\lambda_1 \pm i\lambda_2), \quad U_\pm = \frac{1}{2}(\lambda_6 \pm i\lambda_7), \quad V_\pm = \frac{1}{2}(\lambda_4 \pm i\lambda_5).$$

```
Ipxd=np.dot(Ip,d)
print("Ipdu",Ipxd)
Vpxs=np.dot(Vp,s)
print("Vpsu",Vpxs)
Upxs=np.dot(Up,s)
print("Upsd",Upxs)
```

```
Ipdu [1.+0.j 0.+0.j 0.+0.j]
Vpsu [1.+0.j 0.+0.j 0.+0.j]
Upsd [0.+0.j 1.+0.j 0.+0.j]
```

Thank you