Angular Momentum Matrices

Description

The following codes are done to calculate angular momentum matrices using `sympy`.

In [1]:
```
import sympy as sp
sp.init_printing(use_unicode=True)
```

In [2]:
```
hcut, w = sp.symbols('hbar omega')
```

Defining functions

- $J^2 \implies$ `J2_mat(j)`
- $J_z \implies$ `Jz_mat(j)`
- $J_x \implies$ `Jx_mat(j)`
- $J_y \implies$ `Jy_mat(j)`
- $J_+ \implies$ `J_plus_mat(j)`
- $J_- \implies$ `J_minus_mat(j)`

In [3]:
```
def m_J2_n(j, m, n):
    if m==n:
        return j*(j+1)*hcut**2
    else:
        return 0
J2_mat = lambda j: sp.Matrix(int(2*j+1),int(2*j+1), lambda m,n: m_J2_n(j, m, n))

def m_Jz_n(j, m, n):
    m, n = m-j, n-j
    if m==n:
        return n*hcut
    else:
        return 0
Jz_mat = lambda j: sp.Matrix(int(2*j+1),int(2*j+1), lambda m,n: m_Jz_n(j, m, n))

def m_J_plus_n(j, m, n):
    m, n = m-j, n-j
    if m==n-1:
        return sp.sqrt((j+n)*(j-n+1)) *hcut
    else:
        return 0
J_plus_mat = lambda j: sp.Matrix(int(2*j+1),int(2*j+1), lambda m,n: m_J_plus_n(j, m, n))

def m_J_minus_n(j, m, n):
    m, n = m-j, n-j
    if m==n+1:
        return sp.sqrt((j-n)*(j+n+1)) *hcut
    else:
        return 0
J_minus_mat = lambda j: sp.Matrix(int(2*j+1),int(2*j+1), lambda m,n: m_J_minus_n(j, m, n))

Jx_mat = lambda j: (J_plus_mat(j) + J_minus_mat(j))/2
Jy_mat = lambda j: (J_plus_mat(j) - J_minus_mat(j))/(2*sp.I)
```

Input value of j for which matrices will be calculated.

The same analysis can be done if we are interested in spin matrices. In that case, in our terms it would be donoted by S instead of J.

*For **Pauli Matrices**, we need to put `j1 = 1/2` and we will get the matrices $(\sigma^2, \sigma_z, \sigma_x, \sigma_y)$ with eigenvalues and eigenvectors.*

In [4]:
```
j1 = 1/2    # input
```

Angular momentum - matrix form

In [5]:
```
display('J2', J2_mat(j1), 'Jz', Jz_mat(j1),
'J+', J_plus_mat(j1), 'J-', J_minus_mat(j1),
'Jx', Jx_mat(j1), 'Jy', Jy_mat(j1))
```

'J2'

$$\begin{bmatrix} 0.75\hbar^2 & 0 \\ 0 & 0.75\hbar^2 \end{bmatrix}$$

'Jz'

$$\begin{bmatrix} -0.5\hbar & 0 \\ 0 & 0.5\hbar \end{bmatrix}$$

'J+'

$$\begin{bmatrix} 0 & 1.0\hbar \\ 0 & 0 \end{bmatrix}$$

'J-'

$$\begin{bmatrix} 0 & 0 \\ 1.0\hbar & 0 \end{bmatrix}$$

'Jx'

$$\begin{bmatrix} 0 & 0.5\hbar \\ 0.5\hbar & 0 \end{bmatrix}$$

'Jy'

$$\begin{bmatrix} 0 & -0.5i\hbar \\ 0.5i\hbar & 0 \end{bmatrix}$$

Eigenvalues and Eigenvectors

$$[\,(\,eigenvalue,\ degeneracy,\ [\,eigenvectors\,]\,)\,]$$

In [6]:
```
display('J2', J2_mat(j1).eigenvects(), 'Jz', Jz_mat(j1).eigenvects(),
'Jx', Jx_mat(j1).eigenvects(), 'Jy', Jy_mat(j1).eigenvects(),
# 'J+', J_plus_mat(j1).eigenvects(), 'J-', J_minus_mat(j1).eigenvects()
)
```

'J2'

$$\left[\left(0.75\hbar^2,\ 2,\ \left[\begin{bmatrix} 1.0 \\ 0 \end{bmatrix},\ \begin{bmatrix} 0 \\ 1.0 \end{bmatrix}\right]\right)\right]$$

'Jz'

$$\left[\left(-0.5\hbar,\ 1,\ \left[\begin{bmatrix} 1.0 \\ 0 \end{bmatrix}\right]\right),\ \left(0.5\hbar,\ 1,\ \left[\begin{bmatrix} 0 \\ 1.0 \end{bmatrix}\right]\right)\right]$$

'Jx'

$$\left[\left(-0.5\hbar,\ 1,\ \left[\begin{bmatrix} -1.0 \\ 1.0 \end{bmatrix}\right]\right),\ \left(0.5\hbar,\ 1,\ \left[\begin{bmatrix} 1.0 \\ 1.0 \end{bmatrix}\right]\right)\right]$$

'Jy'

$$\left[\left(-0.5\hbar,\ 1,\ \left[\begin{bmatrix} 1.0i \\ 1.0 \end{bmatrix}\right]\right),\ \left(0.5\hbar,\ 1,\ \left[\begin{bmatrix} -1.0i \\ 1.0 \end{bmatrix}\right]\right)\right]$$

In [ ]:

In [ ]: