# master_project

July 16, 2015

### 0.1 Derivation for the inner product of 3D rotational gradient operator acted on spherical harmonics

Chak-Pong CHUNG, chakpongchung@gmail.com

with application in object/scenes response to a lighting environment, often represented using spherical harmonics,including complex global illumination effects

General reference about spherical harmonics: http://mathworld.wolfram.com/SphericalHarmonic.html

# 1 Spherical harmonics in SymPy

To get back the well known expressions in spherical coordinates we use full expansion:

In [3]:
```
from sympy import Ynm, Symbol, expand_func
from sympy.abc import n,m
theta = Symbol("theta")
phi = Symbol("phi")
expand_func(Ynm(n, m, theta, phi))
```

Out[3]:
```
sqrt((2*n + 1)*factorial(-m + n)/factorial(m + n))*exp(I*m*phi)*assoc_legendre(n, m, cos(theta))
```

Relevant page http://docs.sympy.org/latest/modules/functions/special.html?highlight=spherical%20harmonics#sympy.fu

3d plot of spherical harmonics

http://stsdas.stsci.edu/download/mdroe/plotting/entry1/index.html

# 2 Derivation

We try to derive the dot product of rotational gradient of spherical harmonics with spherical harmonics expansion.

$$\vec{R} = \begin{bmatrix} R_x \\ R_y \\ R_z \end{bmatrix} = \begin{bmatrix} -\frac{cos\theta}{sin\theta}cos\phi\frac{\partial}{\partial\phi} - sin\phi\frac{\partial}{\partial\theta} \\ -\frac{cos\theta}{sin\theta}cos\phi\frac{\partial}{\partial\phi} + cos\phi\frac{\partial}{\partial\theta} \\ \frac{\partial}{\partial\phi} \end{bmatrix}$$

$$R_\alpha = iL_\alpha, where\, \alpha = x, y, z$$

$$\vec{R} = \begin{bmatrix} R_x \\ R_y \\ R_z \end{bmatrix} = \begin{bmatrix} iL_x \\ iL_y \\ iL_z \end{bmatrix} = \begin{bmatrix} \frac{i}{2}(L_+ + L_-) \\ \frac{1}{2}(L_+ - L_-) \\ iL_z \end{bmatrix}$$

$$L_+Y_l^m = c_{l,m,1}Y_l^{m+1}$$

$$L_-Y_l^m = c_{l,m,2}Y_l^{m-1}$$

$$L_z Y_l^m = m Y_l^m$$

after some manipulations of the above identities, it can be shown that (details will be added later)

$$\vec{R}Y_j^p \cdot \vec{R}Y_l^m = -\frac{1}{2}[c_{j,p,1}c_{l,m,2}Y_j^{p+1}Y_l^{m-1} + c_{j,p,2}c_{l,m,1}Y_j^{p-1}Y_l^{m+1}] - mpY_j^pY_l^m$$

$$= (-1) * (-1)^{m+p} \sum_{k=|l-j|}^{l+j} Y_k^{m+p}[\frac{1}{2}(c_{j,p,1}c_{l,m,2}\alpha_{l,m-1,j,p+1,k} + c_{j,p,2}C_{l,m,1}\alpha_{l,m+1,j,p-1,k}) + mp\alpha_{l,m,j,p,k}]$$

where

$$\alpha_{l,m,j,p,k} = \int_0^{2\pi}\int_0^{\pi} Y_j^p Y_l^m Y_k^{-m-p} \sin\theta d\theta d\phi$$

$$= [\frac{(2l+1)(2j+1)(2k+1)}{4\pi}]^{1/2} * \begin{pmatrix} j & l & k \\ 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} j & l & k \\ p & m & -(m+p) \end{pmatrix}$$

here

$$\begin{pmatrix} j & l & k \\ 0 & 0 & 0 \end{pmatrix}$$

is Wigner 3j symbol

# 3 Wigner 3j symbol

reference for Wigner 3j symbol(strange fraction here)
http://docs.sympy.org/dev/modules/physics/wigner.html

```
In [115]: from sympy.physics.wigner import wigner_3j
          wigner_3j(2, 6, 4, 0, 0, 0)
```

Out[115]:

$$\frac{\sqrt{715}}{143}$$

```
In [164]: sqrt(S(5)/143)
```

Out[164]:

$$\frac{\sqrt{715}}{143}$$

```
In [116]: wigner_3j(2, 6, 4, 0, 0, 1)
```

Out[116]:

$$0$$

and

$$Y_j^p = Y_j^p(\theta, \phi)$$

$$c_{l,m,1} = \sqrt{(l-m)(l+m+1)}$$

$$c_{l,m,2} = \sqrt{(l+m)(l-m+1)}$$

using some nice tricks of spherical harmonics , it can be shown that:

$$\frac{1}{2}(c_{j,p,1}c_{l,m,2}\alpha_{l,m-1,j,p+1,k} + c_{j,p,2}C_{l,m,1}\alpha_{l,m+1,j,p-1,k}) + mp\alpha_{l,m,j,p,k}$$

$$= \frac{1}{2}[\frac{(2l+1)(2j+1)(2k+1)}{4\pi}]^{1/2} * \begin{pmatrix} j & l & k \\ 0 & 0 & 0 \end{pmatrix} * [c_{j,p,1}c_{l,m,2}\begin{pmatrix} j & l & k \\ p+1 & m-1 & -(m+p) \end{pmatrix} + c_{j,p,2}c_{l,m,1}\begin{pmatrix} j & l \\ p-1 & m+1 & -( \end{pmatrix}$$

$$= \frac{1}{2}[\frac{(2l+1)(2j+1)(2k+1)}{4\pi}]^{1/2} * \begin{pmatrix} j & l & k \\ 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} k & j & l \\ -(m+p) & p & m \end{pmatrix} * (k^2 - j^2 - l^2 + k - j - (-2pm + 2pm))$$

$$= \frac{1}{2}[\frac{(2l+1)(2j+1)(2k+1)}{4\pi}]^{1/2} * \begin{pmatrix} k & j & l \\ 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} k & j & l \\ -(m+p) & p & m \end{pmatrix} * (k^2 - j^2 - l^2 + k - j - l)$$

$$= \alpha_{l,m,j,p,k} * \frac{1}{2}(k^2 - j^2 - l^2 + k - j - l)$$

### 3.1 so, in the end we have

$$\vec{R}Y_j^p \cdot \vec{R}Y_l^m = (-1)^{m+p} \sum_{k=|l-j|}^{l+j} Y_k^{m+p} * \alpha_{l,m,j,p,k} * \frac{1}{2}(k^2 - j^2 - l^2 + k - j - l)$$

```
In [161]: from sympy import *
          from sympy.physics.wigner import wigner_3j
          init_printing(use_latex='mathjax')

          def dot_rota_grad_SH(j, p, l, m, theta, phi):
              temp=0
              for k in range(abs(l-j), l+j +1):
                  temp+=Ynm(k, m+p, theta,phi)*alpha(l,m,j,p,k)/2*(k**2-j**2-l**2+k-j-l)

              return (-S(1))**(m+p)*temp

          def alpha(l,m,j,p,k):
              return sqrt((2*l+1)*(2*j+1)*(2*k+1)/(4*pi))*wigner_3j(j, l, k, 0, 0, 0)*wigner_3j(j, l, k


          print dot_rota_grad_SH(2,6,2,2,0,0)
```

```
0
```

These codes are now incorporated in SymPy, feel free to contact the author if you need help

```
In [167]: from sympy import Ynm, init_printing, Expr, var, sympify, Dummy, S, Sum, sqrt, pi
          from sympy.physics.wigner import wigner_3j
          # init_printing(use_latex='mathjax')
```

```
In [168]: class Wigner3j(Expr):
              def doit(self, **hints):
                  num = True
                  for i in range(6):
                      if not self.args[i].is_number:
                          num = False
                  if num:
```

```
                return wigner_3j(*self.args)
            else:
                return self

    def alpha(l,m,j,p,k):
        return sqrt((2*l+1)*(2*j+1)*(2*k+1)/(4*pi)) * Wigner3j(j, l, k, S(0), S(0), S(0))*Wigner3

    def dot_rota_grad_SH(j, p, l, m, theta, phi):
        j = sympify(j)
        p = sympify(p)
        l = sympify(l)
        m = sympify(m)
        theta = sympify(theta)
        phi = sympify(phi)
        k = Dummy("k")
        return (-S(1))**(m+p) * Sum(Ynm(k, m+p, theta, phi)*alpha(l,m,j,p,k)/2 *(k**2-j**2-l**2+k
```

In [169]:
```
var("j p l m theta phi")
#dot_rota_grad_SH(1, 5, 1, 1, 1, 2)
dot_rota_grad_SH(2,0,2,1,theta,phi)
```

Out[169]:

$$-\sum_{k=0}^{4} \frac{\sqrt{50k+25}}{4\sqrt{\pi}} \left(k^2+k-12\right) Y_{k}^{1}\left(\theta,\phi\right) Wigner3j(2,2,_k,0,0,0)Wigner3j(2,2,_k,0,1,-1)$$

In [170]:
```
print _.doit()
```

```
-3*sqrt(5)*Ynm(2, 1, theta, phi)/(14*sqrt(pi)) + 2*sqrt(30)*Ynm(4, 1, theta, phi)/(7*sqrt(pi))
```

In [171]:
```
_.expand(func=True)
```

Out[171]:

$$-\sum_{k=0}^{4} \left(\frac{\sqrt{50k+25}}{8\pi} k^2 \sqrt{\frac{2k(k-1)!}{(k+1)!} + \frac{(k-1)!}{(k+1)!}} e^{i\phi} P_{k}^{(1)}\left(\cos\left(\theta\right)\right) Wigner3j(2,2,_k,0,0,0)Wigner3j(2,2,_k,0,1,-1) + \frac{\sqrt{50k+25}}{8\pi} k \sqrt{\frac{2k(k-1)!}{(k+1)!} + \frac{(k-1)!}{(k+1)!}} e^{i\phi} P_{k}^{(1)}\left(\cos\left(\theta\right)\right) Wigner3j(2,2,_k,0,0,0)Wigner3j(2,2,_k,0,1,-1) - \frac{3e^{i\phi}}{2\pi} \sqrt{50k+25} \sqrt{\frac{2k(k-1)!}{(k+1)!} + \frac{(k-1)!}{(k+1)!}} P_{k}^{(1)}\left(\cos\left(\theta\right)\right) Wigner3j(2,2,_k,0,0,0)Wigner3j(2,2,_k,0,1,-1)\right)$$

In [173]:
```
for j in range(2):
    for p in range(-j, j+1):
        for l in range(2):
            for m in range(-l, l+1):
                print j, p, l, m, dot_rota_grad_SH(j, p, l, m, theta, phi).doit()
```

```
0 0 0 0 0
0 0 1 -1 0
0 0 1 0 0
0 0 1 1 0
1 -1 0 0 0
1 -1 1 -1 sqrt(30)*exp(-4*I*phi)*Ynm(2, 2, theta, phi)/(10*sqrt(pi))
1 -1 1 0 -sqrt(15)*exp(-2*I*phi)*Ynm(2, 1, theta, phi)/(10*sqrt(pi))
1 -1 1 1 Ynm(0, 0, theta, phi)/sqrt(pi) + sqrt(5)*Ynm(2, 0, theta, phi)/(10*sqrt(pi))
1 0 0 0 0
1 0 1 -1 -sqrt(15)*exp(-2*I*phi)*Ynm(2, 1, theta, phi)/(10*sqrt(pi))
1 0 1 0 -Ynm(0, 0, theta, phi)/sqrt(pi) + sqrt(5)*Ynm(2, 0, theta, phi)/(5*sqrt(pi))
1 0 1 1 sqrt(15)*Ynm(2, 1, theta, phi)/(10*sqrt(pi))
1 1 0 0 0
1 1 1 -1 Ynm(0, 0, theta, phi)/sqrt(pi) + sqrt(5)*Ynm(2, 0, theta, phi)/(10*sqrt(pi))
1 1 1 0 sqrt(15)*Ynm(2, 1, theta, phi)/(10*sqrt(pi))
1 1 1 1 sqrt(30)*Ynm(2, 2, theta, phi)/(10*sqrt(pi))
```

```
In [ ]:
```

```
In [ ]:
```