ChiTech Whitepaper: Spherical Harmonics

August, 2018

Jan Vermaak

Rev 1.00

Contents

List of Figures

List of Tables

1 Notes on Spherical Harmonics

1.1 Expansion of a function of two angles $f(\varphi,\theta)$

We seek to expand an angularly dependent function $f(\varphi,\theta)$. Why? Don't know yet, but one such expansion from fundamental theory is spherical harmonics:

$$f(\varphi,\theta)=\sum_{\ell=0}^{\infty}\sum_{m=-\ell}^{\ell} f_\ell^m Y_\ell^m(\varphi,\theta). \tag{1.1}$$

This expansion will be very unfamiliar for engineers since most of the scientific computing in common disciplines deal with cartesian coordinates. Unfortunately it will be nearly impossible to explain the development of an expansion of an angular function in spherical harmonics without first exploring the means to calculate its unknowns. To this end let us begin with stating that **there are two flavors of spherical harmonics**. The regular form Y_ℓ^m, which contains complex numbers (challenging to handle), and a real form $Y_{\ell m}$. The unknowns in equation 1.1 has a trail of components the first of which is

$$\begin{aligned}f_\ell^m &= \int_{\hat{\Omega}} f(\varphi,\theta)\,Y_\ell^{m^*}(\varphi,\theta)\,d\hat{\Omega}\\ &= \int_0^{2\pi}\int_0^{\pi} f(\varphi,\theta)Y_\ell^{m^*}(\varphi,\theta).\sin\theta.d\theta.d\varphi.\end{aligned}$$

The reader should try to comprehend that the f_ℓ^m components are almost never computed in this form since doing so means one already has a analytical representation of $f(\varphi,\theta)$. Additionally we have

$$Y_\ell^m(\varphi,\theta)=(-1)^m\sqrt{\frac{(2\ell{+}1)}{4\pi}\frac{(\ell{-}m)!}{(\ell{+}m)!}}.P_\ell^m(\cos\theta)e^{(m\varphi)i} \tag{1.2}$$

and its complex conjugate

$$Y_\ell^{m^*}(\varphi,\theta)=(-1)^m Y_\ell^{(-m)}(\varphi,\theta).$$

This form of the spherical harmonics functions can be very unruly and therefore its more common place to calculate them from the real forms as

$$Y_\ell^m(\varphi,\theta)=\begin{cases}\frac{1}{\sqrt{2}}(Y_{\ell|m|}{-}iY_{\ell,-|m|}) & \text{if } m<0\\ Y_{\ell 0} & \text{if } m=0\\ \frac{(-1)^m}{\sqrt{2}}(Y_{\ell|m|}{+}iY_{\ell,-|m|}) & \text{if } m>0\end{cases} \tag{1.3}$$

Here the real forms are represented by:

$$Y_{\ell m}(\theta,\varphi)=\begin{cases}(-1)^m\sqrt(2)\sqrt{\frac{(2\ell+1)}{4\pi}\frac{(\ell-|m|)!}{(\ell+|m|)!}}P_\ell^{|m|}(\cos\theta)sin\ |m|\varphi & \text{if } m<0\\ \sqrt{\frac{(2\ell+1)}{4\pi}}P_\ell^m(cos\theta) & \text{if } m=0\\ (-1)^m\sqrt(2)\sqrt{\frac{(2\ell+1)}{4\pi}\frac{(\ell-m)!}{(\ell+m)!}}P_\ell^m(\cos\theta)cos\ m\varphi & \text{if } m\geq 0\end{cases} \tag{1.4}$$

Finally the associated Legendre polynomials, P_ℓ^m can be determined from

$$\begin{aligned} P_0^0 = 1, \qquad\quad & P_1^0 = x, \\ P_\ell^\ell = -(2\ell-1)\sqrt{1-x^2}\; & P_{\ell-1}^{\ell-1}(x) \quad \text{and} \\ (\ell-m)P_\ell^m = (2\ell-1)x\; P_{\ell-1}^m(x) & -(\ell+m-1)P_{\ell-2}^m(x). \end{aligned} \tag{1.5}$$

With all these unknows we see that before we can calculate the expansion we need to choose the maximum order $L = \ell_{max}$ after which we need to compute each P_ℓ^m, each $Y_{\ell m}$, each Y_ℓ^m and each $Y_\ell^{m^*}$. Only then can we compute f_ℓ^m. If we do all the work this way we can approximate some functions.

1.2 Prototype code for spherical harmonics

We begin with code to compute the associated Legendre polynomials

```
def AssociatedLegendre(ell,m,x):
    if (abs(m)>ell):
        return 0;

    #====m=0,l=1
    Pn    = x;

    #====m=1,l=1
    Pnpos= -math.sqrt(1-math.pow(x,2));
    #====m=-1,l=1
    Pnneg= -0.5*Pnpos;

    #====m=0,l=0
    if (ell==0):
        return 1;

    if (ell==1):
        if (m==-1):
            return Pnneg;
        if (m==0):
            return Pn;
        if (m==1):
            return Pnpos;

    Pmlp1 = 0
    if (ell==m):
        Pmlp1 = -(2*ell-1)*math.sqrt(1-math.pow(x,2.0))* \
                AssociatedLegendre(ell-1,ell-1,x)
    else:
        Pmlp1 = (2*ell-1)*x*AssociatedLegendre(ell-1,m,x);
        Pmlp1 = Pmlp1 - (ell+m-1)*AssociatedLegendre(ell-2,m,x)
        Pmlp1 = Pmlp1 / (ell-m)

    return Pmlp1
```

We then depict code to calculate the real forms of the spherical harmonics

```
#==============================
def fac(x):
    if (x==0):
        return 1
    if (x==1):
        return 1

    return fac(x-1)*x;

#==============================
def Min1powerm(m):
    if (m==0):
        return 1;
    if ((m%2)==0):
        return 1
    else:
        return -1

#==============================
def Ylm(ell,m,varphi,theta):
    if (m<0):
        return Min1powerm(m)*math.sqrt( \
               ( (2*ell+1)/(2*math.pi) )* \
               fac(ell-abs(m))/fac(ell+abs(m)) )* \
               AssociatedLegendre(ell,abs(m),math.cos(theta))* \
               math.sin(abs(m)*varphi)
    elif (m==0):
        return math.sqrt( \
               ( (2*ell+1)/(4*math.pi) )* \
               fac(ell-m)/fac(ell+m) )* \
               AssociatedLegendre(ell,m,math.cos(theta))* \
               math.cos(m*varphi)
    else:
        return Min1powerm(m)*math.sqrt( \
               ( (2*ell+1)/(2*math.pi) )* \
               fac(ell-m)/fac(ell+m) )* \
               AssociatedLegendre(ell,m,math.cos(theta))* \
               math.cos(m*varphi)
```

And then the complex form of the spherical harmonics

```
#==============================
def Yl_m(ell,m,varphi,theta):
    v = 0+0j;
    if (m<0):
        v = (1/math.sqrt(2))*complex(Ylm(ell,abs(m),varphi,theta), - \
                                     Ylm(ell,-abs(m),varphi,theta))
    elif (m==0):
        v = complex(Ylm(ell,0,varphi,theta),0)
    else:
        v = Min1powerm(m)* \
        (1/math.sqrt(2))*complex(Ylm(ell,abs(m),varphi,theta), \
                                 Ylm(ell,-abs(m),varphi,theta))

    return v
```

From here we need to have a means to compute the integral

$$\begin{aligned} f_\ell^m &= \int_{\hat{\Omega}} f(\varphi,\theta)\, Y_\ell^{m^*}(\varphi,\theta)\, d\hat{\Omega} \\ &= \int_0^{2\pi}\int_0^{\pi} f(\varphi,\theta) Y_\ell^{m^*}(\varphi,\theta). \sin\theta . d\theta . d\varphi . \end{aligned}$$

Suppose we have any function of angle `F0`, `F1`, `F2` and so forth we can define a function that will add the complex conjugate of the spherical harmonics with the simple code

```
#==============================
def Flm(ell,m,varphi,theta):
    return F2(varphi,theta)*Min1powerm(m)* \
           (Legendre.Yl_m(ell,-m,varphi,theta))
```

We can then precompute a vector containing all the $f_{\ell m}$ coefficients using either a Riemann integral or a quadrature rule

```
GLC = Legendre.Quadrature()
GLC.InitializeWithGLC(8,8)


#============================= Build flm
L =7
k=-1
flm = np.zeros((L*(L+2)+1),dtype=np.complex_)
for ell in range(0,L+1):
    for m in range(-ell,ell+1):
        k=k+1
        #flm[k] = Legendre.RiemannAngLM(F1lm,ell,m)

        #print("l=%f, m=%f, flm=" %(ell,m), end='')
        #print(flm[k])
        flm[k] = Legendre.QuadratureIntegrateLM(Flm,GLC,ell,m)
        print("l=%f, m=%f, flm=" %(ell,m), end='')
        print(flm[k])
```

Finally we can compute a representation of the expansion over the span of φ using the code

```
#============================== Build yi1
Ni1=200
varphi1=np.linspace(0,2*math.pi,Ni1)
yi1=np.zeros((Ni1))

for i in range(0,Ni1):
    yi1[i]= 0;
    v = 0+0j
    k=-1;
    for ell in range(0,L+1):
        for m in range(-ell,ell+1):
            k=k+1
            v=v+(flm[k])* \
            (Legendre.Yl_m(ell,m,varphi1[i],math.pi/2))
            yi1[i] = v.real
```

1.2.1 Approximating an isotropic flux

The simplest function to represent is an isotropic flux (i.e. $f(\varphi, \theta) = 1$). Such a function is perfectly capture with $L = 0$, i.e. a single expansion, as shown in Figure 1.1. This is not surprising since the combination of spherical harmonics with order and moment zero results in $\sqrt{\frac{1}{4\pi}} \times \sqrt{\frac{1}{4\pi}}$ which integrates to unity and hence the original function is recovered.

```
def F0(varphi,theta):
    return 1
```

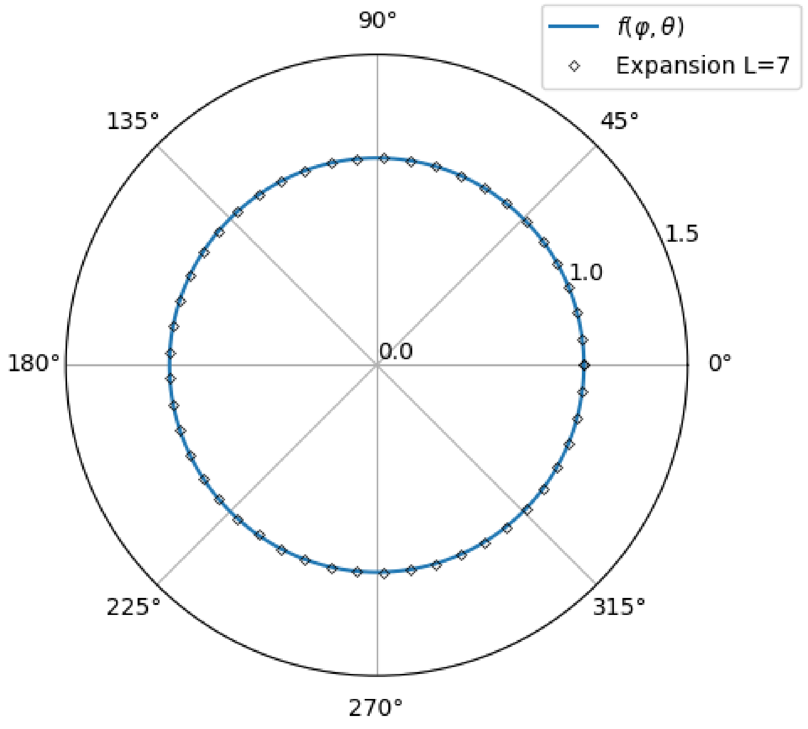

Figure 1.1: Approximation of a pure isotropic function with spherical harmonics. The plot is shown for the azimuthal angle φ only.

1.2.2 Approximating an anisotropic but smooth flux

We can construct an anisotropic function of azimuthal angle as

$$f(\varphi, \theta) = 1+\cos(4\varphi)$$

```
def F1(varphi,theta):
    return 1.0+0.1*math.cos(varphi*4)
```

Such a function requires a few more orders of spherical harmonics in order to capture the shape and, as shown in Figure 1.2, $L = 7$ closely resembles the shape. A function of this shape could appear in a fuel assembly lattice where the effective scattering and absorbtion is a strong function of azimuthal angle.

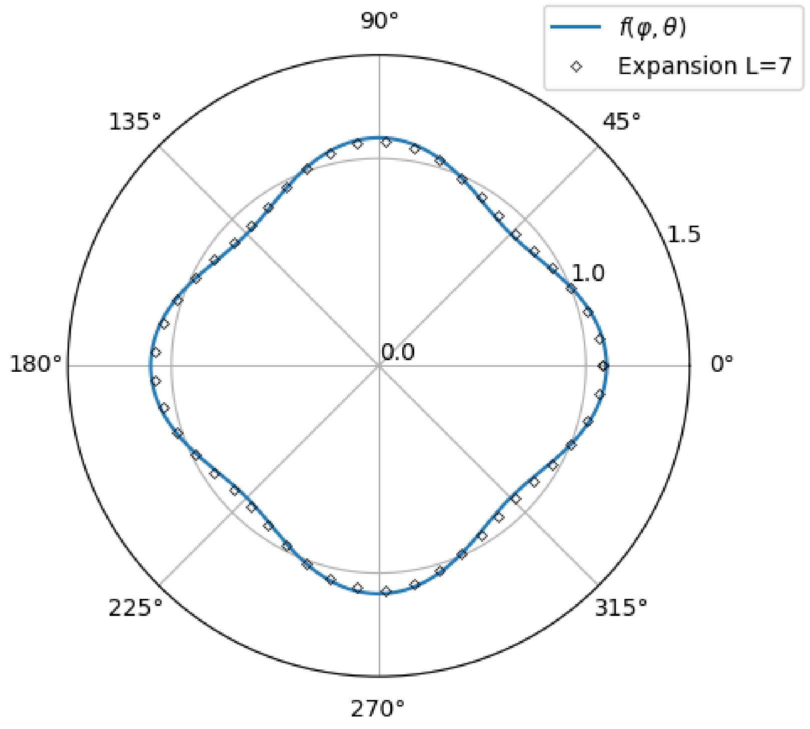

Figure 1.2: Approximation of an anisotropic smooth function with spherical harmonics. The plot is shown for the azimuthal angle φ only. The radial dimension represents the flux magnitude.

1.2.3 Approximating a directional flux (i.e. anisotropic + not-smooth)

As a final consideration we try to construct a function that is very angular, like a beam. Such a function of angle could be

$$f(\varphi,\theta) = \begin{cases} \frac{2}{10} & \text{if} \quad \varphi < \frac{7}{8}\pi \\ \frac{2}{10}+\frac{6}{5}\cos(4\varphi) & \text{if} \quad \frac{7}{8}\pi \leq \varphi \leq \frac{9}{8}\pi \\ \frac{2}{10} & \text{if} \quad \varphi > \frac{9}{8}\pi \end{cases}$$

```
def F2(varphi,theta):
    if (varphi<(7*math.pi/8)):
        return 0.2;
    if (varphi>(9*math.pi/8)):
        return 0.2;

    return 1.2*math.cos(varphi*4)+0.2
```

As expected a total number of 12 spherical harmonic orders are required to accurately represent such a directional flux (see Figure 1.3). An additional 2D plot is shown in Figure 1.4 which more clearly shows the oscillations of the expansion at the directions not aligned with the directional nature of the function.

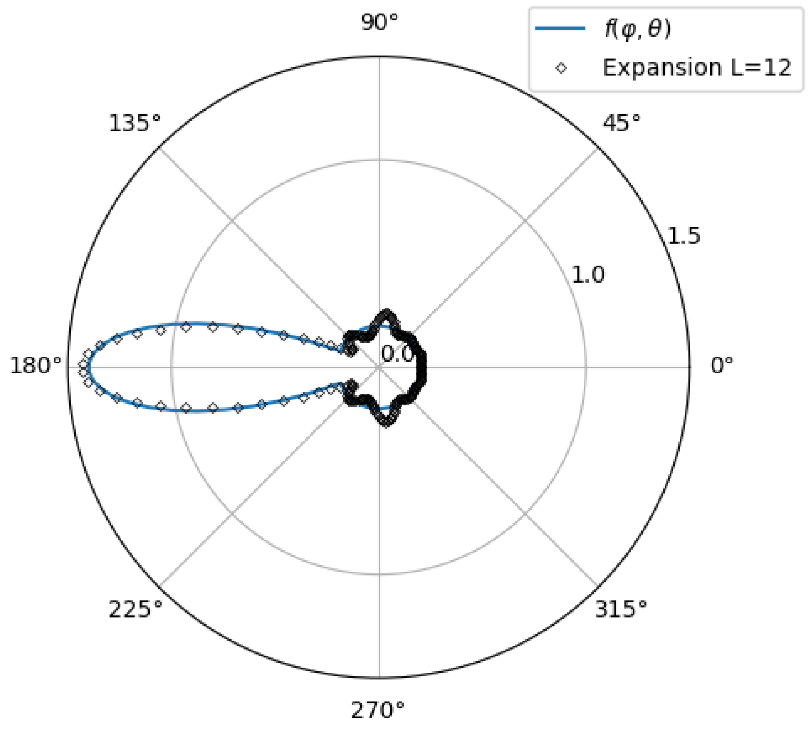

Figure 1.3: Approximation of an anisotropic smooth function with spherical harmonics. The plot is shown for the azimuthal angle φ only. The radial dimension represents the flux magnitude.

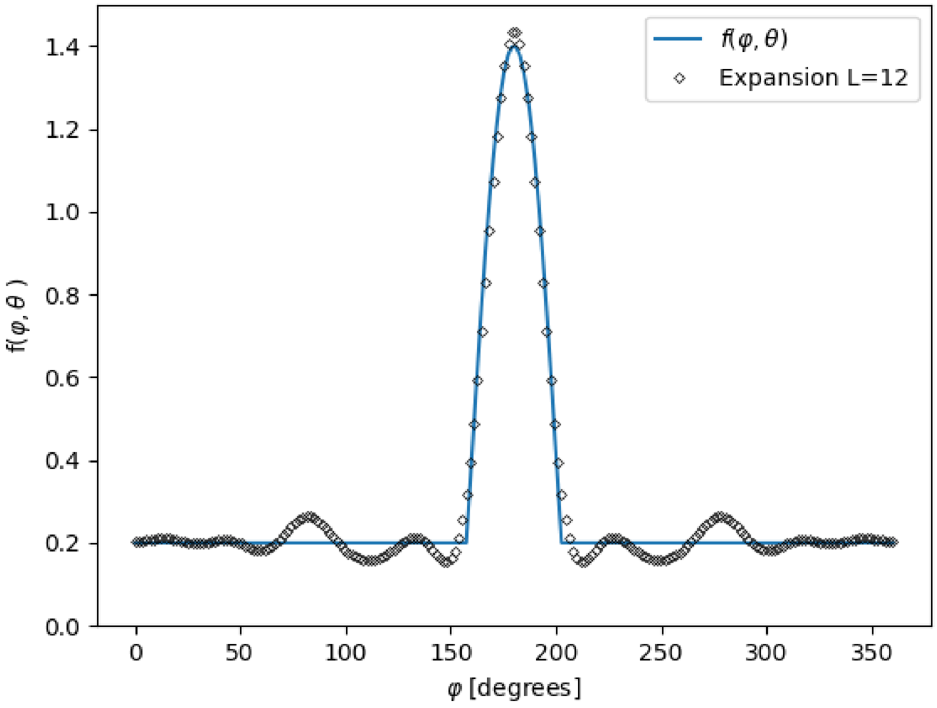

Figure 1.4: Approximation of an anisotropic smooth function with spherical harmonics. The plot is shown for the azimuthal angle φ only.

1.3 Prototype code for real form of the spherical harmonics

For the real form of the spherical harmonics we have a slightly modified real form

$$Y_{\ell m}(\theta,\varphi)=\begin{cases}\sqrt(2)\sqrt{\frac{(2\ell+1)}{4\pi}\frac{(\ell-|m|)!}{(\ell+|m|)!}}P_\ell^{|m|}(\cos\theta)sin\ |m|\varphi & \text{if } m<0\\ \sqrt{\frac{(2\ell+1)}{4\pi}}P_\ell^m(cos\theta) & \text{if } m=0\\ \sqrt(2)\sqrt{\frac{(2\ell+1)}{4\pi}\frac{(\ell-m)!}{(\ell+m)!}}P_\ell^m(\cos\theta)cos\ m\varphi & \text{if } m\geq 0\end{cases} \tag{1.6}$$

and the expansion coefficients are also different

$$f(\varphi,\theta)=\sum_{\ell=0}^{\infty}\sum_{m=-\ell}^{\ell}f_{\ell m}Y_{\ell m}(\varphi,\theta) \tag{1.7}$$

where

$$f_{\ell m}=\int_0^{2\pi}\int_0^{\pi}f(\varphi,\theta)Y_{\ell m}(\varphi,\theta).sin\theta.d\theta.d\varphi$$

for which the code is

```
#==============================
def Ylmcoeff(ell,m,varphi,theta):
    if (m<0):
        return math.sqrt( \
               ( (2*ell+1)/(2*math.pi) )* \
               fac(ell-abs(m))/fac(ell+abs(m)) )* \
               AssociatedLegendre(ell,abs(m),math.cos(theta))* \
               math.sin(abs(m)*varphi)
    elif (m==0):
        return math.sqrt( \
               ( (2*ell+1)/(4*math.pi) )* \
               fac(ell-m)/fac(ell+m) )* \
               AssociatedLegendre(ell,m,math.cos(theta))* \
               math.cos(m*varphi)
    else:
        return math.sqrt( \
               ( (2*ell+1)/(2*math.pi) )* \
               fac(ell-m)/fac(ell+m) )* \
               AssociatedLegendre(ell,m,math.cos(theta))* \
               math.cos(m*varphi)
```

And

```
def Flm(ell,m,varphi,theta):
    return F3(varphi,theta)* \
          (Legendre.Ylmcoeff(ell,m,varphi,theta))

#============================= Build flm
L =12
k=-1
flm = np.zeros((L*(L+2)+1),dtype=np.complex_)
for ell in range(0,L+1):
    for m in range(-ell,ell+1):
        k=k+1
        #flm[k] = Legendre.RiemannAngLM(F1lm,ell,m)

        #print("l=%f, m=%f, flm=" %(ell,m), end='')
        #print(flm[k])
        flm[k] = Legendre.QuadratureIntegrateLM(Flm,GLC,ell,m)
        print("l=%f,␣m=%f,␣flm=" %(ell,m), end='')
        print(flm[k])

#============================== Build yi1
Ni1=400
varphi1=np.linspace(0,2*math.pi,Ni1)
yi1=np.zeros((Ni1))

for i in range(0,Ni1):
    yi1[i]= 0;
    v = 0
    k=-1;
    for ell in range(0,L+1):
        for m in range(-ell,ell+1):
            k=k+1
            v=v+(flm[k])* \
            (Legendre.Ylmcoeff(ell,m,varphi1[i],math.pi/2))
            yi1[i] = v
```

References