

## 球谐光照——球谐函数



chopper

北京大学 计算机硕士

已关注

房燕良、夏寒、奔奔不笨笨、ycz、Yuumu 等 301 人赞同了该文章

早在1877年，Norman Macleod Ferrers就专门写了一本书来介绍球谐函数，后面物理学家把实数球谐函数扩展到复平面上，在复变函数论中作为“特殊函数”来研究，它在物理以及计算化学上有重要的应用，我们主要讨论它在计算机图形渲染上的应用。

球谐函数是拉普拉斯方程的分离 $r$ 变量后，角度部分通解的正交项，那本篇文章就从拉普拉斯方程开始介绍，直至找到我们想要的球谐函数。球谐函数有复数形式和实数形式，我们只关心它的实数形式。

球谐函数有两条重要的性质，正交完备性和旋转不变性。球谐函数构成的函数组，作为正交基，对信号进行投影和重建，例如[7]介绍的辐射度环境贴图，通过9个系数就可以模拟一张环境贴图的漫反射信号，无论是存储还是计算，都有显著的优势。当然，本篇文章希望从数学的角度，来介绍球谐函数，也受限于个人的数学能力，会忽略了一些复杂的推导过程。

文章目录：

- 球谐始源
- 球谐性质
- 附录
- 参考

## 球谐始源

球谐函数是拉普拉斯方程的分离  $r$  变量后，角度部分通解的正交项。这部分将从拉普拉斯方程开始逐项推导，得到最终我们想要知道的球谐函数。内容主要参考“姚端正,梁家宝. 数学物理方法-第4版”和“顾樵. 数学物理方法”两本书。

球面坐标可以表示为

$$\begin{cases} x = r \sin \theta \cos \varphi \\ y = r \sin \theta \sin \varphi \\ z = r \cos \theta \end{cases} \quad (1)$$

三维空间下的拉普拉斯 (Laplace) 方程可以表示为

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} = 0 \quad (2)$$

把球面坐标代入拉普拉斯方程 (参见“[Laplacian in Spherical Coordinates](#)”)，可以得到

$$\frac{1}{r^2} \frac{\partial}{\partial r} \left( r^2 \frac{\partial f}{\partial r} \right) + \frac{1}{r^2 \sin \theta} \frac{\partial}{\partial \theta} \left( \sin \theta \frac{\partial f}{\partial \theta} \right) + \frac{1}{r^2 \sin^2 \theta} \frac{\partial^2 f}{\partial \varphi^2} = 0 \quad (3)$$

我们的目标就是求解拉普拉斯方程的解。首先，把表示距离的变数  $r$  跟表示方向的变数  $\theta$  和  $\varphi$  分离，即

$$f(r, \theta, \varphi) = R(r) Y(\theta, \varphi) \quad (4)$$

$Y(\theta, \varphi)$  表示角度部分，把等式 (4) 代入等式 (3) 可以得到

$$\frac{Y}{r^2} \frac{\partial}{\partial r} \left( r^2 \frac{\partial R}{\partial r} \right) + \frac{R}{r^2 \sin \theta} \frac{\partial}{\partial \theta} \left( \sin \theta \frac{\partial Y}{\partial \theta} \right) + \frac{R}{r^2 \sin^2 \theta} \frac{\partial^2 Y}{\partial \varphi^2} = 0$$

等式两边乘以  $r^2/RY$ ，并移项，可得

$$\frac{1}{R} \frac{\partial}{\partial r} \left( r^2 \frac{\partial R}{\partial r} \right) = -\frac{1}{Y \sin \theta} \frac{\partial}{\partial \theta} \left( \sin \theta \frac{\partial Y}{\partial \theta} \right) - \frac{1}{Y \sin^2 \theta} \frac{\partial^2 Y}{\partial \varphi^2}$$

左边是  $r$  的函数，跟  $\theta$  和  $\varphi$  无关；右边是  $\theta$  和  $\varphi$  的函数，跟  $r$  无关。两边相等，显然是不可能的。除非两边实际上是同一个常数，通常把这个参数记为  $l(l+1)$ 。即

$$\frac{1}{R} \frac{\partial}{\partial r} \left( r^2 \frac{\partial R}{\partial r} \right) = -\frac{1}{Y \sin \theta} \frac{\partial}{\partial \theta} \left( \sin \theta \frac{\partial Y}{\partial \theta} \right) - \frac{1}{Y \sin^2 \theta} \frac{\partial^2 Y}{\partial \varphi^2} = l(l+1)$$

这就分解为两个方程

$$\frac{\partial}{\partial r} \left( r^2 \frac{\partial R}{\partial r} \right) - l(l+1) R = 0$$

$$\frac{1}{\sin \theta} \frac{\partial}{\partial \theta} \left( \sin \theta \frac{\partial Y}{\partial \theta} \right) + \frac{1}{\sin^2 \theta} \frac{\partial^2 Y}{\partial \varphi^2} + l(l+1) Y = 0 \quad (5)$$

当然，我们只关心角度部分的解，即方程 (5) 的解，这个方程也称为**球函数方程**。进一步采用分离变数法，以

$$Y(\theta, \varphi) = \Theta(\theta) \Phi(\varphi)$$

代入球函数方程，得

$$\frac{\Phi}{\sin \theta} \frac{\partial}{\partial \theta} \left( \sin \theta \frac{\partial \Theta}{\partial \theta} \right) + \frac{\Theta}{\sin^2 \theta} \frac{\partial^2 \Phi}{\partial \varphi^2} + l(l+1) \Theta \Phi = 0$$

在方程两边乘以  $\sin^2 \theta / \Phi \Theta$  并移项，即可得

$$\frac{\sin \theta}{\Theta} \frac{\partial}{\partial \theta} \left( \sin \theta \frac{\partial \Theta}{\partial \theta} \right) + l(l+1) \sin^2 \theta = -\frac{1}{\Phi} \frac{\partial^2 \Phi}{\partial \varphi^2}$$

左边是  $\theta$  的函数，跟  $\varphi$  无关；右边是  $\varphi$  的函数，跟  $\theta$  无关。两边相等显然是不可能的，除非两边实际上是同一个常数，这个常数记作  $\lambda$ ，

$$\frac{\sin \theta}{\Theta} \frac{\partial}{\partial \theta} \left( \sin \theta \frac{\partial \Theta}{\partial \theta} \right) + l(l+1) \sin^2 \theta = -\frac{1}{\Phi} \frac{\partial^2 \Phi}{\partial \varphi^2} = \lambda$$

这就又分解为两个常微分方程

$$\frac{\partial^2 \Phi}{\partial \varphi^2} + \lambda \Phi = 0 \quad (7)$$

$$\sin \theta \frac{\partial}{\partial \theta} \left( \sin \theta \frac{\partial \Theta}{\partial \theta} \right) + [l(l+1) \sin^2 \theta - \lambda] \Theta = 0 \quad (8)$$

对于常微分方程 (7)，它有一个隐含的“自然的周期条件”（ $\Phi(\varphi + 2\pi) = \Phi(\varphi)$ ），两者构成本征值问题。即

$$\lambda = m^2, (m = 0, \pm 1, \pm 2, \dots) \quad (9)$$

它的周期解用复数形式，可以表示为

$$\Phi(\varphi) = e^{im\varphi}, m = 0, \pm 1, \pm 2, \dots \quad (10)$$

严格来说，这里忽略了常数项，且是两个cos和sin函数的混合，或者是两个正负虚数的混合，但不影响最终的通解。

复数形式在复变函数论里面，有一个如下所示的转换关系，它们都是复平面上坐标的不同表现形式。由于本篇文章并不是为了严格的数学理论推导，而是为

了梳理球谐函数在计算机上从理论到应用这条线，最终我们采用的是实数形式的球谐函数，所以看看就可以了。

$$\cos m\varphi + i \sin m\varphi = e^{im\varphi}$$

再看常微分方程 (8)，它可以改写为

$$\frac{1}{\sin \theta} \frac{\partial}{\partial \theta} \left( \sin \theta \frac{\partial \Theta}{\partial \theta} \right) + \left[ l(l+1) - \frac{m^2}{\sin^2 \theta} \right] \Theta = 0$$

设  $x = \cos \theta$ , 则

$$\frac{\partial \Theta}{\partial \theta} = \frac{\partial \Theta}{\partial x} \frac{\partial x}{\partial \theta} = -\sin \theta \frac{\partial \Theta}{\partial x}$$

代入上式，化简可得

$$(1-x^2) \frac{\partial^2 \Theta}{\partial x^2} - 2x \frac{\partial \Theta}{\partial x} + \left[ l(l+1) - \frac{m^2}{1-x^2} \right] \Theta = 0 \quad (11)$$

这个方程就是  $l$  次连带勒让德方程，也称为缔合勒让德方程，其中， $m=0$  的特例，即

$$(1-x^2) \frac{\partial^2 \Theta}{\partial x^2} - 2x \frac{\partial \Theta}{\partial x} + l(l+1) \Theta = 0 \quad (12)$$

叫作  $l$  次勒让德方程。

后面只考虑连带勒让德方程，它的解就称为连带勒让德函数，只有当  $\lambda = l(l+1)$ ， $l=0,1,\dots$  时才有有界周期解，用  $P_l^m(x)$  表示，即

$$\Theta(\theta) = P_l^m(\cos \theta) \{m=0, \pm 1, \dots, \pm l\} \quad (13)$$

经过数学家论证，连带勒让德函数表示为

$$P_l^m(x) = \frac{(-1)^m (1-x^2)^{\frac{m}{2}}}{2^l l!} \frac{d^{l+m}}{dx^{l+m}} (x^2-1)^l \quad (14)$$

这叫称为l次m阶连带勒让德函数，“次”的英文是“degree”，“阶”的英文是“order”。当  $m > l$ 。连带勒让德函数里面有一个  $m+l$  次导数计算，在计算机上这个很难处理，但是有递归关系[3]，即

$$\begin{cases} (l-m) P_l^m(x) = x(2l-1) P_{l-1}^m(x) - (l+m-1) P_{l-2}^m(x) \\ P_m^m(x) = (-1)^m (2m-1)!! (1-x^2)^{m/2} \\ P_{m+1}^m(x) = x(2m+1) P_m^m(x) \end{cases} \quad (15)$$

两个!!表示双阶乘，即  $(2m-1)!! = 1 \cdot 3 \cdot 5 \cdots (2m-1)$ 。

连带勒让德函数的递归关系，保证了计算机实现的基础。

此外，再给一个l次m阶连带勒让德函数的关系等式

$$P_l^m(x) = (-1)^m \frac{(l+m)!}{(l-m)!} P_l^{-m}(x) \quad (16)$$

回到球函数方程（5）的求解，它的  $Y(\theta, \varphi)$  通解的复数形式表示为

$$Y(\theta, \varphi) = \sum_{l=0}^{\infty} \sum_{k=-l}^l P_l^k(\cos \theta) e^{im\varphi}, m = 0, \pm 1, \pm 2, \cdots \quad (17)$$

严格来说，由于  $\Phi(\varphi)$  忽略了常数项，这里也是忽略常数项的情况。

一般的l次m阶球谐函数  $Y_{lm}(\theta, \varphi)$  的复数形式可以表示为

$$Y_{lm}(\theta, \varphi) = P_{lm}(\cos \theta) e^{im\varphi}, m = 0, \pm 1, \pm 2, \cdots \quad (18)$$

$l$  表示球谐函数的次数， $m$  表示球谐函数的阶数

球谐函数的模长可以表示为

$$(N_l^m)^2 = \iint_S Y_{lm}(x) [Y_{lm}(x)]^* \sin \theta d\theta d\varphi = \frac{2}{2l+1} \frac{(l+|m|)!}{(l-|m|)!} 2\pi$$

归一化的球谐函数  $Y_l^m(\theta, \varphi)$  的复数形式可以表示为

$$Y_l^m(\theta, \varphi) = K_l^m Y_{lm}(\theta, \varphi) \tag{19}$$

其中

$$K_l^m = \frac{1}{N_l^m} = \sqrt{\frac{2l+1}{4\pi} \frac{(l-|m|)!}{(l+|m|)!}} \tag{20}$$

注意区分两种数学表示的含义， $Y_{lm}(\theta, \varphi)$  表示一般形式的球谐函数， $Y_l^m(\theta, \varphi)$  表示归一化的球谐函数。

当  $m > 0$  时采用实数  $\cos$  部分，当  $m < 0$  时采用虚数  $\sin$  部分，则归一化的球谐函数的实数形式可以表示为

$$Y_l^m(\theta, \varphi) = \begin{cases} \sqrt{2} K_l^m \cos(m\varphi) P_l^m(\cos \theta) & m > 0 \\ \sqrt{2} K_l^m \sin(-m\varphi) P_l^{-m}(\cos \theta) & m < 0 \\ K_l^0 P_l^m(\cos \theta) & m = 0 \end{cases} \tag{21}$$

根据上述计算公式，可以得到前面4次的球谐函数，为

l	m = -3	m = -2	m = -1	m = 0	m = 1	m = 2	m = 3
0				$\frac{1}{2}\sqrt{\frac{1}{\pi}}$			
1			$-\frac{1}{2}\sqrt{\frac{3}{\pi}}y$	$\frac{1}{2}\sqrt{\frac{3}{\pi}}z$	$-\frac{1}{2}\sqrt{\frac{3}{\pi}}x$		
2		$\frac{1}{2}\sqrt{\frac{15}{\pi}}yx$	$-\frac{1}{2}\sqrt{\frac{15}{\pi}}yz$	$\frac{1}{4}\sqrt{\frac{5}{\pi}}(3z^2-1)$	$-\frac{1}{2}\sqrt{\frac{15}{\pi}}zx$	$\frac{1}{4}\sqrt{\frac{15}{\pi}}(x^2-y^2)$	
3	$-\frac{1}{8}\sqrt{\frac{70}{\pi}}y(3x^2-y^2)$	$\frac{1}{2}\sqrt{\frac{105}{\pi}}xyz$	$-\frac{1}{8}\sqrt{\frac{42}{\pi}}y(5z^2-1)$	$\frac{1}{4}\sqrt{\frac{7}{\pi}}z(5z^2-3)$	$-\frac{1}{8}\sqrt{\frac{42}{\pi}}x(5z^2-1)$	$\frac{1}{4}\sqrt{\frac{105}{\pi}}z(x^2-y^2)$	$-\frac{1}{8}\sqrt{\frac{70}{\pi}}x(x^2-3y^2)$

前4次球谐函数

参见[6]，已经推导出前6次的球谐函数。

球谐函数可视化，前面几次的三维图像如图1所示

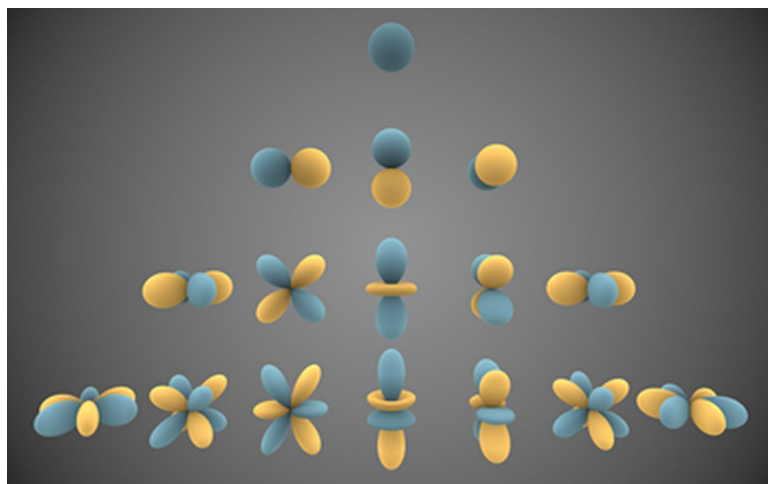


图1. 前4次的球谐函数三维图像

至此，你应该理解两条重要结论：

1. 球谐函数是拉普拉斯方程分离  $r$  变量后，角度部分通解的正交项（后面介绍正交性）。
2. 如何计算球谐函数，可以参见等式（15）（20）（21）。

## 球谐性质

接着，讨论归一化的球谐函数的性质，它具备两条重要的性质构成了它应用的基石：

- 正交完备性
- 旋转不变性

### 正交完备性

对于任意两个归一化的球谐函数在球面上的积分有



$$\iint_S Y_l^m(\theta, \varphi) Y_k^n(\theta, \varphi) \sin \theta d\theta d\varphi = \begin{cases} 0 & m \neq n, \text{ or } l \neq k \\ 1 & m = n, l = k \end{cases} \quad (21)$$

这就表示由球谐函数构成的函数组  $\{Y_l^m(\theta, \varphi)\}$  是正交归一化的。

以某一正交归一函数组为基，把一个给定的函数用这些函数的线性组合来表示，这就是一种重要的展开，这种用正交函数组展开为级数的一个显著的例子就是傅里叶变换。

任意一个球面函数  $f(\theta, \varphi)$  可以用正交归一的球函数  $Y_l^m(\theta, \varphi)$  进行展开，这种展开类似于傅里叶展开，称为**广义傅里叶展开**

$$f(\theta, \varphi) = \sum_{l=0}^{\infty} \sum_{m=-l}^l C_l^m Y_l^m(\theta, \varphi) \quad (22)$$

其中，广义傅里叶系数  $C_l^m$  为

$$C_l^m = \int_0^{2\pi} \int_0^{\pi} f(\theta, \varphi) Y_l^m(\theta, \varphi) \sin \theta d\theta d\varphi \quad (23)$$

当次数  $l \rightarrow \infty$  的时候，展开的级数和会**平均收敛**于  $f(\theta, \varphi)$ 。换句话说，当次数  $l$  越大，那么级数和就会越趋近于被展开的函数  $f(\theta, \varphi)$ ，就称  $\{Y_l^m(\theta, \varphi)\}$  为完备函数组。平均收敛，并不代表收敛，只是表示趋近于的含义。

从计算机的角度来说，如等式 (22) 所示的函数展开， $n$  的取值不可能是无穷大，往往取一个给定系数，则可以确定球谐函数组，例如  $n = 2$ ，那么球谐函数组就是

$$\{Y_l^m(\theta, \varphi)\} = \{Y_0^0, Y_1^{-1}, Y_1^0, Y_1^1\}$$

任意给定  $n$ ，得到的球谐函数组的个数为

$$S = 1 + 3 + 5 + \cdots 2n - 1 = n^2$$

那么，广义傅里叶系数相当于这样一个排列

$$C_0^0, C_1^{-1}, C_1^0, C_1^1, C_2^{-2}, C_2^{-1}, \dots$$

类似的球谐函数也可以构成这样一个类似的排列，若我们用一个普通的系数  $c_k$  来表示上面的广义傅里叶系数，用一个函数  $y_k(\theta, \varphi)$  来表示球谐函数，那么等式 (22) 可以换成另外一种形式

$$f(\theta, \varphi) = \sum_{k=0}^{n^2-1} c_k y_k(\theta, \varphi) \quad (24)$$

这种形式的展开与等式 (22) 是完全一样的，它只是把球谐函数的次数展开，用一个系数来表示，但是它隐藏了一个条件：取的系数个数必需是  $n^2$ 。

反过来，球谐函数组相当于一组正交基，将函数  $f(\theta, \varphi)$  表示为这组正交基的线性组合，生成线性组合系数的过程就称为**投影** (Projection)，例如一个函数可以表示为

$$f(\theta, \varphi) \approx aY_0^0 + bY_1^{-1} + cY_1^0 + dY_1^1$$

生成系数  $\{a, b, c, d\}$  的过程，就是投影，等式 (23) 就确定了投影的方法。相反，利用这组系数和正交基组合，得到原函数的过程，就称为**重建** (Reconstruction)。

投影的过程就是计算函数积分，计算消耗较大，可以采用离线处理来生成广义傅里叶系数；在实时渲染时，就要简单的线性组合，就可以重建原始函数。当然，由于是有限个系数，就必然存在误差。

我们再来看下连带勒让德函数，如图2所示，随着次数的增加，函数的振动频率会越快。对于函数的展开而言，振动频率越大的基底，它就只能表示越高频的信息，往往一个函数里面的高频信息量是较少的。

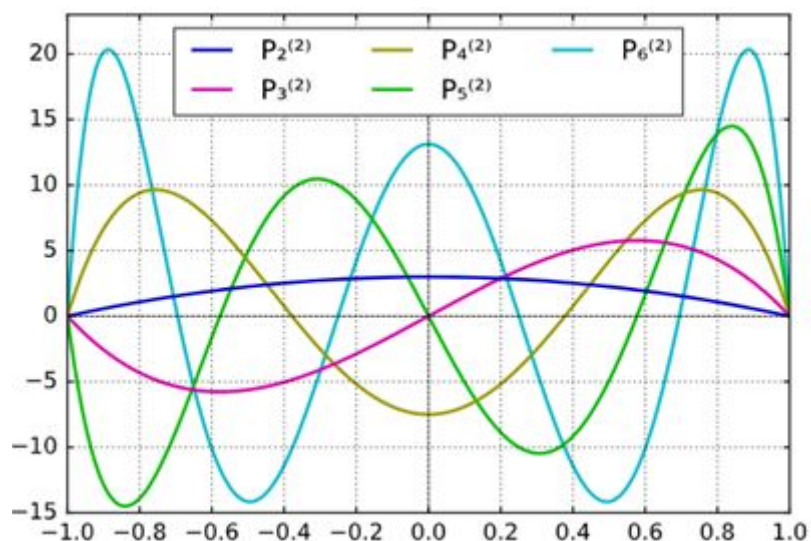


图2. 连带勒让德函数曲线图，次数越大，振频越快

类似的，球谐函数也具备这种随着次数增加，振动频率增加的特性。它就使得 $n$ 的取值不需要很大时，就可以得到很好的重建效果，当然只能还原出低频信息。根据Robin[3]得到的数据，如图3所示，当 $n > 6$ 时，就能还原出整体效果，但是边缘棱角这些高频信息是无法还原出来的。

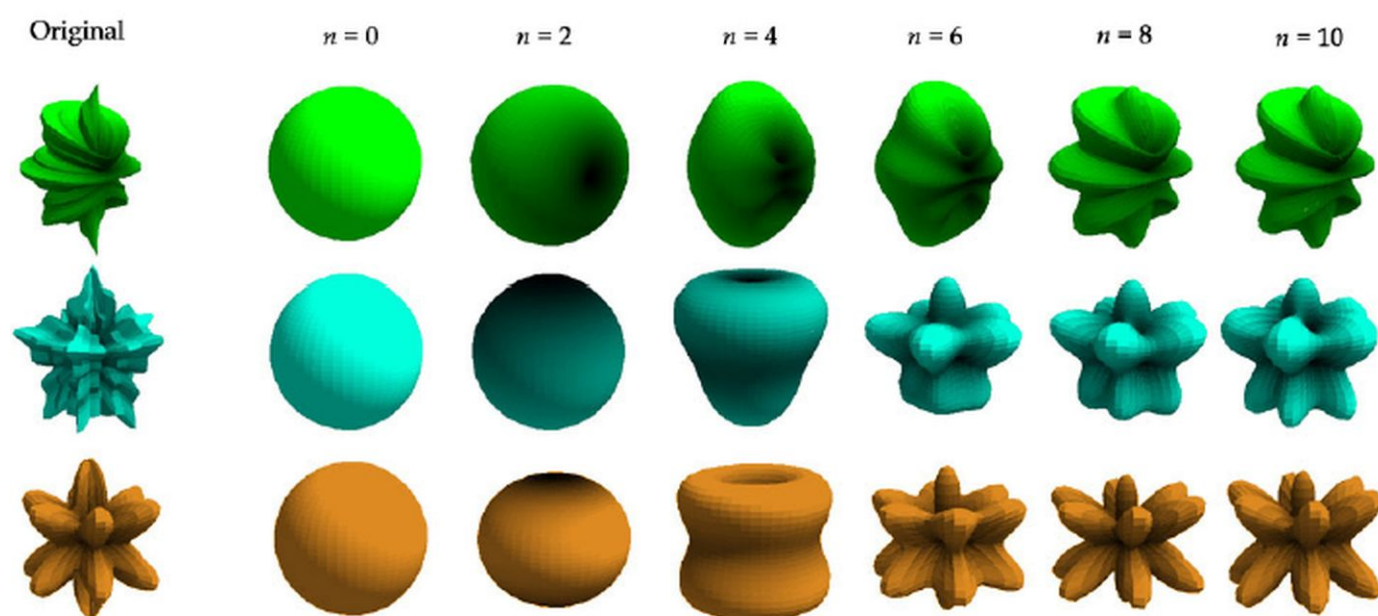


图3. 不同 $n$ 取值下球谐函数的重建效果

由球谐函数构成的函数组构成正交归一的基底，对球面上的函数进行投影和重建，也就是广义傅里叶展开，数学上的完备性，保证了展开结果会趋近于被展开的函数。

## 旋转不变性

第一个问题是：什么叫旋转不变性。

任意一个球面上的函数  $f(\theta, \varphi)$  可以用球谐函数组作为基底展开，需要根据等式 (23) 计算广义傅里叶系数  $C_l^m$ 。如果我们对原函数进行旋转操作的话，设旋转变换表示为  $R(\theta, \varphi)$ ，我们就得到了一个新的函数  $f(R(\theta, \varphi))$ 。对新函数进行展开的话，我们需要重新计算广义傅里叶系数，设为  $B_l^m$ ，这就有点为难了。在图形渲染中，广义傅里叶系数的生成是离线实现的，它的消耗很大，这就表示，一旦光源发生了旋转后，由于原函数的改变导致提前生成的系数失效。**旋转不变性**，表示原函数发生了旋转，只需要对生成的广义傅里叶系数进行变换，就能保证变换后的系数能等价还原出新函数。在图形渲染上的表现就是，当光源发生旋转后，我们只要同步的计算出变换后的广义傅里叶系数，就能保证画面的光照效果不会抖动跳变。旋转不变性，并不是表示源函数发生旋转后，对重建结果没有影响，而是表示通过对系数与匹配的旋转进行变换后，能等价的还原出旋转后的函数。

举个[6]实验的例子，如图4所示，球谐函数表示的光照发生旋转后，仍然能等价重建新的变换函数，但是采用Ambient Cube的方法效果就出现了异常。

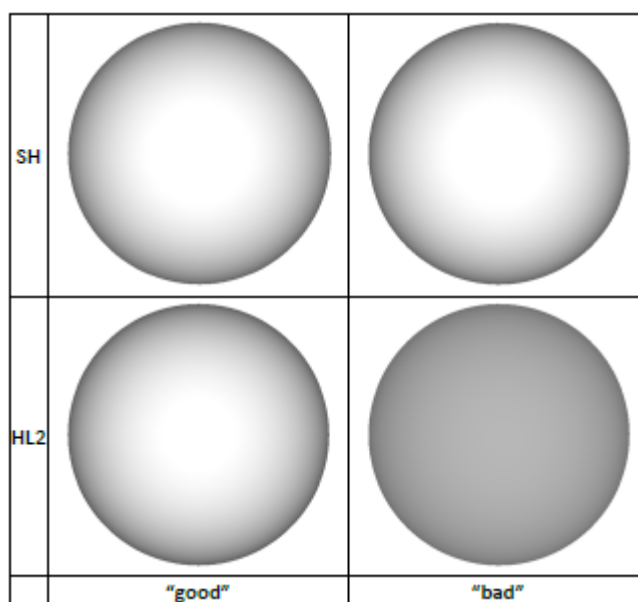


图4. SH表示球谐基底，HL2表示Ambient Cube基底

第二个问题是：怎么对生成的系数进行变换。

针对这个问题，这里写些自己的理解，不做深入的研究。

对于 $l$ 次球谐函数，就会有 $2l + 1$ 个系数，表示为

$$C_l = \{C_l^{-l}, C_l^{-l+1}, \dots, C_l^{l-1}, C_l^l\} \quad (25)$$

设变换矩阵为  $R_{SH}^l$ ，它是一个 $(2l + 1) \times (2l + 1)$ 的矩阵，那么系数的变换就可以表示为

$$B_l^m = \sum_{k=-l}^{k=l} M_l^{m,k} C_l^k \quad (26)$$

或者用向量与矩阵的乘积形式，表示为

$$B_l = C_l \cdot R_{SH}^l \quad (27)$$

那么，经过旋转变换后的函数的展开就可以表示为

$$f(R(\theta, \varphi)) = \sum_{l=0}^{\infty} \sum_{m=-l}^l B_l^m Y_l^m(\theta, \varphi) \quad (28)$$

唯一的区别就是，系数由  $C_l$  变成了  $B_l$ 。

想表达的一点是，系数的变换是基于球谐函数的次数，即第3次球谐函数的系数  $B_3$ ，只能由第3次球谐函数的系数  $C_3$  变换而来。

若取前3次的球谐函数构成正交基，函数组共有0，1，2次三类球谐函数，若采用等式（24）的形式，则3个子矩阵需要整合成一个完整的变换矩阵，对于前3次球谐函数的例子，就组成一个9x9的变换矩阵，它的形状如下所示。

$$\begin{pmatrix} X & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & X & X & X & 0 & 0 & 0 & 0 & 0 \\ 0 & X & X & X & 0 & 0 & 0 & 0 & 0 \\ 0 & X & X & X & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & X & X & X & X & X \\ 0 & 0 & 0 & 0 & X & X & X & X & X \\ 0 & 0 & 0 & 0 & X & X & X & X & X \\ 0 & 0 & 0 & 0 & X & X & X & X & X \\ 0 & 0 & 0 & 0 & X & X & X & X & X \end{pmatrix}$$

9x9的球谐系数变换矩阵

考虑低维的情况[3]。旋转可以用旋转矩阵、欧拉角、四元数等方式表示，任意一个旋转矩阵  $R$  可以用  $Z_\alpha Y_\beta Z_\gamma$  型的欧拉角表示，它们间的变换关系[5]表示为

$$\begin{pmatrix} R_{0,0} & R_{0,1} & R_{0,2} \\ R_{1,0} & R_{1,1} & R_{1,2} \\ R_{2,0} & R_{2,1} & R_{2,2} \end{pmatrix} = \begin{pmatrix} c_\alpha c_\beta c_\gamma - s_\alpha s_\gamma & c_\alpha s_\gamma + s_\alpha c_\beta c_\gamma & -s_\beta c_\gamma \\ -s_\alpha c_\gamma - c_\alpha c_\beta s_\gamma & c_\alpha c_\gamma - s_\alpha c_\beta s_\gamma & s_\beta s_\gamma \\ c_\alpha s_\beta & s_\alpha s_\beta & c_\beta \end{pmatrix} \quad (29)$$

其中，c表示cos，s表示sin。

有了这个变换关系后，就很容易计算出欧拉角  $\alpha, \beta, \gamma$ ，表示为

$$\begin{cases} \sin \beta = \sqrt{1 - R_{2,2}^2} \\ \alpha = \text{atan2f}(R_{2,1} / \sin \beta, R_{2,0} / \sin \beta) \\ \beta = \text{atan2f}(\sin \beta, R_{2,2}) \\ \gamma = \text{atan2f}(R_{1,2} / \sin \beta, -R_{0,2} / \sin \beta) \end{cases}$$

对于， $R_{2,2} = 1$  的退化情况，欧拉角表示为

$$\begin{cases} \alpha = \text{atan2f}(R_{0,1}, R_{0,0}) \\ \beta = 0 \\ \gamma = 0 \end{cases}$$

那么，相应 $l$ 次的球谐系数的变换矩阵可以表示为

$$R_{SH}^l(\alpha, \beta, \gamma) = Z_\gamma Y_{-90} Z_\beta Y_{+90} Z_\alpha$$

对于第0次的球谐变换矩阵为

$$R_{SH}^0(\alpha, \beta, \gamma) = (1) \quad (30)$$

其它维度的矩阵推导比较麻烦，就推导了第1次的球谐变换矩阵，它可以表示为

$$R_{SH}^1(\alpha, \beta, \gamma) = \begin{pmatrix} c_\alpha c_\gamma - s_\alpha c_\beta s_\gamma & -s_\beta s_\gamma & -s_\alpha c_\gamma - c_\alpha c_\beta s_\gamma \\ -s_\alpha s_\beta & c_\beta & -c_\alpha s_\beta \\ c_\alpha s_\gamma + s_\alpha c_\beta c_\gamma & s_\beta c_\gamma & c_\alpha c_\beta c_\gamma - s_\alpha s_\gamma \end{pmatrix} = \begin{pmatrix} R_{1,1} & -R_{1,2} & R_{1,0} \\ -R_{2,1} & R_{2,2} & R_{2,0} \\ R_{0,1} & -R_{0,2} & R_{0,0} \end{pmatrix}$$

变换矩阵  $R_{SH}^l$  的计算可以参见附录D3D的实现，实现了前6次的球谐系数的旋转，对于图形渲染来说，已经够用了。

对于高维矩阵的构造方法非常的复杂，采用的是魏格纳d矩阵（Wigner d-matrices），可以参见文献[4]的讨论，网上也有这个算法的高效实现，有兴趣可以研究研究，参见SHTns。

## 附录

根据等式（15）的递归关系，就可以很容易计算出连带勒让德函数[3]。

```
double P(int l, int m, double x)
{
    // evaluate an Associated Legendre Polynomial P(l,m,x) at x
    double pmm = 1.0;
    if(m>0) {
        double somx2 = sqrt((1.0-x)*(1.0+x));
        double fact = 1.0;
        for(int i=1; i<=m; i++) {
```

```

        pmm *= (-fact) * somx2;
        fact += 2.0;
    }
}
if(l==m) return pmm;
double pmmp1 = x * (2.0*m+1.0) * pmm;
if(l==m+1) return pmmp1;
double pll = 0.0;
for(int ll=m+2; ll<=l; ++ll) {
    pll = ( (2.0*ll-1.0)*x*pmmp1-(ll+m-1.0)*pmm ) / (ll-m);
    pmm = pmmp1;
    pmmp1 = pll;
}
return pll;
}

```

根据等式 (20) (21) , 可以计算出球谐函数[3]。

```

double K(int l, int m)
{
    // renormalisation constant for SH function
    double temp = ((2.0*l+1.0)*factorial(l-m)) / (4.0*PI*factorial(m));
    return sqrt(temp);
}

double SH(int l, int m, double theta, double phi)
{
    // return a point sample of a Spherical Harmonic basis function
    // l is the band, range [0..N]
    // m in the range [-l..l]
    // theta in the range [0..Pi]
    // phi in the range [0..2*Pi]
    const double sqrt2 = sqrt(2.0);
    if(m==0) return K(l,0)*P(l,m,cos(theta));
    else if(m>0) return sqrt2*K(l,m)*cos(m*phi)*P(l,m,cos(theta));
    else return sqrt2*K(l,-m)*sin(-m*phi)*P(l,-m,cos(theta));
}

```



```

FLOAT* WINAPI D3DXSHRotate(FLOAT*out, UINT order, const D3DXMATRIX* in,
{
    FLOAT alpha, beta, gamma, sinb, temp[36], temp1[36];

    TRACE("out %p, order %u, matrix %p, in %p\n", out, order, matrix, in);

    out[0] = in[0];

    if ((order > D3DXSH_MAXORDER) || (order < D3DXSH_MINORDER))
        return out;

    if (order <= 3)
    {
        out[1] = matrix->u.m[1][1] * in[1] - matrix->u.m[2][1] * in[0];
        out[2] = -matrix->u.m[1][2] * in[1] + matrix->u.m[2][2] * in[0];
        out[3] = matrix->u.m[1][0] * in[1] - matrix->u.m[2][0] * in[0];

        if (order == 3)
        {
            FLOAT coeff[]={
                matrix->u.m[1][0] * matrix->u.m[0][0], matrix->u.m[1][1] * matrix->u.m[0][0],
                matrix->u.m[1][1] * matrix->u.m[2][1], matrix->u.m[2][1] * matrix->u.m[0][0],
                matrix->u.m[2][0] * matrix->u.m[2][0], matrix->u.m[2][0] * matrix->u.m[0][0],
                matrix->u.m[0][0] * matrix->u.m[2][0], matrix->u.m[0][1] * matrix->u.m[0][1],
                matrix->u.m[0][1] * matrix->u.m[2][1], matrix->u.m[1][1] * matrix->u.m[1][1],
                matrix->u.m[1][1] * matrix->u.m[2][1], matrix->u.m[2][1] * matrix->u.m[2][1],
                matrix->u.m[2][0] * matrix->u.m[2][0], matrix->u.m[2][0] * matrix->u.m[0][0],
                matrix->u.m[0][0] * matrix->u.m[2][0], matrix->u.m[0][1] * matrix->u.m[0][1],
                matrix->u.m[0][1] * matrix->u.m[2][1], matrix->u.m[1][1] * matrix->u.m[1][1],
                matrix->u.m[1][1] * matrix->u.m[2][1], matrix->u.m[2][1] * matrix->u.m[2][1],
            };

            out[4] = (matrix->u.m[1][1] * matrix->u.m[0][0] + matrix->u.m[2][1] * matrix->u.m[0][0]) *
                out[4] -= (matrix->u.m[1][0] * matrix->u.m[2][1] + matrix->u.m[2][0] * matrix->u.m[0][0]) *
                out[4] += 1.7320508076f * matrix->u.m[2][0] * matrix->u.m[0][0];
            out[4] -= (matrix->u.m[0][1] * matrix->u.m[2][0] + matrix->u.m[0][0] * matrix->u.m[2][0]) *
            out[4] += (matrix->u.m[0][0] * matrix->u.m[0][1] - matrix->u.m[0][1] * matrix->u.m[0][0]) *
        }
    }
}

```

```

    out[5] = (matrix->u.m[1][1] * matrix->u.m[2][2] + r
    out[5] -= (matrix->u.m[1][1] * matrix->u.m[0][2] +
    out[5] -= 1.7320508076f * matrix->u.m[2][2] * matr
    out[5] += (matrix->u.m[0][2] * matrix->u.m[2][1] +
    out[5] -= (matrix->u.m[0][1] * matrix->u.m[0][2] -

    out[6] = (matrix->u.m[2][2] * matrix->u.m[2][2] - 6
    out[6] -= (0.5773502692f * (coeff[0] + coeff[1]) -
    out[6] += (0.5773502692f * (coeff[2] + coeff[3]) -
    out[6] += (0.5773502692f * (coeff[6] + coeff[7]) -
    out[6] += (0.2886751347f * (coeff[9] - coeff[8] + (
        (matrix->u.m[1][2] * matrix->u.m[1][2] - mati

    out[7] = (matrix->u.m[0][0] * matrix->u.m[2][2] + r
    out[7] -= (matrix->u.m[1][0] * matrix->u.m[0][2] +
    out[7] += (matrix->u.m[1][0] * matrix->u.m[2][2] +
    out[7] -= 1.7320508076f * matrix->u.m[2][2] * matr
    out[7] -= (matrix->u.m[0][0] * matrix->u.m[0][2] -

    out[8] = 0.5f * (coeff[11] - coeff[8] - coeff[9] +
    out[8] += (coeff[0] - coeff[1]) * in[4];
    out[8] += (coeff[2] - coeff[3]) * in[5];
    out[8] += 0.86602540f * (coeff[4] - coeff[5]) * in
    out[8] += (coeff[7] - coeff[6]) * in[7];
}

    return out;
}

if (fabsf(matrix->u.m[2][2]) != 1.0f)
{
    sinb = sqrtf(1.0f - matrix->u.m[2][2] * matrix->u.m[2][2]
    alpha = atan2f(matrix->u.m[2][1] / sinb, matrix->u.m[2][2]
    beta = atan2f(sinb, matrix->u.m[2][2]);
    gamma = atan2f(matrix->u.m[1][2] / sinb, -matrix->u.m[0][2]
}
else

```

```

{
    alpha = atan2f(matrix->u.m[0][1], matrix->u.m[0][0]);
    beta = 0.0f;
    gamma = 0.0f;
}

```

```

D3DXSHRotateZ(temp, order, gamma, in);
rotate_X(temp1, order, 1.0f, temp);
D3DXSHRotateZ(temp, order, beta, temp1);
rotate_X(temp1, order, -1.0f, temp);
D3DXSHRotateZ(out, order, alpha, temp1);

```

```

return out;

```

```

}

```

```

static void rotate_X(FLOAT *out, UINT order, FLOAT a, FLOAT *in)
{

```

```

    out[0] = in[0];

```

```

    out[1] = a * in[2];
    out[2] = -a * in[1];
    out[3] = in[3];

```

```

    out[4] = a * in[7];
    out[5] = -in[5];
    out[6] = -0.5f * in[6] - 0.8660253882f * in[8];
    out[7] = -a * in[4];
    out[8] = -0.8660253882f * in[6] + 0.5f * in[8];
    out[9] = -a * 0.7905694842f * in[12] + a * 0.6123724580f *

```

```

    out[10] = -in[10];
    out[11] = -a * 0.6123724580f * in[12] - a * 0.7905694842f *
    out[12] = a * 0.7905694842f * in[9] + a * 0.6123724580f *
    out[13] = -0.25f * in[13] - 0.9682458639f * in[15];
    out[14] = -a * 0.6123724580f * in[9] + a * 0.7905694842f *
    out[15] = -0.9682458639f * in[13] + 0.25f * in[15];
    if (order == 4)

```

```
return;
```

```
out[16] = -a * 0.9354143739f * in[21] + a * 0.3535533845f * in[21];
out[17] = -0.75f * in[17] + 0.6614378095f * in[19];
out[18] = -a * 0.3535533845f * in[21] - a * 0.9354143739f * in[21];
out[19] = 0.6614378095f * in[17] + 0.75f * in[19];
out[20] = 0.375f * in[20] + 0.5590170026f * in[22] + 0.7395099998f * in[22];
out[21] = a * 0.9354143739f * in[16] + a * 0.3535533845f * in[16];
out[22] = 0.5590170026f * in[20] + 0.5f * in[22] - 0.6614378095f * in[22];
out[23] = -a * 0.3535533845f * in[16] + a * 0.9354143739f * in[16];
out[24] = 0.7395099998f * in[20] - 0.6614378095f * in[22] + 0.7395099998f * in[22];
if (order == 5)
    return;
```

```
out[25] = a * 0.7015607357f * in[30] - a * 0.6846531630f * in[30];
out[26] = -0.5f * in[26] + 0.8660253882f * in[28];
out[27] = a * 0.5229125023f * in[30] + a * 0.3061861992f * in[30];
out[28] = 0.8660253882f * in[26] + 0.5f * in[28];
out[29] = a * 0.4841229022f * in[30] + a * 0.6614378691f * in[30];
out[30] = -a * 0.7015607357f * in[25] - a * 0.5229125023f * in[25];
out[31] = 0.125f * in[31] + 0.4050463140f * in[33] + 0.9057110548f * in[33];
out[32] = a * 0.6846531630f * in[25] - a * 0.3061861992f * in[25];
out[33] = 0.4050463140f * in[31] + 0.8125f * in[33] - 0.4192627370f * in[33];
out[34] = -a * 0.1976423711f * in[25] + a * 0.7954951525f * in[25];
out[35] = 0.9057110548f * in[31] - 0.4192627370f * in[33] + 0.9057110548f * in[33];
}
```

```
FLOAT * WINAPI D3DXSHRotateZ(FLOAT *out, UINT order, FLOAT angle)
{
```

```
    UINT i, sum = 0;
    FLOAT c[5], s[5];
```

```
    TRACE("out %p, order %u, angle %f, in %p\n", out, order, angle, in);
```

```
    order = min(max(order, D3DXSH_MINORDER), D3DXSH_MAXORDER);
```

```
    out[0] = in[0];
```

```

for (i = 1; i < order; i++)
{
    UINT j;

    c[i - 1] = cosf(i * angle);
    s[i - 1] = sinf(i * angle);
    sum += i * 2;

    out[sum - i] = c[i - 1] * in[sum - i];
    out[sum - i] += s[i - 1] * in[sum + i];
    for (j = i - 1; j > 0; j--)
    {
        out[sum - j] = 0.0f;
        out[sum - j] = c[j - 1] * in[sum - j];
        out[sum - j] += s[j - 1] * in[sum + j];
    }

    if (in == out)
        out[sum] = 0.0f;
    else
        out[sum] = in[sum];

    for (j = 1; j < i; j++)
    {
        out[sum + j] = 0.0f;
        out[sum + j] = -s[j - 1] * in[sum - j];
        out[sum + j] += c[j - 1] * in[sum + j];
    }
    out[sum + i] = -s[i - 1] * in[sum - i];
    out[sum + i] += c[i - 1] * in[sum + i];
}

return out;
}

```

## 参考

[1] 姚端正, 梁家宝. 数学物理方法-第4版..

[2] 顾樵. 数学物理方法.

[3] Robin Green. "Spherical harmonic lighting: The gritty details." Archives of the Game Developers Conference. Vol. 56. 2003.

[4] Joseph Ivanic, and Klaus Ruedenberg. "Rotation matrices for real spherical harmonics. Direct determination by recursion." The Journal of Physical Chemistry 100.15, 6342-6347, 1996.

[5] Wikipeda. Euler angles

[6] Peter-Pike Sloan. "Stupid spherical harmonics (sh) tricks." Game developers conference. Vol. 9. 2008.

[7] Ravi Ramamoorthi, and Pat Hanrahan. "An efficient representation for irradiance environment maps." Proceedings of the 28th annual conference on Computer graphics and interactive techniques. 2001.

编辑于 2020-07-09

数学物理方法

计算机图形学

实时渲染

15 条评论

⇌ 切换为时间排序

写下你的评论...



YivanLee

2020-07-02

终于看到一篇清晰的了，感谢分享！

👍 6



季同

2020-07-31

"Laplacian in Spherical Coordinates"中公式(8)等式右项的" $1/r^2$ "应为" $2/r$ ", 公式(14)中代入时倒是正确的。推了几遍还以为自己推错了\_(:3」 ∠)\_

👍 3



bactlink 回复 季同

02-02

我和你一样🤔，但是那个pdf结果又是对的

👍 赞



季同 回复 bactlink

02-03

对2333

👍 赞



yi wang

2020-07-02

为什么球谐函数的三维图和氢原子轨道那么像?

👍 2



不发nature不改名 回复 yi wang

2020-07-02

因为氢原子的薛定谔方程也是一个带有拉普拉斯算符的微分方程，其轨道解也是球谐函数

👍 3



杨知守 回复 yi wang

2020-07-03

氢原子轨道的角向部分就是球谐函数

👍 2



张晓宇

2020-07-03

以前想推一下拉普拉斯方程的球面表达，进你这链接一看，难怪推不出。。。

👍 1



boy Graphic

2020-07-02

钦佩楼主的数学功底，排版也非常好。球面调谐函数一类对信号进行压缩的算法，函数，非常适合展示一个人的数学背景，发表相关的论文。可惜的是，在工业实践中，似乎都逃脱不了高光之后被抛弃的命运。

👍 1



chopper (作者) 回复 boy Graphic

2020-07-03

没有被抛弃，图形渲染上还是很重要的，

👍 3



darke青 回复 boy Graphic

2020-11-16

目前在游戏的渲染里面，使用球谐保存光照信息，还是蛮常见的

👍 赞



默飒

04-23

讲到了本质，想关注一下大佬github，分享一下链接

👍 赞



HanetakaYuminaga

2020-07-03

最近在学习数学分析，其实“广义傅立叶展开”还有更一般的形式，称为Stone-Weierstrass逼近定理

👍 赞



leinlin

2020-07-02

当时被卡的地方在 拉普拉斯方程为什么能表示光照能量上

👍 赞



龙六一 回复 leinlin

01-05

为什么拉普拉斯方程的解可以表示光照能量啊？

👍 赞