Calculation of Spherical Harmonic Decomposition Coefficients for a Magnetic Field

Xander Naumenko

January 2021

Abstract

To analyze the effects of magnetic field inhomogeneity in neutron electric dipole experiments (nEDM), spherical harmonic decomposition is often used to precisely characterize the magnetic field in the experimental area. This document will provide the method and result of calculating the coefficients in the decomposition in terms of the measured magnetic field partial derivatives.

1 Background

When conducting nEDM measurements, a very accurate understanding of the background magnetic field is required to be able to get an understanding of the systematic uncertainties associated with any results. In particular, understanding to what degree the field is inhomogeneous is of vital importance. To solve this problem, [Abe+19] introduced a method in which spherical harmonic decomposition is used to analyze the fields in different orders. Note that as all work in this note is essentially building on that of [Abe+19], it is expected that at least section 2 is of familiarity to the reader.

The first step towards understanding the magnetic field at a point is to define it as a sum of orthonormal basis polynomials of the form

$$\vec{B}(\vec{r}) = \sum_{l,m} G_{l,m} \begin{pmatrix} \Pi_{x,l,m}(\vec{r}) \\ \Pi_{x,l,m}(\vec{r}) \\ \Pi_{x,l,m}(\vec{r}) \end{pmatrix} \tag{1}$$

where $\Pi_{l,m}$ are harmonic polynomials in the x, y and z directions of degree l and order m, and $G_{l,m}$ are coefficients representing the linear combination of harmonic polynomials. The key behind the construction of these basis functions is that, due to Maxwell's equations, $\nabla \times \vec{B} = \mu_1 j + \frac{1}{c^2}\frac{\partial \vec{E}}{\partial t}$. However, since there is no current in the experimental apparatus and the electric field will be constant, this reduces to $\nabla \times \vec{B} = 0$. This, coupled with the definition of the scalar magnetic potential of $\vec{B} = \nabla\Sigma$, means that the magnetic potential fulfills Laplace's equation, i.e. $\Delta\Sigma = 0$. Using the fact that the harmonic polynomials are a orthonormal basis for equations satisfying Laplace's equation (by definition), we have that

$$\Sigma(\vec{r}) = \sum_{0}^{\infty} \sum_{-l}^{/} l\Sigma_{l,m}(\vec{r}) \tag{2}$$

with $\Sigma_{l,m}$ being our choice of normalization for the harmonic polynomials. This, coupled with 1, tells us that $\Pi_{i,l,m} = \partial_i \Sigma_{l+1,m}$ for $i \in \{x, y, z\}$. In this case, we choose a normalization of

$$\Sigma_{l,m} = C_{l,m}(\phi) r^l P_l^{|m|}(\cos\theta), \tag{3}$$

with P_l^m being the legendre polynomial of degree l and order m and

$$C_{l,m}(\phi) = \frac{(l-1)!(-2)^{|m|}}{(l+|m|)!} \begin{cases} \cos(m\phi) & \text{for } m \geq 0 \\ \sin(|m|\phi) & \text{for } m < 0 \end{cases}. \tag{4}$$

A point of interest is the relationship between the basis coefficients, $G_{l,m}$, and the magnetic field $\vec{B}$. The goal of this note is to find the relationship between these two quantities explicitly.

2 Approach

The approach used to find the results is to use 1 to generated a set of linear equations in $G_{l,m}$ by repeatedly taking different combinations of derivatives of both sides of the equation. The key to this approach is to note that because each $\Pi_{l,m}$ is of degree l, for a given l after taking l partial derivatives all terms with degree lower than l will have turned into constants and disappeared and all terms of degree higher than l will still be dependent on at least one of x, y or z. Since 1 holds for all $x, y, z \in \mathbb{R}$, taking $\vec{r} = 0$ means only terms of degree l remain and can thus be solved in terms of the magnetic field partial derivatives. After the end of this process, given some chosen l chosen, we would have many equations that looks like

$$\partial_{i_1}\partial_{i_2}\ldots\partial_{i_l}\vec{B}(\vec{0}) = \sum_{m=-l-1}^{l+1} G_{l,m}\partial_{i_1}\ldots\partial_{i_l}\vec{\Pi}_{l,m} \tag{5}$$

with each equation corresponding to a choice of $i_0, \ldots, i_l \in \{x, y, z\}$. Since the order one applies derivatives in does not matter, the number of such equations is simply the number of ways of distributing the l coefficients between the choice of x, y, z where order does not matter times the three equations per vector, which is $3 \cdot \binom{l+2}{l} = \frac{3}{2}(l+1)(l+2)$. This is good since it grows faster than the number of variables to solve for, which is always $2l + 3$. However it also means that the system is overdetermined for all $l \geq 1$. However, as mentioned previously $\nabla \times \vec{B} = 0$ which implies $\partial_i B_j = \partial_j B_i$ for all $i \neq j \in \{x, y, z\}$, which makes the equations solvable.

3 Implementation

To implement the above method for solving the G_{lm}s in terms of the magnetic field, Mathematica was used to solve all the equations analytically. For reference the code will be explained in detail in this section. In absolute terms it is not extremely long, but several steps do deserve explanation.

The first thing in the script is to construct the basis functions $\Pi_{l,m}$ as described above. This is accomplished in the following set of lines:

```
c[l_, m_, phi_] = (l-1)!(-2)^Abs[m]/(l+Abs[m])!If[m>=0, Cos[Abs[m]*phi], Sin[Abs[m]*
    phi]];
sigma[l_, m_, r_, phi_, theta_] = c[l, m, phi]*r^l*LegendreP[l, Abs[m], Cos[theta]];
sigmac[l_, m_, x_, y_, z_]=TransformedField["Spherical"->"Cartesian", sigma[l, m, r,
    phi, theta], {r, theta, phi}->{x, y, z}];
pi[l_, m_, x_, y_, z_]={d[sigmac[l+1, m, x, y, z], x], d[sigmac[l+1, m, x, y, z], y],
     d[sigmac[l+1, m, x, y, z], z]};
```

first the coefficients $c_{l,m}$ are generated, after which they are used to generate the magnetic scalar potential basis functions $\sigma_{l,m}$. these are then converted from spherical coordinates to cartesian ones, represented by `sigmac` using mathematica built in functionality. these are then differentiated to produce $\pi_{l,m}$ as required.

next we must construct the set of coefficients to solve for. in mathematica it is not possible to have a list of unnamed variables, so some string manipulation is required:

```
1 maxL=2
2 glm=Table[Table[Symbol["$g"<>ToString@(i-1)<>StringReplace[ToString@(j-i-1), "-"->"n"
    ]],{j,2i+1}],{i,maxL+1}]
3 g[l_, m_]=Indexed[glm, {l+1,m+l+2}]
4 g[l_]=Indexed[glm,l+1]
```

Here `maxL` is simply the highest order of coefficients we are to look at, and the two `g` functions act as indexing functions for the list of lists that is `glm`. The naming convention for the variables is that that have negative m have the minus sign replaced with the letter "n" to make it a valid Mathematica variable name. For example the coefficient of degree 2 and order -1 would be represented by the variable `$g2n1` and could be retrieved by calling the function `g[2, -1]`.

The next step is to implement 1 in code. This is accomplished with the following line:

```
1 polyExpand=Simplify[TrigExpand[{bx[x, y, z], by[x, y, z], bz[x, y, z]}==Sum[Sum[g[l,m
    ]*pi[l, m, x, y, z], {m, -l-1, l+1}], {l, 0, maxL}]], x^2+y^2+z^2>0]
```

This is exactly the equation in 1, coupled with some simplification to get Mathematica to print things properly. Note that of all the steps in this script, this step is the most time intensive by far. Finally, the last step is to set up the set of linear equations in $G_{l,m}$ and solve them. To do so the following is used:

```
1 equations = {polyExpand};
2 Do[
3     Print[Flatten[Normal[Solve[Simplify[equations[[i+1]], {x==0, y==0, z==0}], g[i],
      MaxExtraConditions->Automatic]]]];
4     AppendTo[equations,Flatten[{D[equations[[i+1]], x], D[equations[[i+1]], y], D[
      equations[[i+1]], z]}]]
5 , {i, 0, maxL}]
```

`equations` here is a list of list of equations, indexed by the order l that each set of equations correspond to. The equations corresponding with $l = 0$ is obviously just 1 itself, hence the first line. Then, the program loops over $0 \leq l \leq$`maxL`, each time on the fourth line appending all permutations of the derivatives of the last set of equations. Note that this actually produces many linearly dependent equations, as it reproduces equations in which derivatives are simply taken in different orders. However, it is extremely easy for Mathematica to see this fact in the later solving step, and as this step is not the bottleneck on performance the added simplicity of not having to keep track of which derivatives have been taken far outweighs the minimizing of equations.

Finally, the last step is in line 3 which is to solve the set of equations that were just layed out.

References

[Abe+19] C. Abel et al. "Magnetic-field uniformity in neutron electric-dipole-moment experiments". In: *Physical Review A* 99.4 (Apr. 2019). ISSN: 2469-9934. DOI: 10.1103/physreva.99.042112. URL: http://dx.doi.org/10.1103/PhysRevA.99.042112.