University of Groningen

Maxwellian Interpolation of Magnetic Fields on a Grid

Author:
Jose Luis Rodrigo Ramon
(s3543811)

Supervisors:
prof. dr. ir. C.J.G. Onderwater
prof. dr. ir. P.R. Onck

July 6, 2018

Abstract

In this paper, Maxwellian interpolation of a magnetic field in a cubic grid is performed. Our goal is to interpolate the magnetic field using the values of the three components of the magnetic field B at the grid points. We present a method for implementing the constraints that are implied by Maxwell's equations in fits to measurements of the magnetic field on a regular grid. The method makes use of Harmonic Polynomials solutions of Laplace equation. The fitting problem is reduced to a least squares fit that makes it possible to find solutions with high precision. We illustrate the method by applying it to some magnetic field values simulated according to Maxwell's equations.

Contents

1 Introduction **3**

2 Function Parametrization **6**

3 Constraints **9**

4 Magnets and magnetic fields **10**

5 Harmonic Polynomials **15**
5.1 A basic set of homogeneous harmonic polynomials in 3 variables 16

6 Interpolation via scalar potential ϕ **17**
6.1 One cube interpolation with harmonic polynomials 21
6.2 General Interpolation with harmonic polynomials 22
6.2.1 Interpolation Method Analysis 25
6.2.2 Problems of convergence 32

7 Interpolation using Maxwell Elements **36**
7.1 Maxwell Elements 39
7.2 Fitting Procedure 41
7.3 Harmonic Interpolation with derivatives 42

8 Conclusions **43**

Appendices **47**

A Numerical derivatives **47**

B Code for section 6.1 **50**

C Code for section 7.1 **53**

D Harmonic Interpolation Script **56**

E Maxwell Elements Script **62**

1 Introduction

Particle physics experiments such as LHCb or ATLAS require precise measurements of the outcomes of the $p - p$ collisions in order to extract results that give clues for new physics beyond the standard model, new insights into dark etc. Doing these physics analyses is only possible due to a superbly ultra-fast working detector as well as sophisticated triggering, readout and software reconstruction.

The high energy modern detectors consist of many pieces of equipment which test for different aspects of an event. These components are arranged in such a way that we can obtain the most data about the particles spawned by an event and we can figure out the type of particle based on where that particle appeared in the detector. Detectors are formed by different pieces that measure different properties of the particles. The fundamental parts of a detector are:

- Tracking chamber: Charged particle trajectories can be very accurately determined because the inner region of the detector is filled with highly segmented sensing devices.
- Electromagnetic calorimeter: Used to measure the energy of e^+, e^- and photons thanks to the showers of e^+/e^- pairs produced by these particles.
- Hadron calorimeter: This device measures the total energy of hadrons. Hadrons interact with dense material in that region producing a shower of charged particles. The energy deposit of these charged particles is measured.
- Muon chambers: Only muons and neutrinos reach the outer part of the detector. Muons are detected but as the neutrinos interact weakly they escape. The presence of neutrinos can be inferred by the "missing energy".
- Magnet: The path of a charged particle curves in a magnetic field. The radius of curvature and the direction tell the momentum and the sign of the charge.

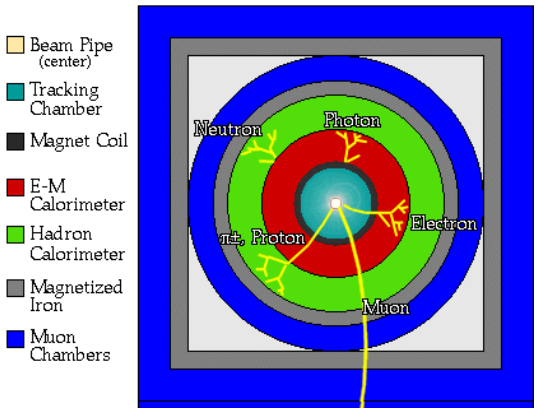

Figure 1: Typical detector scheme with the different sections and their position at the detector respect to the vertex.

The tracking detectors or trackers aim to reconstruct the trajectories of charged particles. One goal in the reconstruction of such trajectories is to estimate the position at which the

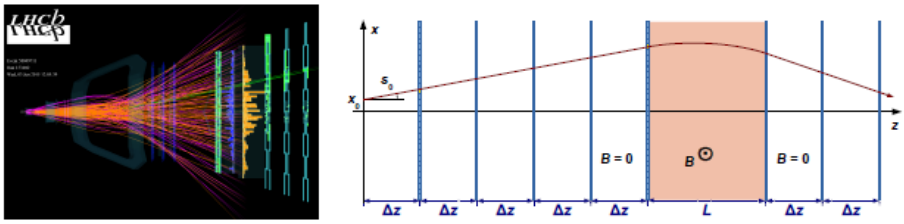

Figure 2: **Left:** Event display from the LHCb experiment at CERN. The picture shows the reconstructed trajectories of charged particles as they transverse the full detector from left to right. The trajectories are bent by the field of the LHCb dipole magnet. Positively and negatively charged particles are bent in opposite directions and low-momentum particles are bent more than high-momentum particles. **Right:** Scheme of the tracking detector setup. The magnet is placed in the pink region, outside that region the magnetic field is zero and the trajectory of the particles is not bent.

particle was created, the second goal is to measure its momentum. The vertex detectors measure the primary interaction vertex and also the secondary vertices from decay particles close to the interaction point using mainly silicon pixel detectors. Closest to the interaction point particle physics experiments use magnets in order to curve the trajectory of the charged particles for particle identification and momentum reconstruction. The magnetic field exerts a Lorentz force on moving charged particle. If the field is homogeneous, the particle is forced onto a circular trajectory with radius of curvature:

$$\rho = \frac{p_T}{qB} \tag{1}$$

where B is the strength of the magnetic field, q is the charge of the particle and p_T is the component of its momentum transverse to the magnetic field lines. Expressing ρ in metres, B in T and q in units of the elementary charge, the transverse momentum in GeV is $p_T \approx 0.3 \cdot e \cdot B \cdot \rho$. If the particle has longitudinal momentum component, it will follow a helix.

The tracking detector setup of a forward spectrometer used mainly in fixed target experiments but also in LHCb is showed in Figure 2. It consists of n detection planes that measure the x coordinate of the trajectory and are spaced at equal intervals Δz along the z axis followed by a magnet of length L and field lines orthogonal to the z and x axes, and another three detector planes spaced at equal intervals Δz. The detection planes are formed by wide cells that give a signal when a particle passes through the cell. In a simplified view the field is constant (homogeneous) inside the magnet and zero outside the magnet so particles follow a straight trajectory before and after the magnet and a circular trajectory inside the magnet.

In the LHCb magnet, which serves as the context of this work the field is indeed far from homogeneous. In such a case the equation of motion must be integrated by tracking the particle through the magnet. A precise knowledge of the magnetic field over that volume is essential for proper simulation and reconstruction of events. Momentum is also directly related with the reconstruction of the invariant mass since $m_0^2 = E - p^2$. The resolution of the invariant mass is related to the momentum and its precision as $\frac{\sigma^2(m_0)}{m_0^2} \sim \frac{p^2\sigma^2(p)}{m_0}$. Tracking detectors in

the magnetic field of the LHCb have to provide momentum measurement for charged particles with a precision of about 0.4% for momenta up to 200 GeV/c. This demands an integrated field of 4 Tm for tracks originating near the primary interaction point.

In a complex detector, the access to the magnetic field is performed hundreds, even thousands of times during the simulation or the reconstruction of a charged particle track. The evaluation of the interpolated function needs to be fast and also it must not occupy a lot of space because it is accessed so many times. For the LHCb magnetic field map a regular grid is used as shown in Figure 3. Measurements using 3D-Hall probes that are moved into different positions inside the region configure this 3D regular grid with B measured values in each of the points of the grid. The B values proportioned by the grid are used to interpolate the magnetic field inside the cubes that conform the grid and create a magnetic field map that is stored and used to track the particles in the reconstruction of the events. LHCb is not the only experiment that uses this approach to create a global map that can be stored and later used when tracking is needed. Other experiments such as OLYMPUS[10] use also a grid to interpolate the magnetic field. In this paper we center our study in regular grids in x ,y and z as the one in LHCb.

Different methods can be used to interpolate these values on a 3D grid. The first approach is the linear interpolation between these grid points but more functions such as Chebysev polynomials[15] or 3D splines[18] can be used as well. However these functions do not obey Maxwell's equations. Inside the LHCb magnet the Maxwell's equations in the absence of current should be verified. The study of how to interpolate the magnetic field such that it obeys Maxwell's equations has been matter of study for so many years[1][5] and continues being improved. Recently a paper has been published [12] (30 May 2018) on the implementation of the Maxwell's equations in the reconstruction of the magnetic field in the g-2 storage ring.The research on this field is necessary and a wide methods are being developed in order to give a correct description of the magnets in detectors.

The aim of this research is to find a set of functions that obey Maxwell equations and that could be used to expand our magnetic field on a regular grid. We look into polynomials in x, y and z as the grid is regular in this system of coordinates and spherical, cylindrical or other kind of symmetries are not expected. Moreover as we need a fast evaluation and a reduction of the complexity of the interpolated function we will use polynomials because they are simplest method to represent a function. The method needs to be precise as the momentum resolution is influenced by the knowledge of the magnetic field and it is directly related with tracking of the particles and invariant mass reconstruction. We will study and quantify the interpolation error of our method. The evaluation of the interpolated function needs to be fast because the magnetic field map is accessed several times during the reconstruction of the events. The creation of the magnetic field map is done before the experiments in the detector start to run and can be computed and stored for later use in particle tracking after the events so it does not really need to be fast.

We have developed a method that can be used to interpolate a magnetic field on a regular grid obeying Maxwell's equations. We have used sets of harmonic polynomials solutions of Laplace's equations. In the first sections we point out how to obtain these sets of functions and which are the advantages of a Maxwellian interpolation, after that the fitting procedure

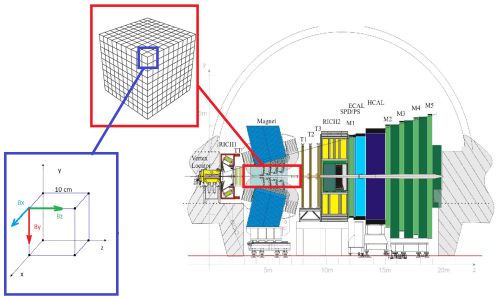

Figure 3: Scheme of the LHCb detector. A regular grid with a grid spacing of 10^{-1} m covering the region enclosed by the red rectangle is where the Hall measurements are done and the regular grid with magnetic field measurements is created covering a volume of 2m $\times$ 1m $\times$ 3.5 m at $\sim$4000 space points

approach is explained and the interpolation error is quantified applying the fit to different generated magnetic field data values. We also give the description of how to do the fitting as well using another sort of method but quite similar which uses the derivatives at the grid points.

2 Function Parametrization

Before centering our study in the harmonic interpolation (Interpolation using a function that obeys Maxwell equations in the absence of current) we are going to discuss some general principles and tricks for evaluating a reliable explicit expression of a function which is given at a number of points. The purpose of this project is how to derive coefficients when the function is known at a number of finite points. We are going to make the discussion of these facts for one variable function although our interpolation is a 3D interpolation (The three magnetic field components) in three variables x, y and z. Let y_j be given at N reference points x_j. We can find intermediate values by linear interpolation. However we might need a large number of reference points to achieve sufficient precision. Another way is to take all the N points into account via Lagrange interpolation formula.

$$y(x) = \sum_{j=1}^{N} \left[\prod_{i \neq j}^{N} \frac{(x - x_i)}{(x_j - x_i)} \right] y_j \tag{2}$$

From there we can see that $y(x)$ is an $(N-1)^{th}$ order polynomial. This interpolation formula can be considered as a weighted average of the function values. This expression is not efficient to be computed repeteadly because we need $N(N-1)$ multiplications. It is better to expand $y(x)$ as

$$y(x) = \sum_{i=0}^{N} a_i x^i \tag{3}$$

where we only need $(N-1)$ multiplications so the evaluation is faster

$$y(x) = a_1 + x(a_2 + x(a_3 + a_4 x + ...)) \tag{4}$$

and we get N linear equations for the N coefficients. This problem can be solved if the matrix we end up with is invertible. If we have more observations N than coefficients M we then usually make a least squares fit.

Different polynomials such as Legendre can be used in (2). However, very often they give a wrong result. One of the reasons is the rounding errors in computers. If we make a least squares fit using many equidistant points (grid points) $N >> M$. The ratio of the largest and smallest element of the $M \times M$ inverted matrix is [20]

$$\frac{\left[(\frac{3M-1}{2})!\right]^2}{\left[M^3\left[\left(\frac{M-1}{2}\right)!\right]^6\right]} \tag{5}$$

This is approximately 10^M for $7 \leq M \leq 15$. Consequently we cannot evaluate more coefficients than there are decimal places in the computer. The use of equidistant reference points and polynomial interpolation causes rounding errors to become so critical. The polynomial

$$y(x) = \prod_{i=1}^{12}(x-i) = x^{12} - 78x^{11} + 2717x^{10} + ... \tag{6}$$

is the only polynomial of degree 12 with 12 equidistant zeros. If we change the term x^{12} to $(1+10^{-6})x^{12}$ some roots change substantially and 4 real roots have disappeared completely. If we go to higher order the effect of rounding errors grows dramatically.

Even in the absence of machine rounding errors the answer could be completely wrong and even get worse the more points we use because of the Runge phenomenon. This phenomenon shows that polynomial interpolation at evenly-spaced points can fail spectacularly to converge. One example of a functions that suffer from this phenomenon is $1/(1+x^2)$.

Why should we use a polynomial model instead of another representation of the function? We want a function that can be fast evaluated and not so complex so a polynomial is the best way to perform that because it involves less number of operations than the evaluation of any other kind of function. Furthermore in principle we do not have symmetries on the grid (it is a 3D regular grid in x, y and z) or in the distribution of the magnetic field in that can be exploited. Moreover, we accept the simplest model to correspond to the reality. The model $y(x) = \sum_{i=1}^{N} a_i x^i$ is more likely to correspond to reality than $y(x) = \sum_{i=10}^{N+10} a_i x^i$. However for example we could think that linear interpolation does not represent a simple model because it represents a function with discontinuity in the first derivative. If we know that the function is continuous we should use a model that is also continuous. Nevertheless, imposing continuity in all higher derivatives is a severe condition and does not give always the most reliable result.

An alternative is to impose the minimization of the radius of curvature of the interpolated function ($\int \frac{y''^2}{(1+y'^2)^{5/2}}$) rather than continuity in all higher derivatives. Then we get a cubic spline fit. Sometimes a spline-model is more reliable than a polynomial interpolation and it is common to use splines to interpolate 3D grids[18].

The use of trigonometric functions can be used as well and it has an interesting feature: we can be sure to get closer to the real function as more and more equidistant points are taken within the same interval. However it has an implicit assumption of periodicity. Trigonometric models have been used to interpolate magnetic fields[13].

We have seen that polynomial interpolation can be very risky for equidistant reference points not as the trigonometric and spline model that guarantee the convergence to the real function. The Chebysev polynomials mentioned before can be very useful because of the orthogonality. The MultiDim function build up in ROOT for the CERN fittings and that has been used to interpolate the magnetic field of the LHCb [15] is based on the use of these sets of polynomials. In a non-orthogonal model to evaluate the value of one coefficient we need to solve all the other coefficients. For an orthogonal model all the coefficients can be evaluated separately. Orthogonality of functions depends on the reference points

$$\sum_n f_i(x_n) f_j(x_n) = \delta_{i,j} \sum_n f_i^2(x_n) \tag{7}$$

An orthogonal model is also a safe way to avoid machine rounding errors. Using an orthogonal model solves in most of the cases the failure of equidistant polynomial interpolation because of machine-rounding errors.

However, our model will be based on a non-orthogonal polynomial interpolation because there is no way of developing an orthogonal model such that it obeys Maxwell Equations [4]. In this paper we will analyze a specific type of interpolation done with Harmonic Polynomials that obey Laplace's equation and we will try to quantify and analyze which problems do we find when trying to interpolate a magnetic field using equidistant points in 3D. We have to take into account all the disadvantages that polynomial interpolation has as we have showed, however as we will see this model presents interesting properties that make the function fitting in some sense profitable.

There is still a problem of how to divide the space in which the function has to be interpolated, in our case the grid dimensions. If the interval of interest is divided into two independent subintervals, the number of coefficients goes up but the computing time needed to evaluate the function is lower since we only use the set of coefficients of one interval at a time. H. Wind presents an empirical law in the CERN Yellow report [4] that states that the interval over which a function is to be defined needs to be divided into so many subintervals that per interval $-\ln E$ coefficients are needed to obtain a relative precision E.

We will take into account all the facts presented here when making our Maxwellian fit. It could seem not so promising the use of non-orthogonal polynomials to fit a function defined in equidistant points since we have pointed so many problems they have, however the polynomials we will use have an important property that is one the main goal of this project: They obey Maxwell's equations because are constructed from Harmonic functions.

As the Maxwell equations introduce more constraints in the fit because they relate the three components of the magnetic field degrees of freedom are reduced and so on less parameters are needed to represent the function. We want a fast evaluation of the interpolated magnetic

field, as the numbers of parameters is reduced less multiplications are needed and less space is required to store the map.

3 Constraints

One of the reasons that makes this approach of Maxwellian Solutions profitable and useful is the reduction of degrees of freedom or constants to be fitted thanks to the constraints introduced by the Maxwell Equations. We can try to find a solution for the Maxwellian interpolation adding the Maxwell constraints afterwards, however this does not decrease the degrees of freedom of the fit. A method using Lagrange multipliers afterwards is showed in the LHCb note 2007-093 [17]. In this model there is a set of P functions used to interpolate and N constraints; the matrix you end up with to invert is $(P + N) \times (P + N)$. This immediately points out the problem of imposing the constraints at the measurements points: if N is big compared to P, the system is overconstrained and singularities appear in the solving.

The alternative approach mentioned there is either to apply the constraints to a limited number of appropriately chosen points or apply a global constraint involving integrals of the Laplacians over the volume. These approaches minimize the Laplacian for the function however the sets of functions are not completely harmonic because the Harmonic condition is added to the estimator of the least squares fit.

We can see that applying Lagrange multipliers we end up with a big matrix $(P+N)(P+N)$. Applying the Maxwell constraints before the fitting makes this matrix smaller such that its dimensions do not depend on the number of points we choose to interpolate, only depend on the maximum degree of the expansion we use i.e. the number of parameters needed set the dimensions of the matrix. Moreover as we will show later this number of parameters needed for the fitting is reduced considerably due to the constraints of Maxwell's equations. With this two arguments we can see why applying the constants before to the model results efficient. The matrix inversion is one of the processes of the fitting in which most of the computation time is used. We mentioned that the creation of the field map does not need to be fast, nevertheless if it is a fast method it allows us to rectify or change the field map if needed easily. If the matrix dimensions are reduced considerably the procedure is faster and this is what happens with a Maxwellian interpolation. However, there are other processes such as matrix evaluation that continue using a lot of computation time.

The method we have used in our study imposes the constraints to the system before making the fitting. We will propose sets of harmonic polynomials that obey Maxwell equations and we will see how the number of parameters needed to develop a function obeying Maxwell's equations is reduced. To represent a magnetic field obeying Maxwell's equations up to the Nth power we need $(N + 2)^2 - 1$ parameters and if we just expand it without the harmonic condition we need $(N + 1)(N + 2)(N + 3)/2$. The number of terms is reduced by a factor $(N+2)/2$. We need to apply at least $(N+2)^2-1$ constraints to this system in order to obtain a solution for the fitting. In our setup we have a magnetic field measured on a grid and in which point of the grid we are given three B measured values (B_x, B_y, B_z). Depending on the number of points we take to make the fit we will be supplied with a number of measured values that will allow us to fit B up to a different maximum power N.

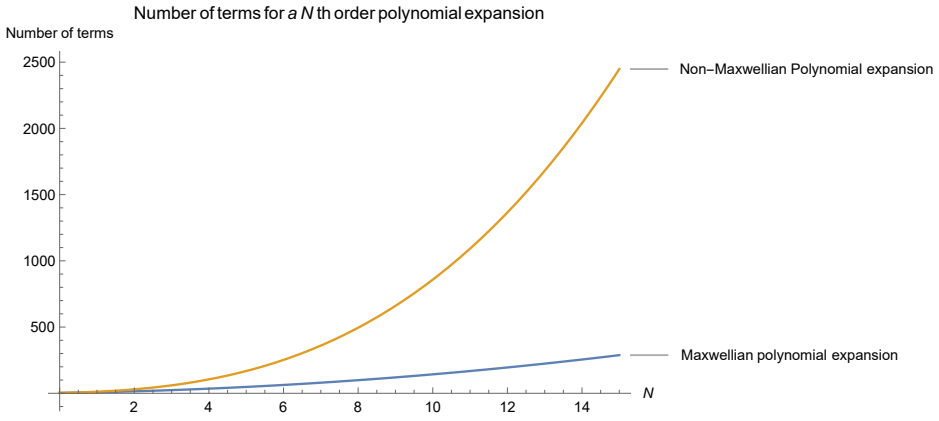

Figure 4: Comparison between the number of terms needed to expand a magnetic field with a polynomial up to the maximum power N obeying Maxwell's equations and the number of terms of a general polynomial expansion of the three B components up to a maximum power N.

In this paper we also present an alternative method in which the derivatives at the corner points are also used and so we have more constraints for a number of points P. We do not use all the derivatives respect to x, y and z of the three components of the magnetic field because as we will show, the Maxwell equations introduce relations between the derivatives and the number of independent derivatives is 5. We show that in the Section 7.

Applying the constraints to the interpolation i.e. finding a Maxwellian solution covers also our other purpose of creating a magnetic field map that requires less storage and that can be evaluated faster because the number of parameters i.e. terms is reduced considerably.

4 Magnets and magnetic fields

To test our procedures we will need magnetic field data that we will generate according to the behavior of typical magnets in accelerators. Magnetostatics in accelerators are conventionally described in terms of multipoles. Multipole fields do provide solutions to Maxwell's equations.

Maxwell's equations may be written in differential form as follows

$$\vec{\nabla}\vec{D} = \rho \tag{8}$$

$$\vec{\nabla}\vec{B} = 0 \tag{9}$$

$$\vec{\nabla} \times \vec{H} = \vec{J} + \frac{\partial \vec{D}}{\partial t} \tag{10}$$

$$\vec{\nabla} \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \tag{11}$$

When the charge density and current density are specified as functions of space and time one can integrate Maxwell's equations to find possible electric and magnetic fields in the system. Usually, however, the solution is not unique: for example, the field within an accelerator

dipole magnet may be modified by propagating an electromagnetic wave through the magnet. By imposing certain constraints (for example, that the field within a magnet are independent of time) it is possible to obtain a unique solution for the fields in a given system of electric charges and currents.

Most realistic situations are sufficiently complicated that solutions to Maxwell's equations cannot be obtained analytically. A variety of computer codes exist to provide solutions numerically when the charges, currents and properties of the materials present are provided. Solving for the fields in realistic three-dimensional systems often requires a reasonable amount of computing power. We do not cover such techniques here, but focus instead on the analytical solutions that may be obtained in idealized situations. Although the solutions in such cases may not be sufficiently accurate they do provide a useful basis for describing fields in real magnets. We will use these idealized situations to test if our fitting procedure is capable of reproducing the fields in an accelerator.

An important feature of Maxwell's equations in materials with constant permittivity and permeability is that the equations are linear in the fields and sources. The principle of superposition applies. If $\vec{B_1}$ and $\vec{B_2}$ are solutions of Maxwell's equations then the field $\vec{B_1} + \vec{B_2}$ will be a solution of Maxwell's equations. It is possible to represent complicated fields as superposition of simpler fields. An important an widely used analysis technique for accelerator magnets is to decompose the fields into a set of multipoles. While it is often the ideal to produce a field consisting of a single multipole component, this is never perfectly achieved in practice: the multipole decomposition indicates the extent to which components rather than the desired multipole are present.

We begin with two-dimensional fields, that is fields that are independent of one coordinate (generally, the coordinate representing the direction of motion of the beam, in our case z direction). We will also introduce variations in the z direction.

Consider a region of space free of charges and currents; for example the interior of an accelerator vacuum chamber. If we further exclude propagating electromagnetic waves, then any magnetic field generated by steady currents outside the vacuum chamber must satisfy

$$\vec{\nabla}\vec{B} = 0 \tag{12}$$

$$\vec{\nabla} \times \vec{B} = 0 \tag{13}$$

These two equations follow from the Maxwell equations given that $\vec{J} = 0$ and derivatives respect to time vanish. These two equations allows to obtain $\vec{B} = -\vec{\nabla}\phi$ which obeys both conditions. Taking the two dimensional case (i.e. constant in the z direction) and solving for polar coordinates.

$$\begin{aligned}\phi =&(E + F\theta)(G + H\log r)+ \\ &\sum_{n=1}^{\infty} J_n r^n \cos n\theta + K_n r^n \sin n\theta + L_n r^{-n} \cos n\theta + M_n r^{-n} \sin n\theta\end{aligned} \tag{14}$$

In practical situations the scalar potential simplifies to

$$\phi = \sum_{n=1}^{\infty} J_n r^n \cos n\theta + K_n r^n \sin n\theta \tag{15}$$

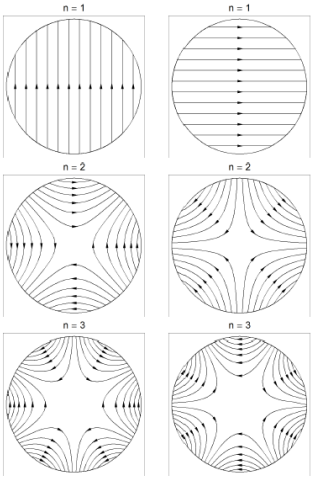

Figure 5: Field lines for "pure" multipole fields. Top: dipole. Middle: quadrupole. Bottom: sextupole. Fields on the left are normal ($J_n = 0$); those on the right are skew ($K_n = 0$). The positive y axis is vertically up; the positive x axis is horizontal and to the right.

Giving components for the magnetic field

$$B_r = -\sum_{n=1}^{\infty} nJ_n r^{n-1} \cos n\theta + nK_n \sin n\theta \tag{16}$$

$$B_\theta = -\sum_{n=1}^{\infty} -nJ_n r^{n-1} \sin n\theta + nK_n \cos n\theta \tag{17}$$

This is an infinite series of cylindrical harmonics that define the allowed distributions of B in 2 dimensions in the absence of currents. Distributions not given by above are not physically realisable. The coefficients J_n and K_n are determined by geometry. Fields obtained from (14) are known as multipole fields. The index n indicates the order of the multipole: $n = 1$ is a dipole field, $n = 2$ is a quadrupole field, $n = 3$ is a sextupole field and so on.

To obtain these equations in Cartesian coordinates, it is enough to know that

$$x = r\cos\theta \quad y = r\sin\theta \tag{18}$$

and express $\cos n\theta$ and $\sin n\theta$ in terms of $\cos\theta$ and $\sin\theta$. We can find the magnetic field components in cartesian coordinates

In Table 1 the 5 lower multipole expansion are shown. $J_n = 0$ represent normal fields and $K_n = 0$ represent skew fields. Applying a general rotation to these solutions field we find another field rotated that is as well a solution valid for a magnetic field obeying Maxwell's equations. We can form a field from the superpositions of the different terms of the multipole expansion and check if our procedure works correctly for them. This results are presented in Section 6.2.1.

Table 1: Multipole expansions in 2D up to order n=5

n	ϕ	B_x	B_y
1	$J_1 x + K_1 y$	J_1	K_1
2	$J_2(x^2 - y^2) + 2K_2 xy$	$2(J_2 x + K_2 y)$	$2(K_2 x - J_2 y)$
3	$J_3(x^3 - 3y^2 x) + K_3(3y^2 - y^3)$	$3(J_3(x^2-y^2)+2K_3 xy)$	$3(K_3(x^2-y^2)-2J_3 xy)$
4	$J_4(y^4 + x^4 - 6x^2y^2) + K_4(4yx^3 - 4xy^3)$	$J_4(4x^3 - 12xy^2) + K_4(12yx^2 - 4y^3)$	$J_4(4y^3 - 12x^2 y) + K_4(4x^3 - 12xy^2)$
5	$J_5(x^5 - 10y^2x^2 + 5y^4 x) + K_5(y^5 + 5yx^4 - 10y^3x^2)+$	$J_5(5x^4 - 30y^2x^2 + 5y^4) + K_5(20yx^3 - 20y^3 x)$	$J_5(-20yx^3+20y^3x)+ K_5(5y^4+5x^4-30y^2x^2)$

In our study we also want to create a dipole in the direction y with small perturbations parametrized as higher order multipoles. We do that because we wish to create a "standard" expansion of the field that we will use to generate supposed realistic data that later we will use to interpolate the magnetic field. We use different approaches and we change some aspects in the interpolation of the data generated by this function to compare how can we improve the fitting. We choose a dipole in the direction y because we know that the LHCb magnet is a 4Tm dipole magnet that ensures particle separation and enables momentum measurements with the xz plane as its main bending plane. B_y is the main field component of the magnet. We know that the field cannot be expressed as simple as we are going to deal with it here.

If we set all the J_n and K_n of the same order we see that higher order terms contribute less because of the lattice of the grid. Let us parametrize a general field expansion that represents a dipole in the direction y with perturbations given by higher multipole terms up to $5th$ power in ϕ. We want the corners of our cubes be given by $\widetilde{x_i} = \frac{x_i}{L}$. We will simplify the grid as an unit cubes grid, so no matter which Lattice are we using our grid points will be given by $\widetilde{x_i} = 0, \pm 1, \pm 2, ...$ Now we parametrize the general expansion field in terms of K_1 L and the coordinates $\widetilde{x_i}$.

$$\begin{aligned} B_x = K_1[&\widetilde{J_1} + L(\widetilde{J_2}x + \widetilde{K_2}y) + L^2(\widetilde{J_3}(x^2 - y^2) + \widetilde{K_3}xy) \\ &+ \widetilde{J_4}L^3(x^3 - 3xy^2) + \widetilde{K_4}L^3(3yx^2 - y^3)+ \\ &\widetilde{J_5}L^4(x^4 - 6y^2x^2 + y^4) + \widetilde{K_5}L^4(4yx^3 - 4y^3x)] \end{aligned} \tag{19}$$

$$\begin{aligned} B_y = K_1[&1 + L(\widetilde{K_2}x - \widetilde{J_2}y) + 3L^2(\widetilde{K_3}(x^2 - y^2) - 2\widetilde{J_3}xy) \\ &+ \widetilde{J_4}L^3(y^3 - 3x^2y) + \widetilde{K_4}L^3(x^3 - 3xy^2)+ \\ &\widetilde{J_5}L^4(-4yx^3 + 4y^3x) + \widetilde{K_5}L^4(y^4 + x^4 - 6y^2x^2) \end{aligned} \tag{20}$$

where x y z in these equations are $\widetilde{x_i}$ and we have changed the parameters of each multipole term dividing all of them by K_1 and also rearranging some constants. We want the coefficients that multiply higher multipole terms to be smaller than K_1 because we have mostly a dipole in the y direction and then perturbations parametrized as higher multipole terms. As far as each of the $\widetilde{J_i}$, $\widetilde{K_i}$ is multiplied by $L^{(i-1)}$ and taking into account that $L < 1$ in the grids

used typically in detectors (in meters as a unit of length) we can see that if we assign to the $\widetilde{J}_i$, $\widetilde{K}_i$ constants values of approximately the same order as K_1 higher order terms will give contributions to lower order of magnitude and so on what we wanted is accomplished. $\widetilde{J}_1$ is set equal to zero. In order to make it easier we will remove the L terms in the expansion (assign value 1) and assign lower order of magnitude than K_1 values to the coefficients $\widetilde{J}_i$ and $\widetilde{K}_i$. Also we will test how the lattice of the grid influences our fitting procedure, in these cases we will reintroduce the L terms.

However the functions we have presented to create magnetic field values can be expanded always exactly because we cut off at a certain power so if we do not introduce higher terms than the order of the expansion in B supposedly the procedure should give always an exact or at least sufficiently accurate result. An expansion of the three components of the magnetic field obeying Maxwell's equations can be represented as:

$$B_x = -B_0 \frac{k_x}{k_y} \sin k_x x \sinh k_y y \sin k_z z \tag{21}$$

$$B_y = B_0 \cos k_x x \cosh k_y y \sin k_z z \tag{22}$$

$$B_z = B_0 \frac{k_z}{k_y} \cos k_x x \sinh k_y y \cos k_z z \tag{23}$$

with $k_y^2 = k_x^2 + k_z^2$. This expression can be used to generate a magnetic field data.

We can also test our procedure generating harmonic functions and then computing their gradients as if it were magnetic fields. Examples of harmonic functions that can be used are for example:

$$\frac{1}{r} \tag{24}$$

$$\frac{x}{r^3} \tag{25}$$

$$-\log(r^2 - z^2) \tag{26}$$

$$-\log(r + z) \tag{27}$$

$$\frac{x}{r^2 - z^2} \tag{28}$$

$$\frac{x}{r(r + z)} \tag{29}$$

As a first approach to quantify the interpolation error of our procedure we will use data values generated with the functions mentioned. However these tests do not provide sufficiently diverse information to make always clear conclusions about our method. On the one hand the multipole expansions are polynomials cut off at some power that in principle are reproducible exactly and on the other hand the gradients obtained from the harmonic functions showed are not analytical in the origin where its values blow up and so on interpolation is conditioned by the nature of these functions.

Let us now discuss which are the Harmonic polynomials that we are going to use in order to make the fit Maxwellian and how can they be obtained.

5 Harmonic Polynomials

As we have said which is our starting point: find a set of functions that will obey Maxwell equations and that could be used to expand our magnetic field inside the grid volume we have to look into which functions can we generate following those equations.

The two equations we have to consider for the magnetic field

$$\vec{\nabla}\vec{B} = 0 \text{ and } \vec{\nabla} \times \vec{B} = 0 \tag{30}$$

can be expressed in an uniquely condition for the potential scalar field ϕ that develops the magnetic field. This constraint is $\Delta\phi = 0$.

We can check easily how the equations in (1) follow from this zero Laplacian potential. The magnetic field is obtained from this scalar potential when these two equations hold because $\vec{\nabla} \times (\vec{\nabla}\phi) = 0$ and from the zero divergence equation $\Delta\phi = 0$.

The research into sets of functions that obey Laplace equation $\Delta\phi = 0$ has been of interest in the history of physics and maths because they can be used to describe accurately the behavior for the magnetic,electrical,gravitational and fluids potential. The solutions to Laplace's equation with boundary constraints are well known in cartesian,cylindrical and spherical coordinates. These sets of functions are well-known and we use them commonly to solve the boundary problems in electrostatics.

For our purpose we have looked into polynomials in x, y and z that accomplish this equation, i.e. such that they have always 0 Laplacian because we have a regular grid in x, y and z and no profitable symmetries are expected for the magnetic field map as we try to do a general method. In this project we have researched into the general solutions of this equation.

E.T. Whitakker gives a general solution for Laplace equation [2] . Let $\phi(x, y, z)$ be any solution (single-valued or many-valued) of the Laplace equation $\Delta\phi = 0$. Let (x_0, y_0, z_0) be some point at which some branch of the function $V(x, y, z)$ is regular. Then if we write:

$$x = x_0 + X, \quad y = y_0 + Y, \quad z = z_0 + Z \tag{31}$$

It follows that for all points situated within a finite domain surrounding the point (x_0, y_0, z_0), this branch of the function can be expanded in an absolutely and uniformly convergent series of the form:

$$\phi = a_0 + a_1X + b_1Y + c_1Z + a_2X^2 + b_2Y^2 + c_2Z^2 + d_2YZ + e_2ZX + f_2XY... \tag{32}$$

For our expansion we have depending on the sum of powers of X, Y, Z a number of terms contributing to the expansion. The terms that appear in the expansion for $n = i + j + k$ with $X^iY^jZ^k$ grows as $\frac{1}{2}(n+1)(n+2)$ where n is the degree of the term (sum of powers of X, Y and Z). For example for $n = 1$ we have three terms that accomplish $i + j + k = 1$: a_1X, b_1Y and c_1Z so the number of terms is three verifying the expression presented above $\frac{1}{2}(1+1)(1+2) = 3$

Now we can substitute this expansion (2) in Laplace's equation

$$\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2} = 0 \tag{33}$$

Equating to zero the coefficients of the various powers of X, Y and Z we obtain an infinite number of relations:

$$a_2 + b_2 + c_2 = 0, \quad \text{etc.} \tag{34}$$

between the constants of the expansion.

There are $\frac{1}{2}n(n-1)$ of these relations between the $\frac{1}{2}(n+1)(n+2)$ coefficients of the terms of any degree n in the expansion of ϕ. From this we see that only $\frac{1}{2}(n+1)(n+2)-\frac{1}{2}n(n-1) = 2n+1$ coefficients of the expansion are really independent. We can deduce from here that the terms of degree n in ϕ must be a linear combination of $(2n+1)$ linearly independent particular solutions of Laplace's equation, which are of degree n in X, Y and Z.

The result obtained tells us that we only need $\sum_{n=0}^{N}(2n+1) = (N+1)^2$ constants or parameters to represent the scalar potential of the magnetic field up to a maximum power N. Moreover, as we know that the magnetic field has no monopoles and also from $\vec{B} = -\vec{\nabla}\phi$ the term a_0 does not provide information to obtain the magnetic field (it has no physical meaning). From there we deduce that only $(N+1)^2 - 1$ parameters are needed to represent a magnetic field B that obeys Maxwell equations up to a maximum power $(N-1)$ (if we choose to develop the potential up to Nth order the magnetic field as it is obtained from the derivatives of ϕ will have a maximum power $N-1$).

We need a set that gives these polynomials for each degree n explicitly and that we can compute and use easily for our interpolation. The polynomials given by E. P. Miles and Ernest Williams for $k = 3$ are given explicitly for each n [3]. E.P. Miles and Ernest Williams found a basic set of homogeneous harmonic polynomials of degree n in k variables with a recursive formula that points how to obtain them and that we have used in our method to interpolate the magnetic field. We explain how these sets of $2n+1$ polynomials are obtained and their properties in the next section.

5.1 A basic set of homogeneous harmonic polynomials in 3 variables

Here we present how to obtain the sets of homogeneous harmonic polynomials of degree n in 3 variables and the derivation of the vanishing Laplacian for these sets. In the original paper of E.P Miles and Ernest Williams a set of homogeneous harmonic polynomials of degree n in k variables is presented. We have restricted here our interest to $k = 3$.

For any set of non-negative integers (b_j) such that $b_1 \leq 1$ and $\sum_{j=1}^{3} b_j = n$, let:

$$(H^n)_{b_1,b_2,b_3}(x_1, x_2, x_3) = \sum (-1)^{[a_1/2]} \frac{n!}{\prod_{j=1}^{3} a_j!} \frac{[\frac{a_1}{2}]!}{\prod_{j=2}^{3} (\frac{b_j - a_j}{2})!} \prod_{j=1}^{3} (x_j)^{a_j} \tag{35}$$

where the summation is extended over all (a_j) such that:

1. $a_j = b_j \bmod 2$, $j = 1, 2, 3$
2. $\sum_{j=1}^{3} a_j = n$
3. $a_j \leq b_j$, $j = 2, 3$.

The polynomials presented in (5) are linearly independent since each contains exactly one different nonvanishing term of the monomials $x_1^{a_1}x_2^{a_2}x_3^{a_3}$, $\sum_{j=1}^3 a_j = n$, $a_1 \leq 1$. Moreover, since the number of terms in $(\sum_{j=1}^3 x_j)^n$ is:

$$\binom{n+3-1}{3-1} \tag{36}$$

the total number of monomials of type $x_2^{a_2}x_3^{a_3}$, $\sum_{j=2}^3 a_j = n$, and of type $x_1{x_2}^{a_2}{x_3}^{a_3}$, $\sum_{j=2}^3 a_j = n-1$, is

$$\binom{n+3-2}{3-2} + \binom{n+3-3}{3-2} = 2n+1 \tag{37}$$

Thus the polynomials (5) are

$$2n+1 \tag{38}$$

in number.

Let us also prove that these polynomials are in fact harmonic, i.e. if we apply the Laplacian we obtain 0 as a result for all of them. Let c_j, $j = 1, 2, 3$, be such that

- $c_j = b \bmod 2$,
- $c_j \leq b_j$ $j = 2, 3$, and
- $\sum_{j=1}^3 c_j = n-2$

The coefficient B_{c_1,c_2,c_3} of $\prod_{j=1}^3 (x_j)^{c_j}$ in $\nabla^2 (H^n)_{b_1,b_2,b_3}(x_1, x_2, x_3)$ is given by

$$\begin{aligned}
&B_{c_1,c_2,c_3=(-1)^{[c_1/2]+1}} \frac{n!}{\prod_{j=1}^3 c_j!} \left[\frac{[\frac{c_1}{2}+1]!}{\prod_{j=1}^3 (\frac{b_j-c_j}{2})!} - \frac{\sum_{j=2}^3 [\frac{c_1}{2}]!(\frac{b_j-c_j}{2})}{\prod_{j=2}^3 (\frac{b_j-c_j}{2})!} \right] = \\
&= (-1)^{[c_1/2]+1} \frac{n! [\frac{c_1}{2}]!}{\prod_{j=1}^3 c_j! \prod_{j=2}^3 (\frac{b_j-c_j}{2})!} \left[[\frac{c_1}{2}] + 1 - \frac{1}{2}(n-b_1) + \frac{1}{2}(n-2-c_1) \right] = \\
&= (-1)^{[c_1/2]+1} \frac{n! [\frac{c_1}{2}]!}{\prod_{j=1}^3 c_j! \prod_{j=2}^3 (\frac{b_j-c_j}{2})!} \left[[\frac{c_1}{2}] - \frac{1}{2}(c_1-b_1) \right] = 0 \quad \text{q.e.d.}
\end{aligned} \tag{39}$$

As showed in the above equation these polynomials in fact solve the Laplacian equation $\Delta\phi = 0$. We have a formula (35) that points out how to calculate these sets of polynomials so we can expand the scalar potential of the magnetic field using them. A similar approach was presented by H. Wind [1]. In the next section we explain how these polynomials that form a basis for ϕ can be used to reconstruct the magnetic field.

6 Interpolation via scalar potential ϕ

We have showed in the previous section sets of polynomials that obey Laplace's equation and that can be used to expand the scalar potential ϕ. Now we are going to explain which procedures can we implement using these equations to interpolate the magnetic field on a grid.

The first step is to set how many points we are going to use for the interpolation and the degree of the polynomial expansion. To expand the potential to a power N we need $(N+1)^2-1$ parameters and so on X measured values are needed being $X \geq (N+1)^2-1$. If we choose a big number of points compared with N the system is overconstrained and the fitting procedure will fail or be too slow because a lot of evaluations are needed.

We also have to take into account how the magnetic field varies in one region. Regions where the gradients of B are big we should use less points and will require more terms in the parametrization than the ones where the field variations are smoother. I want to emphasize the importance of knowing how the divergence of B varies and its values for a good fitting. As we will show in one of the next sections the divergence values of B can be used to perform the fit as well. Calculating the derivatives from the magnetic field values on the grid or implementing a way of measuring them in the corners of the grid would provide information for the fitting and also as we will show later the derivatives can be used as more constraints to make the fit.

In a real B interpolation as the one in the LHCb several regions of parametrization are defined in which different procedures or different number of points are used to make the interpolation. As we are not making our study for a given B map we will give different approaches using different number of terms and different points of the grid.

We are going to use the Harmonic polynomials to expand the scalar potential and then derive the magnetic field from it. We show in Table 2 the harmonic polynomials up to 5th power. In the Table we have also showed the cumulative number of harmonic polynomials up to an order n as well as the number of elements for each order. The number of parameters needed to expand an harmonic function up to a power N can be visualized from Table 2 for the first 5 powers. As each parameter is introduced with one harmonic polynomial, the number of polynomials used to expand the function is equal to the number of parameters needed to expand this function. The polynomial corresponding to the zeroth order has not been taken into account because it will represent a monopole that we know does not exist for the magnetic field.

Visualizing this table we know how many points do we need at least to fit the function up to a Nth order polynomial. For example the 8 corners of one cube$\times$3 values in each corner= 24 constraints allow us to fit a $4th$ degree polynomial in ϕ and hence third order in $\vec{B}$.

Using the 8 corners of the cube to interpolate the magnetic field we can only fit a third order polynomial in B and so on higher contributions than the octupole cannot be parametrized. In order to increase the precision a higher order polynomial in ϕ can be constructed and more grid points have to be used in order to solve the fit. Moreover, as we will show later it is not possible to solve the system of 24 equations we end up with when trying to fit a 4th order polynomial in ϕ with 24 B values of 8 corner points corresponding to one cube. Fitting the scalar potential to a 3rd order polynomial with the eight corners of one cube means a quadratic interpolation in B. As we want to increase the precision of the fitting and we want to take into account the contributions of octupoles, decapoles ... we will fit a higher order

Table 2: Harmonic Polynomials

Order	**Elements**	**Cumulative**	**Polynomials**
0	1	unphysical	$[1]$
1	3	3	$[x, y, z]$
2	5	8	$[xy, xz, yz, x^2 - z^2, y^2 - z^2]$
3	7	15	$[xyz, x^3 - 3xz^2, xy^2 - xz^2, y^3 - 3yz^2, x^2y - yz^2, 3x^2z - z^3, 3y^2z - z^3]$
4	9	24	$[x^3y - 3xyz^2, xy^3 - 3xyz^2, x^3z - xz^3, 3xy^2z - xz^3, 3x^2yz - yz^3, y^3z - yz^3, x^4 - 6x^2z^2 + z^4, y^4 - 6y^2z^2 + z^4, 3x^2y^2 - 3x^2z^2 - 3y^2z^2 + z^4]$
5	11	35	$[x^3yz - xyz^3, xy^3z - xyz^3, xy^4 - 6xy^2z^2 + xz^4, x^3y^2 - x^3z^2 - 3xy^2z^2 + xz^4, x^5 - 10x^3z^2 + 5xz^4, x^4y - 6x^2yz^2 + yz^4, x^2y^3 - 3x^2yz^2 - y^3z^2 + yz^4, y^5 - 10y^3z^2 + 5yz^4, 5x^4z - 10x^2z^3 + z^5, 5y^4z - 10y^2z^3 + z^5, 15x^2y^2z - 5x^2z^3 - 5y^2z^3 + z^5]$

polynomial using more grid points.

When trying to interpolate the magnetic field on a point $\vec{r}$ the first option is to use the corner points of the cube where this point is contained. Another option is to use the grid point closest to the evaluation point as the central point of a 3D cross. In this case we use 7 grid points and there are always 4 points within a distance of one lattice spacing, and 3 at a distance between 1 and 2 lattice spacing. If we need more points because we use a higher degree polynomial we can use more grid points. However we should think about how do we arrange this grid points around our interpolation point $\vec{r}$.

For example it makes no sense taking consecutively cubes in one direction for example as showed in the Figure 6. It seems to be better for the fit if we take cubes surrounding the cube where our interpolation point is. We present here two ways of arranging 7 and 8 cubes respectively around a point.

We can chose to attach to each of the faces of the cube where the interpolation point

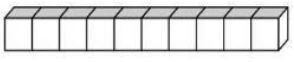

Figure 6:

is contained another cube. The result is a cross formed by a central cube (where the interpolation point is) and 6 more cubes. These three dimensional crosses have an interesting property, they fill in completely the 3D space. With this arrangement we are provided with 28

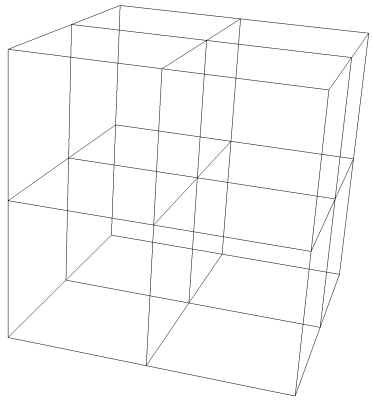

Figure 7: Setup of eight cubes (27 points) used to interpolate the magnetic field of a point close to the center point of this $2L$ cube. When using this arrangement we place the origin of coordinates in the center of this $2L$ cube where L is the lattice of the regular grid

grid points that correspond to 84 values for the fitting procedure. The number of measured values allows us to develop ϕ up to a $8th$ order polynomial. The volume occupied by the cubes is $7L^3$ where L is the lattice constant of the grid. We will use this arrangement to interpolate magnetic field data. As it will be mentioned many times in the next sections we will call this cubes arrangement "Cubes Cross". For simplicity we want the corners of the central cube to be labeled by $\widetilde{x_i} = \pm 1$. So the grid points of this arrangement are positioned at coordinate points $\widetilde{x_i} = \pm 1, \pm 3$. In this case the cubes are not unit cubes but because we prefer to use coordinates that take values $\pm 1, \pm 3$ rather than $\pm 1/2, \pm 3/2$. When we want to transform the expressions of the magnetic fields like equation (18) we will substitute all the L terms by $L/2$ for this arrangement of cubes.

Another option is to use 8 cubes in the following way. Set the cube where our interpolation point is contained. Then find which corner of this cube is closest to the interpolation point. After that add 8 cubes surrounding this corner i.e. the chosen corner is the center of a $2L$ cube. With this construction we are given 27 grid points i.e. 81 values for the fitting procedure that allow us to develop ϕ up to an $7th$ degree polynomial. The volume occupied by the cubes used is now $8L^3$. We will refer to that arrangement of cubes used to interpolate as "Cubes 8". The coordinates of the grid points in this arrangement correspond to values $\widetilde{x_i} = \pm 1, 0$ where the central grid point is the origin of coordinates.

In principle we are thinking about making this construction for the interpolation of one point $\vec{r}$, however in the experiments where these types of grids are used to interpolate the magnetic field we want to create a global map of the magnetic field that can be stored and later used to reconstruct the particles trajectory. We do not make a construction about a point where we want to interpolate and find the magnetic field there with this construction and so on the next point etc... we create a global map subdivided in regions in which we interpolate with one method and then store that interpolated B to be used in all the volume covered by that region so the discrepancies between regions are reduced.

Nevertheless, as we have mentioned above the regions in which the grid is subdivided

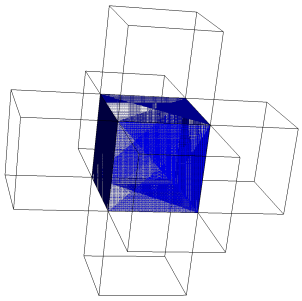

Figure 8: Setup of seven cubes (28 points) used to interpolate the magnetic field of a point contained in the blue cube. When using this arrangement we place the origin of coordinates in the center of the blue cube. The points of the central cube are labelled as $x = \pm 1$, $y = \pm 1$ and $z = \pm 1$

are chosen according to the gradient values of B. We will use these constructions in order to fit the magnetic field to a higher degree polynomial because we have a bigger number of constraints.

Note that the dimensions of the matrix to be inverted are not influenced by the number of grid points we take as we will show later. The dimensions of the matrix are set by the number of parameters used to make the fit. We could think that taking more points for the fitting does not influence the computation time because the matrix to be inverted has the same dimensions if we do not change the maximum order of the polynomial, however taking a bigger number of points makes the evaluation of this matrix slower and so on influences. Moreover taking a lot of points can overconstrain the fitting and give a completely wrong result. Besides, as more points are used more operations are done and can be appreciated how machine rounding errors influence the fit. As a final remark choosing a big number of points for the fitting means that we subdivide the grid in bigger regions and this is not always better because of the gradients of B. On the other hand if we subdivide it in a lot of small regions the storage requirements increase because the storage required goes like the cube of the inverse grid spacing.

In the next sections we present different arrangements to interpolate the magnetic field via scalar potential using different number of grid points and changing the maximum order of the expansion.

6.1 One cube interpolation with harmonic polynomials

In this section we describe the method we have tried to interpolate the magnetic field using only one cube i.e. 8 points $\times$ 3 B components=24 known values for the fitting procedure. We are using the corners of the cube the evaluation point is in. For simplicity we are going to make it explicitly for a standard cube with the center at the origin of coordinates and the corners with $x = \pm 1$, $y = \pm 1$ and $z = \pm 1$.

As we have shown before the number of parameters needed to expand the scalar potential up to a 4th power are 24. We have a system of the same number of parameters and constraints. The solution (if there is one) will be a function that exactly reproduces the B values at the 8 corner points. In principle this procedure does not seem promising because in the real interpolation of a magnetic field we have errors for the values and an exact solution that reproduces exactly the B measured values would not be a good interpolation. Moreover most of the interpolation procedures are based on the minimization of the residuals at the points where we have measurements. To increase the precision a number of measured values greater than the number of parameters is used to fit the function. However, as it is a simple method in which we only have to solve a linear system of 24 equations we will try to solve it.

The method is simple but it has no solution because the matrix of the system we end up with is not invertible. We have implemented a script in Wolfram Mathematica that shows this problem. First of all we expand the scalar potential using the harmonic polynomials and then we calculate its gradient that corresponds to the three components of the magnetic field. We have 24 values 8 for each of the three components of B and we have also the expansions of the magnetic field in terms of the parameters that multiply the harmonic polynomials. We substitute these expansions in the eight corners and we set that equal to each of the B measured values. The result is a system of 24 equations that relates the B values with the parameters of the expansion.

Let us denote $\vec{\beta}$ a vector formed by the B measured values as follows:

$$\vec{\beta} = [B_x(\vec{r_1}), B_y(\vec{r_1}), B_z(\vec{r_1}), ..., B_x(\vec{r_n}), B_y(\vec{r_n}), B_z(\vec{r_n})] \tag{40}$$

where $n = 8$ because we use 8 grid points.

There is a matrix M that connects this vector $\vec{\beta}$ with the vector $\vec{a}$ that contains the 24 parameters of the expansion as follows.

$$\vec{\beta} = M\vec{a} \tag{41}$$

To solve the system we just need to invert that matrix and then we will have the values of the parameters contained in $\vec{a}$ as a function of the B values in $\vec{\beta}$. Let us show the procedure we have implemented in Wolfram Mathematica in which we show that the matrix of this system M is not invertible. The code generated to test that and that shows that the matrix M is not invertible is presented in Appendix A.

This procedure cannot be used to interpolate the magnetic field in a cube using harmonic polynomials. If we want to do an interpolation using only the 8 corners of the cube we will have to expand the potential maximum to a 3rd degree polynomial. If we want to make it possible to fit a 4th order polynomial in B we will have to use more grid points. Let us look into other approaches that we can use with the harmonic polynomials to fit at least a 4th order polynomial in B.

6.2 General Interpolation with harmonic polynomials

In this section we describe a method that can be used to interpolate the magnetic field using an arbitrary number of grid points with harmonic polynomials via scalar potential. In this

case, the method has been implemented in Wolfram Mathematica in such a way that we can change the number of grid points used to interpolate and we can modify the maximum power for the potential ϕ.

First of all we are going to show how the fitting procedure is performed. We use the harmonic polynomials to expand the scalar potential ϕ. For our purposes the program has been tested in the two arrangements of grid points mentioned in the previous section (Cubes 8 and Cubes Cross) centered in the point $\vec{r} = (0, 0, 0)$ because they allow us to fit a 4th order polynomial in B. As these constructions are centered in the origin of coordinates for our test we expand the potential around the point $(0, 0, 0)$. The program can be modified easily giving a different arrangement of points not centered in the origin of coordinates. We explain here the general method that can be used to fit a Nth order polynomial in B using P points such that $P \times 3 \geq (N + 1)^2 - 1$.

The data we are provided with consists of a set of points

$$P = [\vec{r_1}, \vec{r_2}, ..., \vec{r_n}] \tag{42}$$

and the B measured values on these points.

$$\beta = [\vec{B}(\vec{r_1}), ..., \vec{B}(\vec{r_n})] = [(B_x(\vec{r_1}), B_y(\vec{r_1}), B_z(\vec{r_1})), ..., (B_x(\vec{r_n}), B_y(\vec{r_n}), B_z(\vec{r_n}))] \tag{43}$$

We index the three values of B by B_l so the matrix β is formed by the n vectors containing the B values. First of all we expand the scalar potential ϕ as a sum of the Harmonic polynomials.

$$\begin{aligned}
&\phi = a_0 + xa_1 + ya_2 + za_3 + xya_4 + xza_5 + yza_6 + (x^2 - z^2)a_7 + \\
&(y^2 - z^2)a_8 + xyza_9 + (x^3 - 3xz^2)a_{10} + (xy^2 - xz^2)a_{11} + (y^3 - 3yz^2)a_{12} + \\
&(x^2y - yz^2)a_{13} + (3x^2z - z^3)a_{14} + (3y^2z - z^3)a_{15} + (x^3y - 3xyz^2)a_{16} + \\
&(xy^3 - 3xyz^2)a_{17} + (x^3z - xz^3)a_{18} + (3xy^2z - xz^3)a_{19} + (3x^2yz - yz^3)a_{20} \\
&+ (y^3z - yz^3)a_{21} + (x^4 - 6x^2z^2 + z^4)a_{22} + (y^4 - 6y^2z^2 + z^4)a_{23} + \\
&(3x^2y^2 - 3x^2z^2 - 3y^2z^2 + z^4)a_{24} + ...
\end{aligned} \tag{44}$$

The term a_0 as is the term of the monopole will be removed from the expansion and then we have $M = (N + 1)^2 - 1$ parameters. After this expansion of ϕ as the sum of harmonic polynomials and knowing that $\vec{B} = -\vec{\nabla}\phi$ we construct $(N + 1)^2 - 1$ vector functions that contain the gradient of the harmonic polynomials n used in the expansion. We denote these quantities as $B^m{}_l$ with $m = 1, ..., (N + 1)^2 - 1$ and $l = 1, ..., 3$.

Let us now represent the data $B_l(\vec{r})$ at a generic field point $\vec{r}$ as a sum over the $M = (N+1)^2-1$ vector functions $B^m{}_l$ as $B_l(\vec{r}) = \sum_{m=1}^{M} a_m B^m{}_l(\vec{r})$ where the a_m are the coefficients of the expansion of the scalar potential.

We find the coefficients a_m solving a standard linear least squares problem. There are two reasons for minimizing the sum of squares rather than, for example, the sum of the fourth powers. The first reason is that, if there is a random error of measurement with a normal distribution, the least squares solution is the most likely one [21]. The other reason is for convenience; making the fit for any other power would be much more difficult.

There can be various purposes for making a least squares fit besides the smoothing out of measurement errors. We may want a function representation with as few coefficients as possible for a given precision. It is also possible that the least squares solution is more reliable for interpolation than the exact fit even in the absence of measurement errors. As we have showed in the previous section the exact fit for one cube is not possible because the matrix we end up with is not invertible due to Maxwell Equations.

The fit is performed minimizing the estimator:

$$\chi^2 = \sum_{l=1}^{L}\sum_{n=1}^{N}\left[\frac{B_l(\vec{r_n}) - \sum_{m=1}^{M} a_m B_l^m(\vec{r_n})}{\sigma_{l,n}}\right]^2 \tag{45}$$

with $L = 3$, $M = (N+1)^2 - 1$ and the values $\sigma_{n,l}$ are the measurement errors (standard deviation) associated to the values $B_l(\vec{r_n})$. If we are not provided with these errors we just set all the $\sigma_{n,l}$ equal to 1 and the process mentioned here continues being valid. To find the least squares solution, we differentiate this with respect to the a_m coefficients and set the result to zero, obtaining a Matrix problem. We are going to show how this problem is solved explicitly [8].

If we differentiate respect to the a_m's and set that equal to zero we get M equations

$$0 = \sum_{l=1}^{L}\sum_{n=1}^{N}\frac{1}{{\sigma_{n,l}}^2}\left[B_l(\vec{r_n}) - \sum_{m=1}^{M} a_m B_l^m(\vec{r_n})\right]B_l^k(\vec{r_n}) \tag{46}$$

with $k = 1, ..., M$. Interchanging the order of summations, we can write it as the matrix equation

$$\sum_{m=1}^{M}\alpha_{k,m}a_m = \gamma_k \tag{47}$$

where α is an $M \times M$ matrix

$$\alpha_{k,m} = \sum_{l=1}^{L}\sum_{n=1}^{N}\frac{B_l^m(\vec{r_n})B_l^k(\vec{r_n})}{{\sigma_{n,l}}^2} \tag{48}$$

and γ a vector of length M

$$\gamma_k = \sum_{l=1}^{L}\sum_{n=1}^{N}\frac{B_l(\vec{r_n})B_l^m(\vec{r_n})}{{\sigma_{n,l}}^2} \tag{49}$$

Those are the normal equations of the least-squares problem. The inverse matrix $C = \alpha^{-1}$ is called the covariance matrix because it is closely related to the standard uncertainties of the estimated parameters a_m.

The variance associated with the estimate a_m by propagation of errors is

$$\sigma^2(a_m) = \sum_{n=1}^{N}\sum_{l=1}^{L}{\sigma_{n,l}}^2\left(\frac{\partial a_m}{\partial B_l(\vec{r_n})}\right)^2 \tag{50}$$

From (18) we can write

$$a_m = \sum_{k=1}^{M} {\alpha_{m,k}}^{-1} \gamma_k = \sum_{k=1}^{M} C_{m,k} \left[\sum_{n=1}^{N} \sum_{l=1}^{L} \frac{B_l(\vec{r_n}) B_l^m(\vec{r_n})}{{\sigma_{n,l}}^2} \right] \tag{51}$$

and then derive

$$\frac{\partial a_m}{B_l(\vec{r}_n)} = \sum_{l=1}^{L} \sum_{m=1}^{M} C_{m,k} \frac{B_l^m(\vec{r_n})}{{\sigma_{n,l}}^2} \tag{52}$$

The result we get for the standard deviations is

$$\sigma^2(a_m) = \sum_{i=1}^{M} \sum_{k=1}^{M} C_{m,k} C_{m,i} \left[\sum_{l=1}^{L} \sum_{n=1}^{N} \frac{B_l^m(\vec{r_n}) B_l^k(\vec{r_n})}{{\sigma_{n,l}}^2} \right] \tag{53}$$

The term in brackets in this equation is the matrix α so the equation (24) is reduced to

$$\sigma^2(a_m) = C_{m,m} \tag{54}$$

The diagonal elements of C are the variances (squared uncertainties) of the fitted parameters. The offdiagonal elements $C_{m,k}$ are the covariances between a_m a_k.

The least squares method reduces the fitting to a matrix inversion of α. The parameters can be extracted from equation (22). The problem is reduced to an inversion of a matrix $M \times M$ where M is the number of parameters used to make the expansion of the potential using the Harmonic Polynomials. This procedure can be used to find the magnetic field B up to a power $N-1$ with an arbitrary number of grid points P such that $P \times 3 \geq (N+1)^2 - 1$. If we have a big number of grid points the procedure can be fitted to a bigger power in B.

We have developed a program using Wolfram Mathematica that realizes the interpolation of the magnetic field on a grid using harmonic polynomials. The script is shown in Appendix C. The Harmonic polynomials are constructed with the function developed by Alexander Urintsev for Wolfram Mathematica[22]. that calculate these sets of homogeneous harmonic polynomials.

In order to quantify the interpolation error different tests have been performed using different data values according to the functions mentioned in section 4. We show that in the next section.

6.2.1 Interpolation Method Analysis

In this section different tests that have been made to know the accuracy of our procedure are presented. The tests have been done mostly for two different number of points in the two different configurations mentioned in Section 6. We have restricted the maximum power expansion of the scalar potential to 5 for speed computation reasons and because we do not expect higher terms to contribute significantly to the Magnetic field.

In order to quantify the interpolation error of our procedure we calculate the residuals at

the grid points and we calculate for each of the three components in each of the n grid points used the relative error as:

$$E(B_i(\vec{r}_n))(\%) = \frac{|B_{measured,i}(\vec{r}_n) - B_{interpolated,i}(\vec{r}_n)|}{|B_{measured,i}(\vec{r}_n)|} 100(\%) \tag{55}$$

where $i = x, y, z$ and n indexes the points of the grid used in the interpolation. For the tests we have created magnetic field values and we have stored these values on a grid of lattice constant $L = 1$. Then we have applied the interpolation procedure to these B values and calculated the interpolation error. The different magnetic fields generated to test the procedure have been created according to the magnetic functions mentioned in Section 4.

The first test we have created evaluates how the interpolation error varies if we take more points to compute the magnetic field. For this configuration we maintain fixed the power expansion $N = 5$ and we use gradually more points to compute the interpolation. This initial test will show us if increasing the number of points used to interpolate increases the precision. For this procedure we have taken consecutively more points around a central cube which center is the origin of coordinates. The magnetic field used to generate the data values that are stored on the grid and later used to interpolate has been created using a polynomial expansion of the scalar field up to a $5th$ power with the harmonic homogeneous polynomials and also with the magnetic field expansion mentioned in Section 4 as a sum of different multipoles added to a strong dipole in the direction y.

For each of the interpolations done with a different number of points we have calculated the relative errors of each component according to equation (56). Then we have calculated what we have called the average relative error as the sum of all the relative errors of the three components over the N points used divided by the number of points$\times 3$ used to interpolate. Figures 9 and 10 show how the error fluctuates when changing the number of points interpolating a magnetic field generated with an harmonic expansion of phi to the 5th power.

The fluctuations in the error are of the order 10^{-13}. Nevertheless we see that our relative errors are of the order $\sim 10^{-13} - 10^{-12}(\%)$ and so the interpolation procedure reproduces correctly the function despite the number of points used because the machine precision of Mathematica is $\sim 10^{-15}$. The behavior of the error obtained is due to the number of operations involved that gives rise to machine rounding errors. Moreover it can be observed that as more points are taken (more operations involved) the error starts to increase slightly and randomly. We recall here a result presented in Section 2 equation (4) that points out the problem with rounding errors in computers when many equidistant points $N >> M$ are taken. However it can be seen that the number of points taken does not influence clearly the behavior of the error for the functions used and the fact mentioned in (4) does not give rise to a wrong result here. As the interpolated function is a polynomial of the same order as the interpolating function this result could be expected. However we could have end up with completely wrong results because of the reasons mentioned in section 2. We have made the calculation up to ~ 1000 points and in our procedure we suggested to use one of the two configurations of cubes in which we use $28 - 27$ grid points. If we want to parametrize the magnetic field with a big number of terms then we need a lot of grid points and this will be useful. Taking more points and increasing the order of the polynomial used to interpolate will increase the precision in principle. Nevertheless the convergence should be calculated

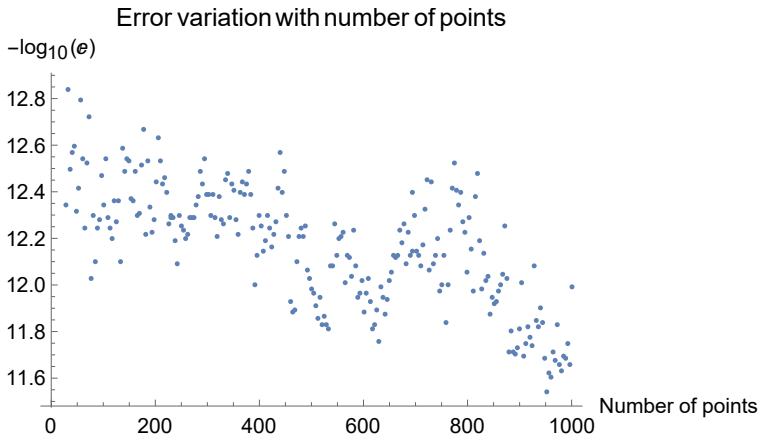

Figure 9: Interpolation error versus number of points used to make the interpolation. The error represented is the "average relative error" calculated as the sum of all the relative errors of the three components in all the grid points used to interpolate divided by the number of points times 3. In the y axes $-\log_{10} E(\%)$ is represented as a function of the number of points. The measured interpolated values have been generated using an expansion of ϕ using equations (19) and (20).

explicitly for this interpolating polynomial of higher order n and for more magnetic field data because the result presented here only shows the results for a given magnetic field and an expansion of the scalar potential up to a 5th higher degree polynomial. The convergence tests for functions that are not polynomials expansions is studied in the next section. We have repeated the same test of the number of points for measurements obtained from an expansion of ϕ with harmonic polynomials to the $5th$ power. The results have roughly the same behavior as for the other function interpolated.

Another test that we have computed to check the validity of our procedure consists on the generation of a dipole in an arbitrary direction in two dimensions assigning random values to J_1 K_1 from Table 2 and then applying random rotations to the configurations. Note that we also introduce a constant component in the z direction B_{0z} because we calculate the relative errors between the interpolation and the measured values and we need to compute $\frac{|B_{measured}-B_{interpolated}|}{|B_{measured}|}$. If $B_{measured}$ is zero somewhere then the program cannot divide by zero and we cannot estimate the interpolation error. We have run a program that computes the interpolation using a $5th$ order polynomial for ϕ over 400 randomly oriented dipoles and then we get the relative error for each of the three B components in each of the grid points used to interpolate the function with equation (56).

We have tested the procedure for the two arrangements of cubes mentioned in the previous section: The cross arrangement of 7 cubes and the 8 cubes around the origin arrangement with a normalized Lattice $L = 1$. The Figure 11 shows all the residuals for B_x, B_y and B_z obtained over the 400 random dipoles.

The Relative Errors are mostly of the order $10^{-14} \sim 10^{-13}$ what indicates. These quantities are expressed in %, the machine epsilon of Wolfram Mathematica is 2.22045×10^{-16} and the

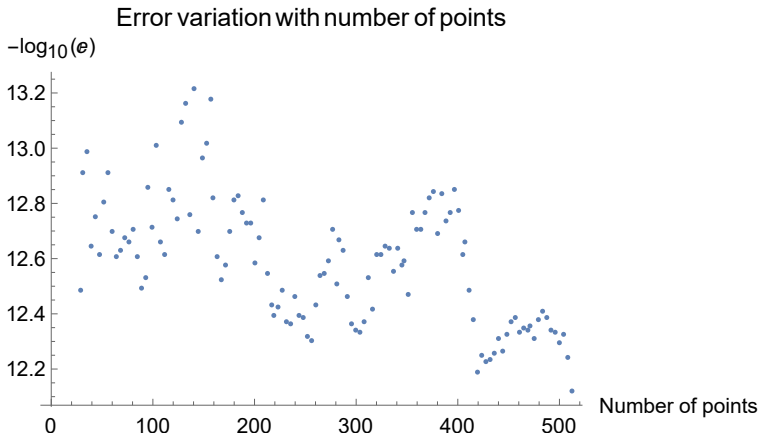

Figure 10: Interpolation error versus number of points used to make the interpolation. The measured interpolated values have been generated using an expansion of ϕ with harmonic polynomials.

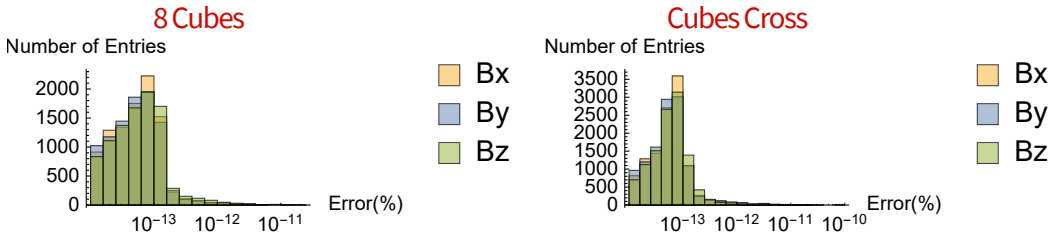

Figure 11: Dipole Interpolation Errors. Left Plot shows the errors when using "8 Cubes" arrangement to interpolate and Right plot shows the errors when using "Cubes Cross" arrangement to interpolate. The two plots represent the interpolation errors calculated from (47) of 400 randomly oriented dipoles according to Table 1.

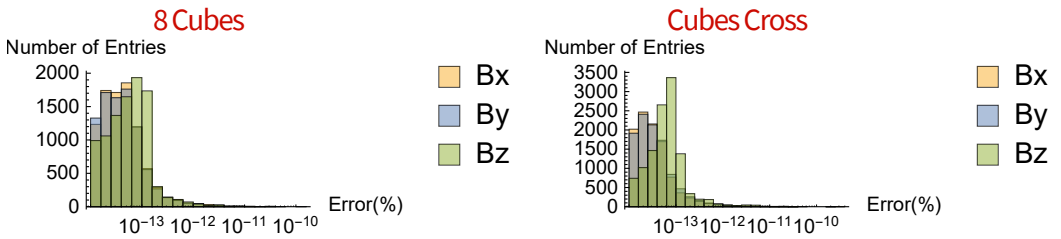

Figure 12: Quadrupole Interpolation Errors. The two plots represent the interpolation errors calculated from (47) of 400 randomly oriented quadrupoles created according to Table 1

Machine precision is 10^{-15}. The Machine Epsilon gives the difference between 1.0 and the next-nearest number representable as a machine-precision number. We can assure that the results for dipole interpolation are sufficiently accurate. As we are applying rotations and random generated numbers are used to generate the dipole and this implies calculations we have again loses of exact precision because of machine rounding error as for the test of number of points. When we apply a rotation for example there are two numbers, one bigger and one smaller that are substracted or added and information is lost there. Decimals are lost during the calculation and the result for the residuals is not zero in all the cases. However when the interpolation of a dipole is tested with exact values for the three components, for example $\vec{B} = (2.0, 3.0, 3.1546)$ the result for the residuals is 0 for all of them. From the Figure 4 we can also see that the behavior of the residuals obtained for the two cubes configurations is roughly the same.

This is the easiest case in which we have a constant function and the procedure reproduces it correctly. We have to check as well if the procedure continues working correctly and not giving big residuals when higher order components with dependence in the spatial coordinates are introduced.

Higher multipoles have been tested as well with the two configurations. The procedure followed has been the same, we have generated over 400 quadrupoles, sextupoles and octupoles with random values and then we plot the residuals of the three components. We always add the dipole part of the expansion i.e. three constants for each of the B components for the reasons mentioned above when computing the relative error. All the interpolations have been made using the procedure explained in section 6.2 expanding the scalar potential to a maximum power 5. As the expansion of the scalar potential reaches always higher powers than the expansions used to generate the data that we later use to interpolate the procedure should give sufficiently accurate results for the quadrupole, sextupole, octupole.

The results are presented in Figures 5,6, and 7. The residuals again are mostly $\leq 10^{-13}\%$. The maximum errors obtained are not greater than $10^{-9}\%$. Bigger differences for the residuals of the three components are obtained.

We have tested also how the interpolation error scales if we change the Lattice of our grid. For a fixed magnetic function that we have created using a random harmonic polynomial expansion of ϕ up to the $5th$ power we have interpolated the data for different grid spacing

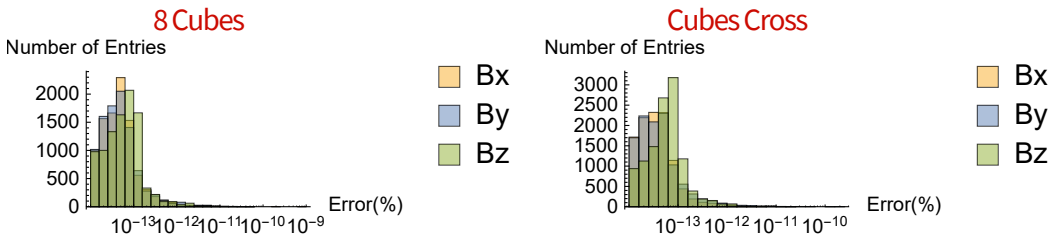

Figure 13: Sextupole Interpolation Errors. The two plots represent the interpolation errors calculated from (47) of 400 randomly oriented sextupoles created according to Table 1

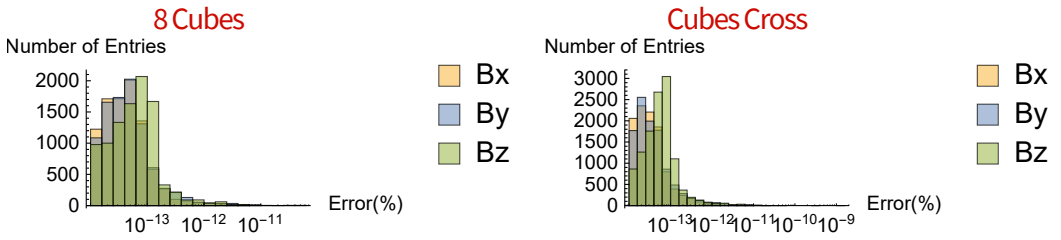

Figure 14: Octupole Interpolation Errors. The two plots represent the interpolation errors calculated from (47) of 400 randomly oriented octupoles created according to Table 1

from $L = [10^{-1}, 10^{-3}]$m. The test has been performed with the two cubes arrangement. As shown in the Figure 15 the interpolation error does not change significantly if the grid spacing is changed. However, this behavior we cannot assure that will remain for other interpolated data.

Another test that we have performed consists in estimating how good remains our interpolation if we want to interpolate a polynomial that is one degree more than our B expansion that we have limited to $4th$ order. In order to estimate that we create a magnetic field:

$$\vec{B} = \vec{B}_{4thorder} + K_1 h \vec{B}_{higher} \tag{56}$$

$\vec{B}_{4thorder}$ is a general polynomial expansion of the magnetic field to a $4th$ power. We will use the general field expansion used in the previous section parametrized in terms of K_1. $\vec{B}_{higher}$ is constructed from the gradient of an harmonic polynomial ${H^n}_{b_1,b_2,b_3}(x,y,z)$ of order $n \geq 5$; $\vec{B}_{Higher} = \left(\frac{\partial H^n}{\partial x}, \frac{\partial H^n}{\partial y}, \frac{\partial H^n}{\partial z}\right)$. To compare how the interpolation error is influenced by the addition of this higher order term we parametrize this extra term with h where it goes from 0 to 1 so we compare the magnitude of this term with the magnitude of the dipole term. We compute the interpolation for different values of h and the results obtained are presented in Figures 16 and 17.

From Figure 16 we observe that the interpolation error seems to scale smoothly but for some values of h it blows up extremely. These peaks are different for different extra terms added so it indicates that it is related to the extra term added. The structure used (Corners Cross or Corners 8) also changes the distribution of the peaks so it influences these peaks as well. The origin of this peaks is not clear but it is related to the structure used and also the function used to add the extra term. In our study we could not find which is the reason

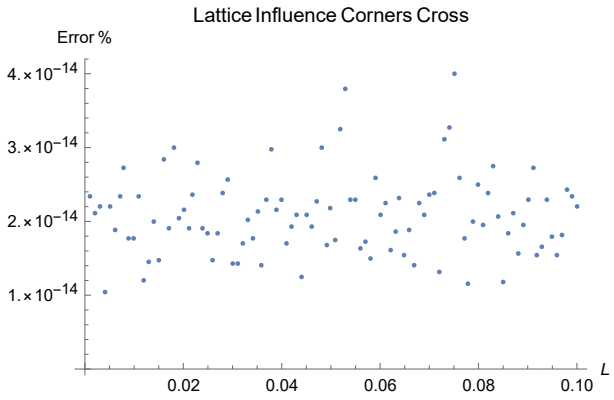

Figure 15: Interpolation error for different grid lattices L. The errors have been calculated for grid spacings that go from 10^{-3} to 10^{-1} in steps of 10^{-2}. We do not observe a clear influence of the grid lattice in the results

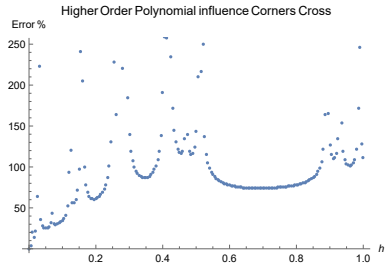

Figure 16: Average Interpolation error (%) as a function of the higher order term parameter h for Corners Cross structure. The peaks observed are particular for each different extra term.

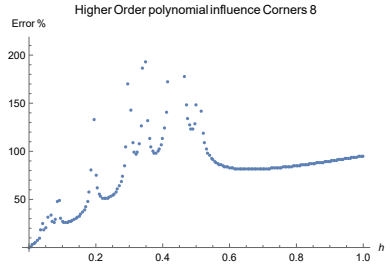

Figure 17: Average Interpolation error (%) as a function of the higher order term parameter h for Corners 8 structure.

of this behavior however it seems to be related to the roots of the extra polynomial added. However this does not interfere in our aim for quantifying how the interpolation error scales. As the procedure is tested for normalized grid positions $(x, y, z = 0, \pm 1, \pm 2, ...)$ we expect the higher term of order n parameter to be multiplied by a factor L^n where $L < 1$ in the grids commonly used. As an example the grid in the LHCb magnet is $L = 10^{-1}m$ and in other experiments the grid is $L = 10^{-3}$ so we study the behavior of the parameter that introduces the extra polynomial from 0 to 10^{-3} and we see that it scales linearly. After making a fit we can quantify limits for the error of this extra term.

Table 3: Quantification of the interpolation error of the extra term

	Error Range (%)	**Parameter h range**
Corners 8	[0, 1]	$[0 \sim 0.03]$
Corners Cross	[0, 1]	$[0 \sim 0.02]$

We set the maximum relative error of our interest, sufficiently accurate, as 10^{-2} (1%) because when bigger errors are obtained as a function of h the linear fit for the scale of the interpolation error does not work anymore.For different extra terms added the linear behavior is conserved in the error range established however the slope of the linear fit varies slightly but the order of magnitude of the slope is roughly always the same. The behavior for the two arrangements of cubes is similar.

Our procedure reproduces with sufficient accuracy multipole expansions and adding an extra term does not increase extremely the error if the parameter of this perturbation term is small enough inside the range typically obtained for real fields in which higher terms always contribute less to the magnetic field. Let us now study how our method can interpolate functions that are not a polynomial cut off at a certain power.

6.2.2 Problems of convergence

In order to ensure if our method is capable of reproducing general magnetic fields or harmonic functions and to test the convergence of the method we have used the harmonic functions mentioned in Section 4. The results presented in the previous section showed us that the procedure is capable of reproducing an exact solution with the machine precision but we need to know how good could it be if the function interpolated is not a polynomial expansion cut at a certain power. We need a function that it is harmonic in order to test that. In the Section 4 we showed several examples of functions that can be used as a potential from which we derive a magnetic field. These functions are not analytical in the origin of coordinates and its values blow up when we approach the origin. We cannot interpolate these functions close to the origin of coordinates. In order to avoid that we have tried to interpolate these functions inside cubes that do not contain the zero. We have just displaced the points of the arrangements in the space by a vector $\vec{a} = (d, d, d)$ in which we have chosen the three components to be equal for simplicity. With this displacement vector what we try to do is define regions displaced $r = \sqrt{3}d$ from the origin in which the derivatives are lower and then study how the interpolation behaves there. As a increases r in the interpolation points increases so the functions start to be smoother and we can try to interpolate them.

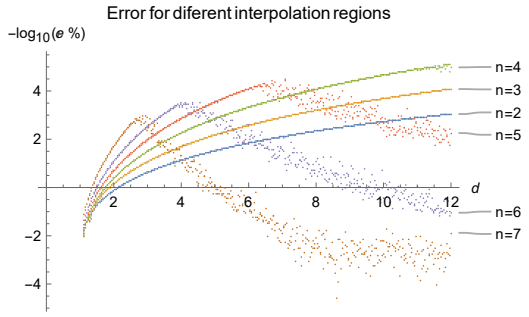

Figure 18: Average Interpolation error (%) as a function of the displacement d for the interpolation of the function $\phi = \frac{1}{r}$ with a 5th order expansion using harmonic polynomials.

We have worked with the lattice constant equal to 1 and the configuration Corners 8. We work with the function $\phi = \frac{1}{r}$. Computing the gradient of that function we obtain 3 functions that can be used to generate magnetic field data. The function will be reproducible with our harmonic polynomials in regions not so close to the origin of coordinates. As a first approach we compare the interpolation error for different d using an harmonic interpolation up to different maximum power in ϕ. The interpolation error varies as a function of d because in regions closer to the origin of coordinates as it was expected our harmonic polynomials are not capable of reproducing the function sufficiently accurately. However we observe as well that for bigger d the interpolation error increases as well depending on the degree of the interpolation polynomial. The derivatives of the function have to be taken into account. In regions where the derivatives are higher the function varies rapidly and so on higher order polynomials are needed to reproduce the function. However for regions further from the origin taking an interpolation polynomial of order N big enough could lend results that are worst than the linear interpolation. Figure 18 shows clearly why taking a higher order polynomial to make the fit is not always translated into an increase of the precision. The model is not orthogonal and so on each term of the expansion cannot be evaluated separately and we cannot test if each term separately increases or decreases the error. This behavior indicates that the polynomials do not converge to interpolate the function i.e. taking more terms is not always translated in an increase of the precision. As the d influences the error because taking more terms is not always worth while we expect the lattice to influence as well. Although this is showed only for the interpolation of the function $\vec{F} = \vec{\nabla}\frac{1}{r}$ this behavior is similar for the functions mentioned at the end of Section 4 that are not analytical in the origin of coordinates.

We could think that making the lattice narrower will increase our precision because we interpolate in a smaller region, however as the convergence is not well-behaved taking a polynomial of order N in one sufficiently bigger region could give us completely wrong results. We show that with the graphs in Figure 19.

We have calculated the variation of the interpolation error as a function of the Lattice for different n (maximum degree of the polynomials used to expand ϕ) and different d that define regions closer or further from the origin of coordinates.. The lattice in fact influences the interpolation error and there seems to be an optimum lattice ~ 0.06. For the regions closer to the origin that we have expressed with $d = 3$ the accuracy reached with 6 th and 5 th order

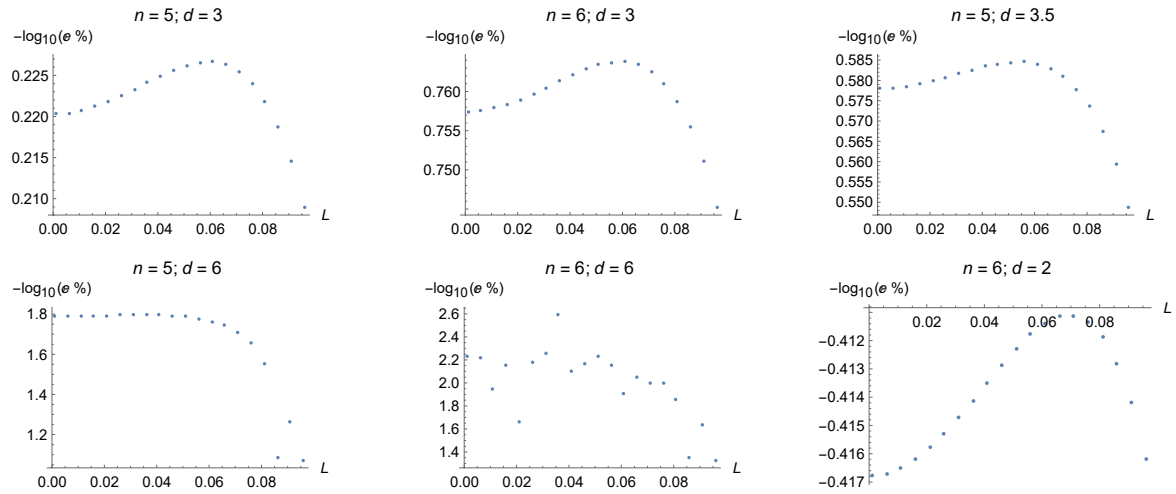

Figure 19: Lattice influence for different combinations of n (order of the ϕ expansion) and d displacement of the interpolation cube from the origin when interpolating the vectorial function $\vec{F} = \vec{\nabla}\frac{1}{r}$ with harmonic polynomials.

interpolation polynomials is in the best case approximately $\sim 1(\%)$. However for the range of lattices studied the errors are not higher than $10(\%)$ except the case $d = 2$ in which we know we are close enough to the origin of coordinates to expect a poor interpolation because of the bigger derivatives. However there seems to be as well a minimum for the interpolation error for a specific grid lattice. For the region labeled as $d = 6$, further from the origin a 5 th order polynomial reaches a precision of 10^{-2} for small enough lattice grid spacing but as the grid lattice increases after the optimum the error blows up. Taking a higher polynomial degree (6th order) gives a fluctuating behavior for the error around the precision reached with $5th$ order and is not influenced clearly by the lattice. This is shown in the Figure 20 in which a longer range of lattices is used.

Now we take a 7 th order polynomial in that region to check if we could improve the accuracy. The result is a similar behavior as for $n = 6$ but the errors are two orders of magnitude bigger. We see the problem of convergence here, taking a higher order polynomial decreases the precision that now is not influenced by the grid spacing.

Let us now return to study in more detail the closer regions as $d = 2$. We saw that the interpolation was poor using a 5th order polynomial because it gave use the bigger errors of order 10%. Now using a 7 th order polynomial in the region $L = [10^{-3} - 10^{-1}]$ units of distance the precision is increased. Nevertheless if we plot the error over a large range of grid spacing we obtain the plot of Figure 22. We observe peaks again for values of the grid spacing similar as the obtained for the higher order test mentioned in the previous section. However here the errors for these peaks are not as big as they were before and also the precision is increased as the Lattice increases. This result is confusing because for a region in which the derivatives are higher i.e. closer to the origin a smaller lattice that is translated in taking a smaller region to interpolate seems to be more suitable for achieving better precision. This

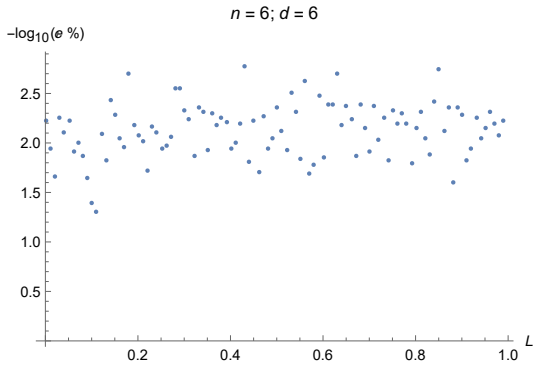

Figure 20: Lattice influence for the interpolation of the function $\vec{\nabla}\frac{1}{r}$ using a 6th order polynomial in ϕ for the region defined by $d = 6$. The errors fluctuate randomly around a precision $10^{-2}\%$ due to machine rounding errors.

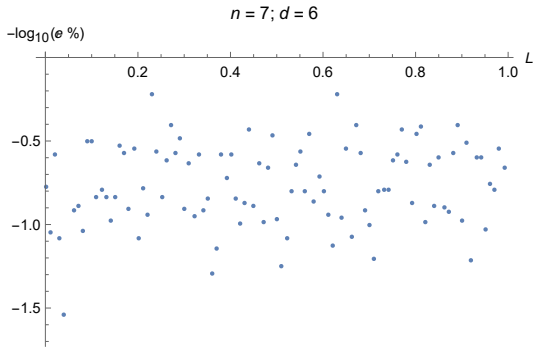

Figure 21: Lattice influence for the interpolation of the function $\vec{\nabla}\frac{1}{r}$ using a 7th order polynomial in ϕ for the region defined by $d = 6$. The errors fluctuate randomly around a precision 10% due to machine rounding errors. Taking a 7th order polynomial in that region $d = 6$ gives less accurate results than taking a 6th order polynomial in ϕ.

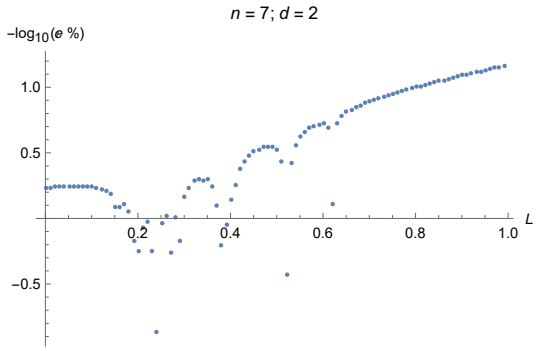

Figure 22: Average Interpolation error (%) as a function of the grid spacing for the region defined by $d = 2$ using a 7th order polynomial in ϕ

recalls the fact that there are optimum lattices that improve the result and that particular values of L that ruin the interpolation.

From the results obtained and particularly from Figure 18 we can extract that in principle a higher order polynomial does not give always a better approximation, it is influenced by the d and so on by the derivatives. It is important to know how the function changes via the derivatives because we can make an idea of how many terms do we need to achieve a precision and if a polynomial of order N will give a better approximation than a polynomial of order $N-1$. Nevertheless the results obtained do not indicate that the method fails and that there is no way of controlling it. We have tried to interpolate a function that has a singularity at the origin and the results have been decent in most of the regions even close enough to the origin. It is so difficult to interpolate a function as $\vec{\nabla}\frac{1}{r}$ and the errors we have obtained for the correct choice of the expansion are small. We see that the method is robust because it is capable of interpolating that function within an small error. However there is something clear, the harmonic polynomials do not always converge to reproduce any kind of harmonic function.

With this results we also point out the importance of knowing the derivatives of our magnetic field map. As it is necessary to know how to divide the volume in interpolation regions and to know how many terms should we take the derivatives play an important role in the interpolation. In the next section we present a method that uses the derivatives of the magnetic field at the grid points to fit the function and that provides sufficient accuracy.

7 Interpolation using Maxwell Elements

In this section we describe a method that can be used to interpolate a magnetic field on a grid using the B values and the derivatives at the grid points. This algorithm is valid for static current-free magnetic fields. It is based on the use of fourth-degree vector polynomial functions (Maxwell Elements) which solve the static current-free Maxwell equations.

This method was proposed by Paul Probert [5] and presents relevant new features for the magnetic field interpolation because of its accuracy and speed advantage over other methods. Furthermore, the interpolation error of this procedure can be reduced significantly if the grid

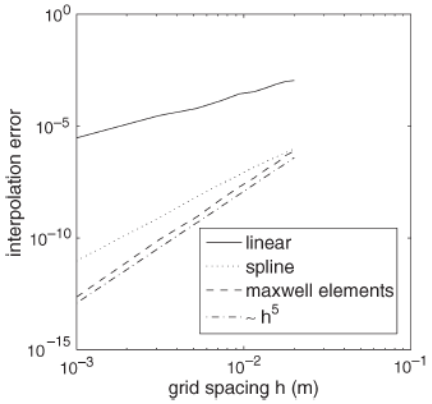

Figure 23: Interpolation errors versus grid spacing, as the magnitude of the vector difference between the interpolated field and the exact result, divided by the magnitude of the exact field[5].

spacing is reduced as shown in Figure 23. If we compare the interpolation error of this method with the interpolation error of other methods we see that it scales as the fifth power of the grid spacing and so on it is reduced considerably more than other methods such as spline 3D interpolation or linear interpolation.

Because of the Maxwell equations for a static magnetic field in the absence of current we always calculate the field gradient along with the field. The gradient of B can be represented by a symmetric traceless matrix with five independent numbers. This can be done because of the relations that the Maxwell equations impose between the derivatives.

$$\nabla \vec{B} = \frac{\partial B_x}{\partial x} + \frac{\partial B_y}{\partial y} + \frac{\partial B_z}{\partial z} = 0 \tag{57}$$

$$\vec{\nabla} \times \vec{B} = \widehat{x}(\frac{\partial B_z}{\partial y} - \frac{\partial B_y}{\partial z}) + \widehat{y}(\frac{\partial B_z}{\partial x} - \frac{\partial B_x}{\partial z}) + \widehat{z}(\frac{\partial B_y}{\partial z} - \frac{\partial B_z}{\partial y}) = 0 \tag{58}$$

This implies

$$\frac{\partial B_z}{\partial y} = \frac{\partial B_y}{\partial z} \tag{59}$$

$$\frac{\partial B_z}{\partial x} = \frac{\partial B_x}{\partial z} \tag{60}$$

$$\frac{\partial B_y}{\partial z} = \frac{\partial B_z}{\partial y} \tag{61}$$

From the gradient of B we see that knowing two of the derivatives $\frac{\partial B_i}{\partial x_i}$ specified there we can infer the third one. The rotational sets the relations for the other 6 derivatives leaving only three independent. So we only need two of the $\frac{\partial B_i}{\partial x_i}$ derivatives and three of the $\frac{\partial B_i}{\partial x_j}$ derivatives.

The magnetic field and its gradient can be represented by a vector Y_l,$l = 1, ..., 8$, com-

prising three components of B and five independent components of ∇B.

$$Y(\vec{r}) = [B_x(\vec{r}), B_y(\vec{r}), B_z(\vec{r}), \nabla B(\vec{r})_{xx}, \nabla B(\vec{r})_{xy}, \nabla B(\vec{r})_{xz}, \nabla B(\vec{r})_{yy}, \nabla B(\vec{r})_{yz}] \tag{62}$$

Where a general field point is $\vec{r} = (x, y, z)$ and $\nabla B_{ij} = \frac{\partial B_i}{\partial x_j}$. We need to precompute and store these Y values in the regular grid. These values will be the data we use to make the fit. We have to remark here that we are using the derivatives in the grid points and in our first approach we set our main goal to interpolate the magnetic field using only the B values at the grid points. In the original paper where this procedure was explained the derivatives were given directly as measured data. However the derivatives can be calculated numerically from the B values. We just need three consecutively points in one direction if we want to calculate numerically the derivative along this direction (There are also more methods like the Five-point stencil where 5 points are used). As we have a regular cubic grid this can be done easily except in the boundaries grid points where we can differentiate numerically using only 2 points. Nevertheless these approximated derivatives will introduce more errors in our method and if we want to increase the precision we need to make the grid finer. We show a little study of the error of the derivatives in Appendix A.

Suppose that we have the Y values stored we seek a general polynomial to express the field at some point $\vec{r} = (x, y, z)$. We expand about some point $\vec{r_0} = (x_0, y_0, z_0)$.

$$\begin{aligned}
\vec{B} = \widehat{x} \sum_{i,j,k=0}^{i+j+k \leq N} a_{ijk}(x - x_0)^i (y - y_0)^j (z - z_0)^k \\
+ \widehat{y} \sum_{i,j,k=0}^{i+j+k \leq N} b_{ijk}(x - x_0)^i (y - y_0)^j (z - z_0)^k \\
+ \widehat{z} \sum_{i,j,k=0}^{i+j+k \leq N} c_{ijk}(x - x_0)^i (y - y_0)^j (z - z_0)^k
\end{aligned} \tag{63}$$

where N is the maximum degree of the polynomial. In this method we develop the magnetic field directly rather than constructing it from a potential as we have done in the other procedure showed here. Due to differentiation, the B field in the Harmonic Polynomials interpolation will be represented by a polynomial of one degree less than the scalar potential, forcing the polynomial fitting of the scalar potential on the grid to a higher degree in order to obtain the same precision.

The upper limit on the degree N is set by the limited amount of data we use to interpolate. The number of terms of the expansion up to a power N is equal to $(N+1)(N+2)(N+3)/2$. We have expanded each of the three components of B as a sum of powers x^i, y^j and z^k with $i + j + k \leq N$. The number of terms for a given power n scales as $1/2(n + 1)(n + 2)$. For each of the three components of B we have a total number of terms up to a maximum power N:$\sum_{n=0}^{N} 1/2(n + 1)(n + 2) = (N + 1)(N + 2)(N + 3)/6$. As we have three components of B the total number of terms will be $(N + 1)(N + 2)(N + 3)/2$. We wish to interpolate the magnetic field in a point using only the corners of the cube in which this point is contained.

With 8 points of the cube×8 values for each point (3 B values plus 5 derivatives) we are supplied with 64 known values for the fitting procedure. In order to ensure that the grid is not too fine (the storage required scales as the cube of the inverse grid spacing) while maintaining accuracy, we seek the highest possible degree polynomial. This allows us to develop the magnetic field up to a power of 3 because the number of parameters needed for $N = 3$ is $(3+1)(3+2)(3+3)/2 = 60$. However due to the Maxwell equations this number of parameters needed is reduced in the same way as it was for the Harmonic Potential and so on we can develop the magnetic field up to a 4th power. We show that in the next section.

7.1 Maxwell Elements

For a maximum power N we have seen that for each of the components of B we need $(N+1)(N+2)(N+3)/6$ parameters or different powers of x, y and z to expand the field up to a maximum power N. Applying directly the Maxwell equations to the expansion of the magnetic field we find relations between the parameters that reduce the degrees of freedom of our system. We have designed a procedure in Wolfram Mathematica that finds these equations that relate the parameters for a given maximum power of the expansion M. Finding the reduced matrix of the system (there are equations that appear from the Maxwell equations that are not linearly independent) we know how many parameters we need to represent B up to a maximum power M satisfying Maxwell equations.

As we have seen before, the number of parameters needed when we applied the harmonic condition to the potential where reduced to only $(N+1)^2 - 1$ parameters for a maximum power of the potential N. Now we fix the maximum power of the magnetic field. As B is formed from the derivatives of ϕ if we want to develop the magnetic field from ϕ to the same degree as the expansion of B to a maximum power directly the degree of B will be $M = N-1$ where N is the maximum power of the potential.

As we will show now, the number of terms needed to expand the magnetic field up to a maximum power $N-1 = M$ is $(M+2)^2 - 1 = (N+1)^2 - 1$. This condition for the function parameters obeying Maxwell equations is equivalent to the one that we found before for the terms in the potential ϕ up to a power N $(N+1)^2 - 1$.

We have run this explicitly for $M = 4$ because we get $(6)^2 - 1 = 35$ free parameters that we can find using the 64 known values of the 8 corners of one cube. We can fit also a 5th order polynomial in B with the 64 known values. The code used with Mathematica is shown in Appendix C. The procedure is designed to find out how many parameters we have free after imposing the Maxwell conditions.

The Maxwell equations imposed 80 equations, after reducing the matrix of that system we have seen that only 70 of these equations are linearly independent. The number of parameters of the expansion $(4+1)(4+2)(4+3)/2 = 105$ is reduced to only $105 - 70$(equations)$= 35 = (4+1)^2 - 1$. For developing the magnetic field B up to a 4th degree polynomial obeying Maxwell equations we need 35 parameters. As we see this result is in agreement with the general solution for the harmonic potential.

The 70 sets of equations give sufficient information to solve for sets of parameters that ac-

complish these equations. We have seen that expanding the field only requires 35 functions (number of parameters). The system of equations is an undetermined linear system with 105 parameters and 70 equations. We can find 105-70=35 sets of the 105 parameters that solve this matrix problem. Based on the harmonic solution of the potential that states that an harmonic function of degree n can be expanded as a sum of $2n+1$ independent functions we arrange the sets of parameters in powers of the polynomial. We find $2(m+1)+1$ (As B is formed from the derivatives of this harmonic function the power of B is lower that the degree of the potential $m = n-1$ with m being the power of B and n the power of the harmonic function) sets of parameters for a power m that accomplish the equations imposed to B. For example we find 5 sets of parameters involving only the parameters of the linear terms in the expansion of B because $2 \times 2 + 1 = 5$. Doing this we create sets of parameters that construct vector polynomials functions that are closely related with the harmonic polynomials described in the previous section and that are similar to the concept of elements in numerical partial differential equations and finite-element analysis. The sets we found are not uniquely and because the system is an undetermined linear system we could find an infinity of new parameter values verifying as well the Maxwell equations.

The 70 equations relate the coefficients of the same power $i+j+k=m$ because we need the divergence and curl equations to hold for all different powers and the derivatives involved are always of first order. We do not have relations between $a_{2,0,0}$ and $a_{1,0,0}$ for example. We are solving for the linear elements so we pick the equations involving the linear parameters.

$$b_{1,0,0} = a_{0,1,0} \tag{64}$$

$$a_{0,0,1} = c_{1,0,0} \tag{65}$$

$$b_{0,0,1} = c_{0,1,0} \tag{66}$$

$$a_{1,0,0} + b_{0,1,0} + c_{0,0,1} = 0 \tag{67}$$

We need to form 5 elements ($2(m+1)+1=5$) from these equations. We have 9 parameters and 4 equations, so we have 5 degrees of freedom. The simplest way is to set all the parameters to zero except two for which we assign a value.

$$b_{1,0,0} = 1 \quad a_{0,1,0} = 1 \tag{68}$$

And the rest of parameters equal to zero. We could have choose instead of assigning the value 1 any other number but for simplicity we do that with the lower integers possible.

$$b_{0,0,1} = 1, c_{0,1,0} = 1 \tag{69}$$

$$a_{0,0,1} = 1, c_{1,0,0} = 1 \tag{70}$$

$$a_{1,0,0} = 1, c_{0,0,0} = -1 \tag{71}$$

$$a_{1,0,0} = 1, b_{0,1,0} = -1 \tag{72}$$

Each of these lines represents one element. Note that by "element" we refer to a vector function which is apolynomial in x, y and z as in (15). By substituting $a_{i,j,k}$, $b_{i,j,k}$ and $c_{i,j,k}$ of these lines into (15), one obtains an element. We can compute the gradients of these elements and form the quantities. As we increase the power m the task becomes a little bit more

difficult and tedious but the procedure for finding the elements is easy and follows as we have showed.

As we can see the method for obtaining these Maxwell elements is tedious and long so we can try to form similar vector functions directly using the harmonic polynomials mentioned to construct vector functions in a similar way as it has been done in Section 6. We show in Section 7.3 how easily we can obtain functions similar to the Maxwell Elements without having to do all the algebra of the Maxwell equations and solving for the parameters.

7.2 Fitting Procedure

In this section we describe the method used to interpolate the magnetic field on a cube using these 35 Maxwell elements. When substituting the parameters specified by the 35 Maxwell elements in (15) we obtain a function of three components in three variables that obeys Maxwell's equations. We can compute the gradients of these elements and create the quantities Y_l designating element m as $Y^m{}_l(\vec{r})$, $m = 1, ..., 35$, $l = 1, ..., 8$.

When we want to interpolate the magnetic field on a point we use the eight corners of the cube that contains this point $\vec{r}$. We have to expand the series around a point $\vec{r}_0$. We will set this point as the center of the cube in which we are interpolating. The grid need not to be regular, and the points $\vec{r}_n$ need not to lie exactly on the vertices of a cube. In the procedure that Paul Probert explains in its paper the points lie on a regular grid in cylindrical coordinate space, even though the field data are stored as components in the Cartesian x, y, z space.

We represent the data $Y_l(\vec{r})$ at the field point where we want to calculate the magnetic field by a sum over the $M = 35$ elements as $Y_l(\vec{r}) = \sum_{m=1}^{M} f_m Y^m{}_l(\vec{r})$ where the f_m are the coefficients that multiply each element in the expansion.

The method we use to find the coefficients f_m is the standard linear least squares problem described in section 4.2. We minimize the squares of the residuals at the corner points. We let our error be given by:

$$\chi^2 = \sum_{l=1}^{L} \sum_{n=1}^{N} \left[\frac{Y_l(\vec{r_n}) - \sum_{m=1}^{M} f_m Y_l^m(\vec{r_n})}{\sigma_{l,n}} \right]^2 \tag{73}$$

where $N = 8$ because we use the eight corner points of a cube and $L = 8$ because each vector Y_l has eight components. The values $\sigma_{n,l}$ represent the measurement error (standard deviation) of the data values at the given points. As we have showed before this problem is solved by differentiating respect to the coefficients f_m and equating this to zero. We obtain a matrix problem $Af = g$, where

$$A_{u,v} = \sum_{l=1}^{L} \sum_{n=1}^{N} \frac{Y_l^u(\vec{r_n}) Y_l^v(\vec{r_n})}{\sigma_{n,l}^2} \tag{74}$$

$$g_u = \sum_{l=1}^{L} \sum_{n=1}^{N} \frac{Y_l(\vec{r_n}) Y_l^u(\vec{r_n})}{\sigma_{n,l}^2} \tag{75}$$

And for the errors of the obtained parameters f_m we obtain

$$\sigma^2(f_m) = C_{j,j} \tag{76}$$

Where $[C_{u,v}] = [A_{u,v}]^{-1}$ is the inverse matrix of A. The diagonal elements of $[C]$ are the variances (squared uncertainties) of the fitted parameters. The off-diagonal elements $C_{u,v}$ are the covariances between f_u f_v.

The matrix A is a 35×35 matrix that we have to invert for each interpolation cube using 4th power Maxwell elements expansion. If we do not expand B to a power 4 this matrix is reduced. In the routine we have designed we can decide up to which power (maximum 4) we want to interpolate. If we use the Maxwell elements up to a power m the matrix to be inverted will be $M \times M$ where $M = (m+2)^2 - 1$. The code in which we implement this method is showed in Appendix D.

The matrix inversion influences the computation time of the procedure, however the size of this matrix is small enough that it can be solved entirely and easily with optimized linear Algebra routines in cache memory. Furthermore we have to remark that most of the computation time is spent in calculating the values of the elements to fill the $A_{u,v}$ matrix.

To test the procedure we have implemented a code in Wolfram Mathematica that computes this interpolation for a given set of data. In this implementation we have simplified the grid and we test it in the standard centered cube we presented before. In this cube the center is the origin of coordinates and the corners of the cube are labeled by $x = \pm 1$, $y = \pm 1$ and $z = \pm 1$. Our point for the expansion is then the point $\vec{r_0} = (0, 0, 0)$.

As well as the harmonic polynomials interpolation the method is capable of reproducing perfectly the multipole expansions. However we have not tested a hard function as $\frac{1}{r}$ because we cannot go deeper into that in this Bachelor Research Project and Paul Probert in its paper provides sufficient information about the accuracy of the method interpolating real magnetic field data values. The method is more accurate than other interpolation methods as spline 3D and linear interpolation. However the algebra behing finding the Maxwell elements is tedious and we can try to overcome it. This is what we explain in the next section.

7.3 Harmonic Interpolation with derivatives

As well as we have done it for the Maxwell elements we can form the 35 vector functions from the 35 Harmonic polynomials used to expand an scalar potential up to the 5th power. If we denote $\phi = \sum_{m=0}^{M} a_m H_m(x, y, z)$ where H are the harmonic homogeneous polynomials used in the previous sections and M is 35 in this case because we expand the magnetic field up to a 4th order polynomial . We can form the 35 quantities Y_l^m as:

$$Y^m(\vec{r}) = [\frac{\partial H_m(\vec{r})}{\partial x}, \frac{\partial H_m(\vec{r})}{\partial y}, \frac{\partial H_m(\vec{r})}{\partial z}, \frac{\partial^2 H_m(\vec{r})}{\partial x^2}, \frac{\partial^2 H_m(\vec{r})}{\partial x \partial y}, \frac{\partial^2 H_m(\vec{r})}{\partial z \partial x}, \frac{\partial^2 H_m(\vec{r})}{\partial y^2}, \frac{\partial^2 H_m(\vec{r})}{\partial z \partial y}] \tag{77}$$

With these 35 vector functions formed from the Harmonic polynomials we know that they satisfy the Maxwell's equations for static-free current regions. The same procedure used with

the Maxwell Elements can be applied now using these 35 Y^m vector functions.

The Maxwell Elements do not need to be calculated as we have done in the section 7.1. and they can be formed directly from the Harmonic polynomials. The polynomial vector functions obtained are not identical but they are closely related. We can extend this procedure to higher order interpolation of the magnetic field without having to solve the Maxwell equations as we have done in section 7.2.

8 Conclusions

In this report, we have presented a method for incorporating the constraints that are imposed by Maxwell's equations into fits of magnetic field values defined on a regular 3D grid. Maxwell's equations are introduced in the fits by making use of a fitting function that consists of a series of harmonic polynomials in 3 variables (x, y, z). In order to fit the measurements a Least squares fit is performed reducing the problem to a matrix inversion. The size of this matrix problem is small enough such that modern desktop processors, using optimized linear algebra routines can solve the problem entirely in cache memory. The fitting of the function does not need a lot of time to be performed, however if a lot of points are taken into account for the fitting the matrix evaluation will slow the procedure. Moreover the matrix dimensions can increase if we use a high order degree polynomial; as a consequence the procedure will require more time to fit the function. However as the fitting of the function does not need to be really fast this is not a problem.

The method reproduces pure multipole magnetic fields with machine precision and is capable of reproducing other functions with less precision but still accurate if the correct order of the expansion is chosen. Moreover, in principle the lattice used does not influence clearly the precision of the method.

The important advantages of this method are the reduction of the number of parameters needed to express the fitted function as a consequence of the constraints introduced by Maxwell's equations. If the number of parameters is reduced then the evaluation of the function is faster and the storage required to save the magnetic field map is reduced. For a magnetic field map of a particle physics detector as the one in LHCb that is accessed thousands of times during the reconstruction of the events this results advantageous.

However the procedure is not an orthogonal model what states that the evaluation of the terms introduced to the fitting function cannot be done separately and so on it cannot be seen beforehand if an extra term reduces or increases the precision. Moreover as we are using a polynomial model for equidistant points the convergence is not always guaranteed as seen when trying to interpolate difficult functions. A further study of how the polynomials converge to any sort of function should be done in order to ensure the correct behavior of the polynomials. Moreover the strange behavior observed when trying to interpolate an extra term where the error blows should be investigated in order to extract which is the reason behind that and ensure that this does not imply other facts that could lend to complete wrong results when interpolating real data.

Apart from the Harmonic polynomials interpolation two more methods are presented in which the derivatives at the grid points are used to fit the function as well as the B values. Both methods solve Maxwell equations in the absence of current and are based in the same fitting procedure as the Harmonic Polynomials interpolation because they consist in minimization of the least squares fit. The Maxwell elements method results precise but the task we have to do before for solving the Maxwell equations for the polynomials is tedious and requires a lot of algebra. We can overcome this problem and make it easier using the harmonic polynomials directly to construct vector functions that can work as the Maxwell elements. However these methods include the use of derivatives and the map of the LHCb does not measure the derivatives in the grid points directly and computing them by numerical methods introduces more errors in the creation of the map.

The LHCb magnet that serves as a motivation for this work could use the harmonic polynomials procedure to fit the B values in order to create the magnetic field map. If the derivatives are used the errors of these values will be of the order of 10^{-4} because of the Lattice. In principle this instead of improving the fit could decrease the precision because the magnetic field values of the LHCb map are found to be stable within an error of $\sigma = \pm 0.07$ Gauss $\sim 7 \times 10^{-5}$ T. Without using the derivatives the Harmonic Polynomials method can be used as well and in principle it could achieve the current precision for the magnetic field map (10^{-3}). Moreover both methods solve Maxwell's equations so it seems promising against the current interpolation procedure that does not take into account that fact in the same way. Moreover the method ties into effort to use Taylor methods for fast tracking of the particles through the magnet.

References

[1] H. Wind. J. Computational Physics 2, 274 (1968)

[2] E. T. Whittaker. Math Ann. 57, 333 (1902)

[3] E. P. Miles, JR and E. Williams. Proc. Am. Math. Soc. 6, 191 (1955)

[4] Yellow Report CERN 72-21, 1972. pages 53-106

[5] P. Probert, "High-performance interpolation of stellarator magnetic fields", IEEE Trans. Plasma Sci., vol. 39, no. 4, pp. 1051-1054, Apr. 2011.

[6] A. A. Alves Jr. et al. The LHCb detector at LHC, JINST 3 (2008) S08005

[7] Proceedings of the CAS-CERN Accelerator School: Magnets, Bruges, Belgium, 16-25 June 2009, edited by D. Brandt, CERN-2010-004

[8] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C* ,2nd ed. New York: Cambridge Univ. Press, 1992.

[9] Paul Wellin, Richard Gaylord, Samuel Kamin. *An Introduction to Programming with Mathematica*

[10] J. C. bernauer, J. Diefenbach. Measurement and tricubic interpolation of the magnetic field for the OLYMPUS experiment

[11] H. Wind, Nuclear Instrum. Methods 84, 117-124 (1970)

[12] Geoffrey T. Bodwin, Hee Sok Chung, José Repond. Implementation of Maxwell's equations in the reconstruction of the magnetic field in the g−2 storage ring May 30, 2018. 27 pp

[13] H. Wind, Nuclear Instrum. Methods 84, 117-124 (1970)

[14] A. Keune Nuclear Physics B (Proc. Suppl.) 197(2009) 163-166

[15] Adlène Hicheur, Géraldine Conti, Parametrization of the LHCb field map

[16] M. Schubiger, H. Schindler, F. Blanc, R. Lindner, M. Martinelli, P. Sainvitu Magnetic field map with 2014 measurements for the LHCb dipole magnet.

[17] A. Hicheur, G. Conti LHCb note 2007-093

[18] John G. Aspinall Spline Techniques for Magnetic fields,PFC/RR-83-27 DOE/ET-51013-98 UC-20 June 8, 1984.

[19] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C* , 2nd ed. New York: Cambridge Univ. Press, 1992.

[20] I. R. Savage and E. Lukacs, Tables of inverse finite segments of the Hilbert matrix in Contributions to the solution of systems of linear equations and the determination of eigenvalues(Ed. O. Taussy), National Bureau of Standards Applied Mathematics Series 39 pp. 105-108

[21] W.T. Eadie, D. Dryard, F.E. James, M. Roos and B. Sadoulet *Statistical Methods in experimental physics*(North-Holland, Amsterdam 1971), Section 8.4.4.

[22] http://library.wolfram.com/infocenter/MathSource/3203/+.

Appendices

A Numerical derivatives

In all the procedures used to test the interpolation of the magnetic field inside one grid cube the derivatives have been supposed to be given as measured data however not all the magnetic field measurements on a grid provide us with these measured values and if we want to use them we should find a way to calculate them. Numerical derivatives can be calculated by different methods. As we are provided with values on a regular grid this can be achieved easily. In this section we explain how these derivatives can be calculated numerically from the B values and which is the error that we introduce when we calculate the derivatives numerically.

We treat here different methods of differentiation of one variable, it can be applied to functions of 3 variables as our magnetic field B.

The derivative of a function is defined as

$$\frac{dy}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{78}$$

Thanks to Taylor theorem we know that a function can be expanded as a polynomial:

$$f(x+h) = f(x) + f'(x)h + \frac{f''(x)}{2!}h^2 + ... + \frac{f^{(n)}(x)}{n!}h^n + R_n \tag{79}$$

with $R_n = \frac{f^{(n+1)}(\epsilon)}{(n+1)!}h^n + 1$ where $x < \epsilon < x + h$. This means that as more terms for the expansion we take the error will be lower and that as smaller is the difference h the error R_n will be smaller for a given number of terms of the expansion for functions that are not a polynomial.

Taylor series allows us to calculate numerical derivatives of first order

$$f(x+h) = f(x) + f'(x) + O(h^2) \tag{80}$$

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h) \tag{81}$$

This numerical derivative is defined as forward differentiation. There can be two more different approaches for the numerical derivatives, the backward differentiation and the centered differentiation depending on the points used.

$$f'(x) = \frac{f(x) - f(x-h)}{h} + O(h) \quad \text{Backward} \tag{82}$$

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - O(h^2) \quad \text{Centered} \tag{83}$$

Using the Taylor expansion up to 2nd order can be used to calculate more accurate approximated first order derivatives.

$$f(x+2h) = f(x) + f'(x)2h + \frac{f''(x)}{2!}(2h)^2 + ... \tag{84}$$

From equations 85 and 81 we can derive:

$$f''(x) = \frac{f(x+2h) - 2f(x+h) + f(x)}{h^2} + O(h) \tag{85}$$

And now substituting this expression in equation 80 we can get a new estimate for the first order derivative:

$$f'(x) = \frac{-f(x+2h) + 4f(x+h) - 3f(x)}{2h} + O(h^2) \tag{86}$$

So we have approximations of the following approximations of the derivatives.

Table 4: Forward and Backward numerical approximations of first derivatives

	Error $O(h)$	**Error** $O(h^2)$
Forward	$\frac{f(x+h) - f(x)}{h}$	$\frac{3f(x) - 4f(x-h) + f(x-2h)}{2h}$
Backward	$\frac{f(x) - f(x-h)}{h}$	$\frac{f(x+h) - f(x-h)}{2h}$

The centered approximations of the derivatives give us a better estimation with smaller errors.

Table 5: Centered numerical approximations of first derivatives

	Error $O(h^2)$	**Error** $O(h^4)$
Centered	$\frac{f(x+h) - f(x-h)}{2h}$	$\frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h}$

The most accurate derivative we can compute has an error $O(h^4)$ because

$$\begin{aligned} \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h} = \\ = f'(x) - \frac{1}{30} f^{(5)}(x) h^4 + O(h^5) \end{aligned} \tag{87}$$

So the numerical approximation of the derivative will have an error of the order h^4 where h is the Lattice of the grid if we use this procedure. It can be applied easily to 3D dimensions and to the three components of B, because as the grid is regular and we only calculate first order derivatives respect one spatial direction we left the other two coordinates constant in the formula. So we can compute the first derivative of a function of three variables $f(x, y, z)$ respect y as:

$$\frac{\partial f}{\partial y} = \frac{-f(x, y+2h, z) + 8f(x, y+h, z) - 8f(x, y-h, z) + f(x, y-2h, z)}{12h} \tag{88}$$

Better estimates of the derivatives can be obtained with Richardson extrapolation but the grid spacing has to be reduced to half its length and this cannot be done in our grid estimates we see that if we calculate the derivatives numerically the error we introduce is of the order of h^4. The Lattice spacing used in the LHCb detector for example is $h = 10^{-1}$ m. That supposes an error of the order $\sim 10^{-4}$ for the estimated derivatives.

However, we see that reducing the grid spacing reduces considerably this error and as mentioned by Paul Probert in its paper the interpolation error using Maxwell Elements scales with the grid spacing as h^5 so if the grid lattice is reduced both errors (numerical derivatives and interpolation error) could be decreased enough.

B Code for section 6.1

We set the maximum power of the scalar potential expansion and we define the eight corners of the cube.

$n = 4$;

P = Import["Corners.dat", "Table"]

$\{\{-1,-1,-1\},\{-1,-1,1\},\{-1,1,-1\},\{-1,1,1\},\{1,-1,-1\},\{1,-1,1\},\{1,1,-1\},\{1,1,1\}\}$

Using the function Polyharm we calculate the sets of harmonic polynomials in x ,y, z and we arrange them in a list with sublists for the different sum of powers of x, y, z.

f[n_]:=Complement[PolyHarm[1, n, {x, y, z}, 1], PolyHarm[1, $n-1$, {x, y, z}, 1]];

HarmonicPolynomialsList:=Table[$f[i]$, {i, 0, n}];

We expand the potential as a sum of the Harmonic polynomials up to the 4th power.

Potential[x_, y_, z_]:=

Sum[(HarmonicPolynomialsList[[i, j]]) * (Subscript[a, $i-1$, j]),

{i, 1, $n+1$}, {j, 1, 2($i-1$) + 1}]

$a_{0,1}+xa_{1,1}+ya_{1,2}+za_{1,3}+xya_{2,1}+xza_{2,2}+yza_{2,3}+\left(x^2-z^2\right)a_{2,4}+\left(y^2-z^2\right)a_{2,5}+xyza_{3,1}+$
$\left(x^3-3xz^2\right)a_{3,2}+\left(xy^2-xz^2\right)a_{3,3}+\left(y^3-3yz^2\right)a_{3,4}+\left(x^2y-yz^2\right)a_{3,5}+\left(3x^2z-z^3\right)a_{3,6}+$
$\left(3y^2z-z^3\right)a_{3,7}+\left(x^3y-3xyz^2\right)a_{4,1}+\left(xy^3-3xyz^2\right)a_{4,2}+\left(x^3z-xz^3\right)a_{4,3}+\left(3xy^2z-xz^3\right)a_{4,4}+$
$\left(3x^2yz-yz^3\right)a_{4,5}+\left(y^3z-yz^3\right)a_{4,6}+\left(x^4-6x^2z^2+z^4\right)a_{4,7}+\left(y^4-6y^2z^2+z^4\right)a_{4,8}+\left(3x^2y^2-3x^2z^2-3y^2z^2+z\right.$

The parameters are indexed with two indexes. The first one indicates the power of the term in the expansion and the second one is for differentiating the $2n+1$ different parameters for each power n.

Consequently we form the magnetic field from the gradient of the potential.

{Bx, By, Bz} = Grad[Potential[x, y, z], {x, y, z}];

Bx

$a_{1,1}+ya_{2,1}+za_{2,2}+2xa_{2,4}+yza_{3,1}+\left(3x^2-3z^2\right)a_{3,2}+\left(y^2-z^2\right)a_{3,3}+2xya_{3,5}+6xza_{3,6}+$

$\left(3x^2y-3yz^2\right)a_{4,1}+\left(y^3-3yz^2\right)a_{4,2}+\left(3x^2z-z^3\right)a_{4,3}+\left(3y^2z-z^3\right)a_{4,4}+6xyza_{4,5}+\left(4x^3-12xz^2\right)a_{4,7}+$
$\left(6xy^2-6xz^2\right)a_{4,9}$

By

$a_{1,2}+xa_{2,1}+za_{2,3}+2ya_{2,5}+xza_{3,1}+2xya_{3,3}+\left(3y^2-3z^2\right)a_{3,4}+\left(x^2-z^2\right)a_{3,5}+6yza_{3,7}+$
$\left(x^3-3xz^2\right)a_{4,1}+\left(3xy^2-3xz^2\right)a_{4,2}+6xyza_{4,4}+\left(3x^2z-z^3\right)a_{4,5}+\left(3y^2z-z^3\right)a_{4,6}+\left(4y^3-12yz^2\right)a_{4,8}+$
$\left(6x^2y-6yz^2\right)a_{4,9}$

Bz

$a_{1,3}+xa_{2,2}+ya_{2,3}-2za_{2,4}-2za_{2,5}+xya_{3,1}-6xza_{3,2}-2xza_{3,3}-6yza_{3,4}-2yza_{3,5}+$
$\left(3x^2-3z^2\right)a_{3,6}+\left(3y^2-3z^2\right)a_{3,7}-6xyza_{4,1}-6xyza_{4,2}+\left(x^3-3xz^2\right)a_{4,3}+\left(3xy^2-3xz^2\right)a_{4,4}+$
$\left(3x^2y-3yz^2\right)a_{4,5}+\left(y^3-3yz^2\right)a_{4,6}+\left(-12x^2z+4z^3\right)a_{4,7}+\left(-12y^2z+4z^3\right)a_{4,8}+\left(-6x^2z-6y^2z+4z^3\right)a_{4,9}$

Create a list that contains the 24 equations. Note that we only save in that list one part of the equation, i.e. the evaluation of the three expansions of B in the 8 corners.

```
M = Dimensions[P][[1]];
EquationsList = ConstantArray[0, {M, 3}];

For[i = 1, i ≤ M, i++,
EquationsList[[i, 1]] = (Bx/.{x → Corners[[i, 1]], y → Corners[[i, 2]], z → Corners[[i, 3]]});
EquationsList[[i, 2]] = (By/.{x → Corners[[i, 1]], y → Corners[[i, 2]], z → Corners[[i, 3]]});
EquationsList[[i, 3]] = (Bz/.{x → Corners[[i, 1]], y → Corners[[i, 2]], z → Corners[[i, 3]]});
]
```

We define the 24×24 matrix of the system to be filled in taking the coefficients that multiply each of the 24 parameters in the equations described above.

```
M = ConstantArray[0, {24, 24}];
EquationsList = Flatten[EquationsList];
For[i = 1, i ≤ L, i++,
 aux1 = Delete[Normal@CoefficientArrays[EquationsList[[i]], ParametersList], 1];
M[[i]] = aux1[[1]];
Clear[aux1];
```

]

TableForm[*M*]

Now we calculate the determinant of that matrix to see if it is invertible. **Det**[*M*]

0

C Code for section 7.1

First we set the maximum power of the polynomial. In our case m=4. We can change this in the procedure and see how many free parameters are we left with a higher or lower degree. The result is always equivalent to the Harmonic potential condition.

Develop B_x as a sum of powers in x, y, and z up to a power m.

```
m = 4;
Bx[x_, y_, z_]:=Sum[If[i + j + k ≤ m,
Subscript[a, i, j, k] * x^(i) * y^(j) * z^(k), 0.], {k, 0, m}, {j, 0, m}, {i, 0, m}]
Bx[x, y, z]
```

$$a_{0,0,0}+za_{0,0,1}+z^2a_{0,0,2}+z^3a_{0,0,3}+z^4a_{0,0,4}+ya_{0,1,0}+yza_{0,1,1}+yz^2a_{0,1,2}+yz^3a_{0,1,3}+y^2a_{0,2,0}+y^2za_{0,2,1}+y^2z^2a_{0,2,2}+y^3a_{0,3,0}+y^3za_{0,3,1}+y^4a_{0,4,0}+xa_{1,0,0}+xza_{1,0,1}+xz^2a_{1,0,2}+xz^3a_{1,0,3}+xya_{1,1,0}+xyza_{1,1,1}+xyz^2a_{1,1,2}+xy^2a_{1,2,0}+xy^2za_{1,2,1}+xy^3a_{1,3,0}+x^2a_{2,0,0}+x^2za_{2,0,1}+x^2z^2a_{2,0,2}+x^2ya_{2,1,0}+x^2yza_{2,1,1}+x^2y^2a_{2,2,0}+x^3a_{3,0,0}+x^3za_{3,0,1}+x^3ya_{3,1,0}+x^4a_{4,0,0}$$

We repeat the same forB_y and B_z.

```
By[x_, y_, z_]:=Sum[If[i + j + k ≤ m,
Subscript[b, i, j, k] * x^(i) * y^(j) * z^(k), 0.], {k, 0, m}, {j, 0, m}, {i, 0, m}]
By[x, y, z]
```

$$b_{0,0,0}+zb_{0,0,1}+z^2b_{0,0,2}+z^3b_{0,0,3}+z^4b_{0,0,4}+yb_{0,1,0}+yzb_{0,1,1}+yz^2b_{0,1,2}+yz^3b_{0,1,3}+y^2b_{0,2,0}+y^2zb_{0,2,1}+y^2z^2b_{0,2,2}+y^3b_{0,3,0}+y^3zb_{0,3,1}+y^4b_{0,4,0}+xb_{1,0,0}+xzb_{1,0,1}+xz^2b_{1,0,2}+xz^3b_{1,0,3}+xyb_{1,1,0}+xyzb_{1,1,1}+xyz^2b_{1,1,2}+xy^2b_{1,2,0}+xy^2zb_{1,2,1}+xy^3b_{1,3,0}+x^2b_{2,0,0}+x^2zb_{2,0,1}+x^2z^2b_{2,0,2}+x^2yb_{2,1,0}+x^2yzb_{2,1,1}+x^2y^2b_{2,2,0}+x^3b_{3,0,0}+x^3zb_{3,0,1}+x^3yb_{3,1,0}+x^4b_{4,0,0}$$

```
Bz[x_, y_, z_]:=Sum[If[i + j + k ≤ m,
Subscript[c, i, j, k] * x^(i) * y^(j) * z^(k), 0.], {k, 0, m}, {j, 0, m}, {i, 0, m}]
Bz[x, y, z]
```

$$c_{0,0,0}+zc_{0,0,1}+z^2c_{0,0,2}+z^3c_{0,0,3}+z^4c_{0,0,4}+yc_{0,1,0}+yzc_{0,1,1}+yz^2c_{0,1,2}+yz^3c_{0,1,3}+y^2c_{0,2,0}+y^2zc_{0,2,1}+y^2z^2c_{0,2,2}+y^3c_{0,3,0}+y^3zc_{0,3,1}+y^4c_{0,4,0}+xc_{1,0,0}+xzc_{1,0,1}+xz^2c_{1,0,2}+xz^3c_{1,0,3}+xyc_{1,1,0}+xyzc_{1,1,1}+xyz^2c_{1,1,2}+xy^2c_{1,2,0}+xy^2zc_{1,2,1}+xy^3c_{1,3,0}+x^2c_{2,0,0}+x^2zc_{2,0,1}+x^2z^2c_{2,0,2}+x^2yc_{2,1,0}+x^2yzc_{2,1,1}+x^2y^2c_{2,2,0}+x^3c_{3,0,0}+x^3zc_{3,0,1}+x^3yc_{3,1,0}+x^4c_{4,0,0}$$

Next step is to define the magnetic field with its three components and calculate the divergence.

```
B = {Bx, By, Bz};

Div[{Bx[x, y, z], By[x, y, z], Bz[x, y, z]}, {x, y, z}];
```

Knowing that the divergence of B is equal to zero for each of the different powers of x,y and z we get the coefficients that multiply each of the different powers of x,y, and z in the divergence of B and we create a list which contains these coefficients. Each element of this list is equal to zero as we know $\nabla B = 0$. This list contains the set of equations imposed by the divergence.

```
List1 = CoefficientRules[Div[{Bx[x, y, z], By[x, y, z], Bz[x, y, z]}, {x, y, z}], {x, y, z}];

DivergenceEquations = ConstantArray[0, Length[List1]];

For[i = 1, i ≤ Length[List1], i++,

DivergenceEquations[[i]] = List1[[i, 2]]

]
```

Now we calculate the curl of the magnetic field

```
CurlB = Curl[{Bx[x, y, z], By[x, y, z], Bz[x, y, z]}, {x, y, z}]

List3 = CoefficientRules[Curl0[[2]], {x, y, z}];

EquationsCurl2 = ConstantArray[0, Length[List3]];

For[i = 1, i ≤ Length[List3], i++,

EquationsCurl2[[i]] = List3[[i, 2]]

]

List4 = CoefficientRules[Curl0[[3]], {x, y, z}];

EquationsCurl3 = ConstantArray[0, Length[List4]];

For[i = 1, i ≤ Length[List4], i++,

EquationsCurl3[[i]] = List4[[i, 2]]

]
```

Now we put together all these equations in a List

```
EquationsList = {DivergenceEquations, EquationsCurl1, EquationsCurl2, EquationsCurl3};
```

```
EquationsList = Flatten[EquationsList]

Length[EquationsList]
```

80

Now we construct the matrix of this system

```
MatrixSystem =

Normal[

CoefficientArrays [EquationsList, ParametersList[[2]]] ;
```

Reduce the matrix of this system to see how many independent equations we have

```
ReducedMatrix = RowReduce[MatrixSystem];

{L, M} = Dimensions[ReducedMatrix]
```

{80, 102}

We see that the dimensions of the matrix are 80 equationsx102 coefficients because the linear terms do not appear because when we make the derivatives they disappear

Let us now see which rows are rows of zeros and see how many independent equations we have.

```
ZerosTest = ConstantArray[0, 80];

For[i = 1, i ≤ Dimensions[ReducedMatrix][[1]], i++,

  ZerosTest[[i]] = Sum[Abs[ReducedMatrix[[i, j]]], {j, 1, 102}]

]

ZerosTest
```

10 rows are completely zeros what means that only the remaining rows are the independent equations.

D Harmonic Interpolation Script

```
(*Harmonic Interpolation only with B data(not derivatives)*)

Points=Import["Corners4.dat","Table"];
(* Import a data file that contains the coordinates of the
grid points used to interpolate*)
(*Lattice Constant*)

Lattice=1; (* Specify the Lattice of the grid*)

Corners=Lattice*Points;
(* Calculate the new coordinates of the
grid points for a specified Lattice *)

DataList=Import["BData.dat","Table"]
(* We take the measured values of the magnetic field
from a pre-defined file*)



Nn=Length[Corners];(*Number of grid points*)
L=3; (*3 components each element, only Bx By Bz*)
nphi=5;(* maximum degree of the potential expansion used to interpolate*)
(*We can choose that in our procedure,
 we can change the maximum degree of the polynomial to fit B depending
 on its nature*)
(* Use the function PolyHarm to calculate the sets of harmonic polynomials
in x,y,z*)

func[n_]:=Complement[PolyHarm[1,n,{x,y,z},1],PolyHarm[1,n-1,{x,y,z},1]];
sum=0;Grid[Table[{i,d=Dimensions[func[i]][[1]],sum+=d,func[i]},{i,1,nphi}],
Frame->All];

HarmonicPolynomialsList:=Table[func[i],{i,0,nphi}];
(*List with sublists arranging the HarmonicPolynomials
for the diferent sum of powers x,y,z *)

ParametersList=HarmonicPolynomialsList;
For[i=1,i<=nphi+1,i++,
For[j=1,j<=2(i-1)+1,j++,
ParametersList[[i,j]]=Subscript[a,i-1,j];
]
];
```

```
ParametersList=Delete[ParametersList,1];
ParametersList=Flatten[ParametersList];
(* Here we filled in with the coefficient (Subscript[a,i-1,j]) and remove
the first term that we do not need to expand the B field,
 the first constant only appears in the potential,
when we make the derivatives to calculate B it disappears*)

M=Length[Flatten[ParametersList]];(*number of coefficients( parameters),
here we put it variable depending on the maximum power we take*)

Potential[x_,y_,z_]:=Sum[(HarmonicPolynomialsList[[i,j]])*
(Subscript[a,i-1,j]),{i,1,nphi+1},{j,1,2(i-1)+1}];
(*Define the potential as an expansion in terms of the Harmonic
Polynomials and the parameters Subscript[a,i-1,j]*)

HarmonicPolynomialsListVector=Delete[HarmonicPolynomialsList,1];
HarmonicPolynomialsListVector=Flatten[HarmonicPolynomialsListVector];
(* Convert the list of Harmonic Polynomials to a vector*)

fElements=ConstantArray[0,{M,3}];

For[i=1,i<=M,i++,
fElements[[i,1]]=D[HarmonicPolynomialsListVector[[i]],x];
(* Bx as the derivative of the potential respect to x*)
fElements[[i,2]]=D[HarmonicPolynomialsListVector[[i]],y];
(*By as the derivative of the potential respect to y*)
fElements[[i,3]]=D[HarmonicPolynomialsListVector[[i]],z];
(*Bz as the derivative of the potential respect to z*)
];
(* Create the M vector functions from the gradient
of the Harmonic polynomials*)
numeration=ParametersList;

numeration={"Element Number",numeration};

numeration=Flatten[numeration];

TableOfElements1=Prepend[fElements,{"Bx","By","Bz"}];

TableOfElements2=MapThread[Prepend,{TableOfElements1,numeration }];

Print[Grid[TableOfElements2,Frame->All]];
(* Shows the harmonic polynomials used to interpolate *)

(*Construct the matrix A that has to be inverted to fit the function*)
A=ConstantArray[0,{M,M}];
(* Fill in the matrix A*)
```

```
For[u=1,u<=M,u++,
For[v=1,v<=M,v++,

A[[u,v]]= Sum[(((fElements[[u,l]])*(fElements[[v,l]]))/.
{x->Corners[[n,1]],y->Corners[[n,2]],z->Corners[[n,3]]}),
{l,1,L},{n,1,Nn}]
]
];
TableForm[A]
(* Show the matrix A*)

(*Construct the vector g*)
G=ConstantArray[0,{M}];

For[u=1,u<=M,u++,

For[l=1,l<=L,l++,
For[k=1,k<=Nn,k++,
aux1=Corners[[k,1]];
aux2=Corners[[k,2]];
aux3=Corners[[k,3]];
G[[u]]=G[[u]]+(fElements[[u,l]]/.{x->aux1,y->aux2,z->aux3})*(DataList[[k,l]]);
]
]
]
(* Show G*)
TableForm[G]

(* Invert A matrix*)

AInverse=Inverse[A];
TableForm[AInverse]
ParametersSolvedList=ConstantArray[0,M];

ParametersSolvedList=AInverse.G;
(* Get the parameters from the matrix inversion*)

(* Now, reconstruct the function we have interpolated*)

BInterpolated=ConstantArray[0,3];
(* Define a vector to be filled in
by the interpolated function*)

For[l=1,l<=L,l++,
BInterpolated[[l]]=Sum[fElements[[m,l]]*ParametersSolvedList[[m]],{m,1,M}];
(* for each of the 3 components of B
we sum the M elements(harmonic polynomials for phi) each one multiplied
```

```
by its parameter solved in the matrix inversion*)



Print["Bx=",BInterpolated[[1]]]
Print["By=",BInterpolated[[2]]]
Print["Bz=",BInterpolated[[3]]]
(* Shows the interpolated function *)

(* Let us check how good is our procedure computing the residuals
at the grid points used*)
Residuals=ConstantArray[0,{Nn,L}];

For[n=1,n<=Nn,n++,
For[l=1,l<=L,l++,
aux1=Corners[[n,1]];
aux2=Corners[[n,2]];
aux3=Corners[[n,3]];

aux4=BInterpolated[[l]]/.{x->aux1,y->aux2,z->aux3};

Residuals[[n,l]]=Abs[DataList[[n,l]]-aux4];
]
]

(* We can compute also the residual of the norm of B in each corner*)

NormResiduals=ConstantArray[0,Nn];
For[n=1,n<=Nn,n++,
NormResiduals[[n]]=Abs[DataList[[n,1]]^2+DataList[[n,2]]^2+DataList[[n,3]]^2-
((((BInterpolated[[1]])^2)+(BInterpolated[[2]]^2)+(BInterpolated[[3]]^2))/.
{x->Corners[[n,1]],y->Corners[[n,2]],z->Corners[[n,3]]})]
]
(* Calculates the residuals of the norm of B in the grid points *)
TableOfResiduals2=MapThread[Prepend,{Residuals,NormResiduals}];
TableForm[TableOfResiduals2,TableHeadings->
{Corners,{"Bx^2+By^2+Bz^2","Bx","By","Bz"}}]
(* Show the residuals in a table*)

Export["ResidualsHarmonicnoderivatives.dat",TableOfResiduals2,"Table"];
(* Save the residuals in a file*)


(* As we have calculated the difference between the data
and the interpolated function in the corners,
now we compute the relative error as the difference divided by the
"true" value that is the magnetic field data values imported from DataList-dat*)
```

```
RelativeError=ConstantArray[0,{Nn,L}];(* to assign the same dimensions*)


For[n=1,n<=Nn,n++,
For[l=1,l<=L,l++,
If[DataList[[n,l]]!=0,
RelativeError[[n,l]]=((Residuals[[n,l]])/(Abs[DataList[[n,l]]]))*100;
,RelativeError[[n,l]]="NoData"];
]
]


RelativeErrorNorm=ConstantArray[0,Nn];
For[n=1,n<=Nn,n++,
RelativeErrorNorm[[n]]=(NormResiduals)/(((DataList[[n,1]])^2)+
((DataList[[n,2]])^2)+((DataList[[n,3]])^2));
](* Calculate also the relative errors for the norm of B in the grid points*)

Print["Relative Errors"]
TableForm[RelativeError,TableHeadings->{Corners,{"Bx (%)","By(%)","Bz(%)"}}]

(* Show the relative errors *)



Export["RelativeErrorsOnlyBdata.dat",RelativeError,"Table"]

Sum[Residuals[[n,l]],{n,1,Nn},{l,1,L}]

AverageResidual=Sum[Residuals[[n,l]],{n,1,Nn},{l,1,L}]/(Nn*L)
(* sum of all the residuals divided by the number of values,
We call that the average residual*)

Relativex=ConstantArray[0,Length[Corners]];
Relativey=ConstantArray[0,Length[Corners]];
Relativez=ConstantArray[0,Length[Corners]];

For[i=1,i<=Length[Corners],i++,
Relativex[[i]]=RelativeError[[i,1]];
Relativey[[i]]=RelativeError[[i,2]];
Relativez[[i]]=RelativeError[[i,3]];
]
```

```
Histogram[{Relativex, Relativey, Relativez}]

Export["HarmonicResiduals.dat", Relativex, "Table"];
Export["HarmonicResiduals.dat", Relativey, "Table"];
Export["HarmonicResiduals.dat", Relativez, "Table"];

(* Create an histogram with all the relative errors  for the three components*)


Max[Residualsz]
Max[Residualsy]
Max[Residualsx]
(* See which are the minimum and maximum relative errors obtained *)
```

E Maxwell Elements Script

```
(* Code used to interpolate a set of given data by Bdata.dat
with the Maxwell Elements procedure*)
(* BData.dat contains values of B and its derivatives
at the corners of a cube *)

Nb=4;(*Up to which order we want to expand the magnetic field, maximum 4*)
M=Sum[2*(n)+1,{n,0,Nb+1}]-1;
(*number of elements we are going to use to interpolate, maximum is 35*)
(*The number of elements is a function of the maximum power of B*)

Nn=8;(*8 corner points for each element*)
L=8; (*8 components each element*)

(*Real data values at the 8 corner points*)
Corners=Import["Corners.dat","Table"];
(* We take the values of the coordinates of each of the eight corner points
from a pre-defined file*)
DataList=Import["BData.dat","Table"];
(* We take the values of magnetic field ( and its derivatives in this case)
from a pre-defined file,
in this case these values are generated with another function called
MagneticFieldGenerator.m *)

BGeneratedFunction=Import["BFunction.dat","Table"];
(* Get the expression of the function we have generated to create
the values we use to interpolate, i.e.
the function we interpolate with the 35 elements*)
(*Maxwell Elements Interpolation*)

(* We write the rules for the 35 elements directly*)

(*Constant Elements*)
(*Element nr1*)
EA=ConstantArray[0,M];
EB=ConstantArray[0,M];
EC=ConstantArray[0,M];

(* Define three lists that we fill in with the coefficient rules
that are used to construct the polynomial
for each element*)
EA[[1]]={{0,0,0}->1};
EB[[1]]={{0,0,0}->0};
EC[[1]]={{0,0,0}->0};
(*Element nr2*)
```

```
EA[[2]]={{0,0,0}->0};
EB[[2]]={{0,0,0}->1};
EC[[2]]={{0,0,0}->0};

(*Element nr3*)
EA[[3]]={{0,0,0}->0};
EB[[3]]={{0,0,0}->0};
EC[[3]]={{0,0,0}->1};
(*Linear Elements*)
(*Element nr4*)
EA[[4]]={{0,0,0}->0};
EB[[4]]={{0,0,1}->1};
EC[[4]]={{0,1,0}->1};

(*Element nr5*)
EA[[5]]={{0,0,1}->1};
EB[[5]]={{0,0,0}->0};
EC[[5]]={{1,0,0}->1};

(*Element nr6*)
EA[[6]]={{0,1,0}->1};
EB[[6]]={{1,0,0}->1};
EC[[6]]={{0,0,0}->0};

(*Element nr7*)
EA[[7]]={{1,0,0}->1};
EB[[7]]={{0,0,0}->0};
EC[[7]]={{0,0,1}->-1};

(*Element nr8*)
EA[[8]]={{1,0,0}->1};
EB[[8]]={{0,1,0}->-1};
EC[[8]]={{0,0,0}->0};
(*Quadratic Elements*)
(*Element nr9*)
EA[[9]]={{0,1,1}->1};
EB[[9]]={{1,0,1}->1};
EC[[9]]={{1,1,0}->1};

(*Element nr10*)
EA[[10]]={{0,2,0}->1,{2,0,0}->-1};
EB[[10]]={{1,1,0}->2};
EC[[10]]={{0,0,0}->0};

(*Element nr11*)
EA[[11]]={{0,0,2}->1,{2,0,0}->-1};
EB[[11]]={{0,0,0}->0};
```

```
EC[[11]]={{1,0,1}->2};

(*Element nr12*)
EA[[12]]={{0,0,0}->0};
EB[[12]]={{0,0,2}->1,{0,2,0}->-1};
EC[[12]]={{0,1,1}->2};

(*Element nr13*)
EA[[13]]={{1,1,0}->2};
EB[[13]]={{2,0,0}->1,{0,2,0}->-1};
EC[[13]]={{0,0,0}->0};

(*Element nr14*)
EA[[14]]={{0,0,0}->0};
EB[[14]]={{0,1,1}->2};
EC[[14]]={{0,2,0}->1,{0,0,2}->-1};

(*Element nr15*)
EA[[15]]={{1,0,1}->2};
EB[[15]]={{0,0,0}->0};
EC[[15]]={{2,0,0}->1,{0,0,2}->-1};
(*Cubic Elements*)
(*Element nr16*)
EA[[16]]={{1,1,1}->6};
EB[[16]]={{0,2,1}->-3,{2,0,1}->3};
EC[[16]]={{0,3,0}->-1,{2,1,0}->3};

(*Element nr17*)
EA[[17]]={{1,1,1}->6};
EB[[17]]={{0,0,3}->-1,{2,0,1}->3};
EC[[17]]={{2,1,0}->3,{0,1,2}->-3};

(*Element nr18*)
EA[[18]]={{2,1,0}->3,{0,1,2}->-3};
EB[[18]]={{3,0,0}->1,{1,0,2}->-3};
EC[[18]]={{1,1,1}->-6};

(*Element nr19*)
EA[[19]]={{0,3,0}->1,{0,1,2}->-3};
EB[[19]]={{1,2,0}->3,{1,0,2}->-3};
EC[[19]]={{1,1,1}->-6};

(*Element nr20*)
EA[[20]]={{3,0,0}->2,{1,2,0}->-3,{1,0,2}->-3};
EB[[20]]={{0,1,2}->3,{2,1,0}->-3};
EC[[20]]={{2,0,1}->-3,{0,2,1}->3};
```

```
(*Element nr21*)
EA[[21]]={{1,0,2}->3,{1,2,0}->-3};
EB[[21]]={{0,3,0}->2,{0,1,2}->-3,{2,1,0}->-3};
EC[[21]]={{2,0,1}->3,{0,2,1}->-3};

(*Element nr22*)
EA[[22]]={{1,2,0}->3,{1,0,2}->-3};
EB[[22]]={{2,1,0}->3,{0,1,2}->-3};
EC[[22]]={{0,0,3}->2,{2,0,1}->-3,{0,2,1}->-3};
(*Element nr23*)
EA[[23]]={{0,0,3}->1,{0,2,1}->-3};
EB[[23]]={{1,1,1}->-6};
EC[[23]]={{1,0,2}->3,{1,2,0}->-3};
(*Element nr24*)
EA[[24]]={{2,0,1}->-3,{0,2,1}->3};
EB[[24]]={{1,1,1}->6};
EC[[24]]={{3,0,0}->-1,{1,2,0}->3};
(*Quartic Elements*)
(*Element nr25*)
EA[[25]]={{4,0,0}->1,{2,0,2}->-3,{2,2,0}->-3,{0,2,2}->3};
EB[[25]]={{3,1,0}->-2,{1,1,2}->6};
EC[[25]]={{3,0,1}->-2,{1,2,1}->6};

(*Element nr26*)
EA[[26]]={{0,4,0}->1,{2,0,2}->3,{2,2,0}->-3,{0,2,2}->-3};
EB[[26]]={{3,1,0}->-2,{1,3,0}->4,{1,1,2}->-6};
EC[[26]]={{3,0,1}->2,{1,2,1}->-6};

(*Element nr27*)
EA[[27]]={{0,0,4}->1,{2,0,2}->-3,{2,2,0}->3,{0,2,2}->-3};
EB[[27]]={{3,1,0}->2,{1,1,2}->-6};
EC[[27]]={{3,0,1}->-2,{1,2,1}->-6,{1,0,3}->4};

(*Element nr28*)
EA[[28]]={{1,1,2}->-6,{3,1,0}->4,{1,3,0}->-2};
EB[[28]]={{4,0,0}->1,{2,0,2}->-3,{0,2,2}->3,{2,2,0}->-3};
EC[[28]]={{2,1,1}->-6,{0,3,1}->2};

(*Element nr29*)
EA[[29]]={{1,1,2}->6,{1,3,0}->-2};
EB[[29]]={{0,4,0}->1,{2,0,2}->3,{0,2,2}->-3,{2,2,0}->-3};
EC[[29]]={{2,1,1}->6,{0,3,1}->-2};
(*Element nr30*)
EA[[30]]={{1,1,2}->-6,{1,3,0}->2};
EB[[30]]={{0,0,4}->1,{2,0,2}->-3,{0,2,2}->-3,{2,2,0}->3};
EC[[30]]={{0,1,3}->4,{2,1,1}->-6,{0,3,1}->-2};
```

```
(*Element nr31*)
EA[[31]]={{1,0,3}->-2,{1,2,1}->-6,{3,0,1}->4};
EB[[31]]={{0,1,3}->2,{2,1,1}->-6};
EC[[31]]={{4,0,0}->1,{2,2,0}->-3,{2,0,2}->-3,{0,2,2}->3};

(*Element nr32*)
EA[[32]]={{1,0,3}->2,{1,2,1}->-6};
EB[[32]]={{0,1,3}->-2,{0,3,1}->4,{2,1,1}->-6};
EC[[32]]={{0,4,0}->1,{2,2,0}->-3,{2,0,2}->3,{0,2,2}->-3};

(*Element nr33*)
EA[[33]]={{1,0,3}->-2,{1,2,1}->6};
EB[[33]]={{0,1,3}->-2,{2,1,1}->6};
EC[[33]]={{0,0,4}->1,{2,2,0}->3,{2,0,2}->-3,{0,2,2}->-3};

(*Element nr34*)
EA[[34]]={{2,1,1}->3,{0,1,3}->-1};
EB[[34]]={{1,0,3}->-1,{3,0,1}->1};
EC[[34]]={{1,1,2}->-3,{3,1,0}->1};
(*Element nr35*)
EA[[35]]={{0,3,1}->1,{0,1,3}->-1};
EB[[35]]={{1,2,1}->3,{1,0,3}->-1};
EC[[35]]={{1,1,2}->-3,{1,3,0}->1};

(*Defined all the rules for the 35 elements, we construct a 35x8 matrix
called fElements*)
fElements=ConstantArray[0,{M,L}];
(*Each row of the fElements matrix correspond to the 8 values of the vector
Ym (m designes the number of the element),
so fElements[[m,l]] is equal to the component l of the element m
where l goes from 1 to 8 corresponding to
[Bx,By,Bz,d(Bx)/dx,d(Bx)/dy,d(Bx)/dz,d(By)/dy,d(By)/dz] *)
For[i=1,i<=M,i++,
fElements[[i,1]]=FromCoefficientRules[EA[[i]],{x,y,z}];
fElements[[i,2]]=FromCoefficientRules[EB[[i]],{x,y,z}];
fElements[[i,3]]=FromCoefficientRules[EC[[i]],{x,y,z}];
fElements[[i,4]]=D[FromCoefficientRules[EA[[i]],{x,y,z}],x];
fElements[[i,5]]=D[FromCoefficientRules[EA[[i]],{x,y,z}],y];
fElements[[i,6]]=D[FromCoefficientRules[EA[[i]],{x,y,z}],z];
fElements[[i,7]]=D[FromCoefficientRules[EB[[i]],{x,y,z}],y];
fElements[[i,8]]=D[FromCoefficientRules[EB[[i]],{x,y,z}],z];
];
(*Code to show a table with all the elements and its derivatives,
so we know the functions we are using to interpolate the magnetic
field*)
numeration=Range[1,M];
```

```
numeration={"Element Number",numeration};

numeration=Flatten[numeration];

TableOfElements1=
Prepend[fElements,{"Bx","By","Bz","d(Bx)/dx","d(Bx)/dy",
"d(Bx)/dz","d(By)/dy","d(By)/dz"}];

TableOfElements2=MapThread[Prepend,{TableOfElements1,numeration }];

Grid[TableOfElements2,Frame->All]
(*End of the Table Output*)
(*Construct the matrix A that has to be inverted to fit the function*)
A=ConstantArray[0,{M,M}];
For[u=1,u<=M,u++,
For[v=1,v<=M,v++,
A[[u,v]]= Sum[(fElements[[u,l]])*(fElements[[v,l]]),{l,1,L}]
]
];
TableForm[A]

(*Construct the matrix A that has to be inverted to fit the function*)
Aevaluated=ConstantArray[0,{M,M}];
(* Fill in the matrix A*)
For[u=1,u<=M,u++,
For[v=1,v<=M,v++,

Aevaluated[[u,v]]= Sum[A[[u,v]]/.
{x->Corners[[n,1]],y->Corners[[n,2]],z->Corners[[n,3]]},{n,1,Nn}]
]
];
TableForm[Aevaluated]
(* Show the matrix A*)

(*Construct the vector g*)
G=ConstantArray[0,{M}];

For[u=1,u<=M,u++,

For[l=1,l<=L,l++,
For[k=1,k<=Nn,k++,
aux1=Corners[[k,1]];
aux2=Corners[[k,2]];
aux3=Corners[[k,3]];
G[[u]]=G[[u]]+(fElements[[u,l]]/.{x->aux1,y->aux2,z->aux3})*(DataList[[k,l]]);
]
]
```

```
]
TableForm[G]


(* Invert A matrix*)

AMatrixInverse=Inverse[Aevaluated];

ElementsParametersSolved=ConstantArray[0,M];

ElementsParametersSolved=AMatrixInverse.G;
(* obtain the parameters for the 35 elements*)


(* Now, reconstruct the function we have interpolated*)

BInterpolated=ConstantArray[0,8];(* Define a vector to be filled
in by the interpolated function*)

For[l=1,l<=L,l++,
BInterpolated[[l]]=Sum[fElements[[m,l]]*ElementsParametersSolved[[m]],{m,1,M}];
(* for each of the 8 components of B( we also compute the 5
independent derivatives)
we sum the M elements each one multiplied by its parameter
solved in the matrix inversion*)


]

Print["Bx=",BInterpolated[[1]]]
Print["By=",BInterpolated[[2]]]
Print["Bz=",BInterpolated[[3]]]

(* Let us check how good is our procedure computing the residuals
at the 8 corners*)
Residuals=ConstantArray[0,{L,Nn}];

For[n=1,n<=Nn,n++,
For[l=1,l<=L,l++,
aux1=Corners[[n,1]];
aux2=Corners[[n,2]];
aux3=Corners[[n,3]];

aux4=BInterpolated[[l]]/.{x->aux1,y->aux2,z->aux3};

Residuals[[n,l]]=Abs[DataList[[n,l]]-aux4];
]
```

```
]
(* We can compute also the residual of the norm of B in each corner*)
NormResiduals=ConstantArray[0,8];
For[n=1,n<=Nn,n++,
NormResiduals[[n]]=DataList[[n,1]]^2+DataList[[n,2]]^2+DataList[[n,3]]^2-
((((BInterpolated[[1]])^2)+(BInterpolated[[2]]^2)+(BInterpolated[[3]]^2))/.
{x->Corners[[n,1]],y->Corners[[n,2]],z->Corners[[n,3]]})
]

TableOfResiduals2=MapThread[Prepend,{Residuals,NormResiduals}];
TableForm[TableOfResiduals2,TableHeadings->
{Corners,{"Bx^2+By^2+Bz^2","Bx","By","Bz","d(Bx)/dx",
"d(Bx)/dy","d(Bx)/dz","d(By)/dy","d(By)/dz"}}]
Export["ResidualsMaxwellElements.dat",TableOfResiduals2,"Table"]

VectorPlot3D[{BInterpolated[[1]],BInterpolated[[2]],BInterpolated[[3]]},
{x,-1,1},{y,-1,1},{z,-1,1}]
(*Plot the Interpolating function in the domain of interpolation ( unit cube)*)

(* As we have calculated the difference between the data and the
interpolated function in the corners
, now we compute the relative error as the difference divided by the "true"
value that we assume to be the magnetic field generated*)
RelativeError=ConstantArray[0,{Nn,L}];(* to assign the same dimensions*)



For[n=1,n<=Nn,n++,
For[l=1,l<=L,l++,
If[DataList[[n,l]]!=0,
RelativeError[[n,l]]=((Residuals[[n,l]])/(Abs[DataList[[n,l]]]))*100;
,RelativeError[[n,l]]="NoData"];
]
]

RelativeErrorNorm=ConstantArray[0,Nn];
For[n=1,n<=Nn,n++,
RelativeErrorNorm[[n]]=(NormResiduals)/(((DataList[[n,1]])^2)+
((DataList[[n,2]])^2)+((DataList[[n,3]])^2));
]

Print["Relative Errors"]
TableForm[RelativeError,TableHeadings->{Corners,{"Bx (%)","By(%)","Bz(%)"}}]

                Export["RelativeErrorsMaxwell.dat",RelativeError,"Table"]
```