# Calculating Magnetic Signatures using the Method of Moments in Julia

Lisette de Bruin

# Calculating Magnetic Signatures using the Method of Moments in Julia

by

Lisette de Bruin

to obtain the degree of Master of Science,
in Computational Science and Engineering,
at the Delft University of Technology,
to be defended publicly on Thursday August 1, 2024 at 13:00.

| | | |
|---|---|---|
| Student number: | 4856821 | |
| Project duration: | December 11, 2023 – August 1, 2024 | |
| Supervisors: | Dr. ir. D.J.P. Lahaye, | TU Delft |
| | Dr. ir. A.R.P.J. Vijn, | TU Delft |
| | Dr. ir. E.S.A.M. Lepelaars, | TNO |
| Thesis Committee: | Dr. ir. M. Verlaan, | TU Delft |
| | Dr. ir. N.V. Budko, | TU Delft |

*Cover image:* Gajbhiye, S. (2024, 7 May). Weak magnetic field may have set the stage for life on Earth. Earth.com.
https://www.earth.com/news/weak-magnetic-field-may-have-set-the-stage-for-life-on-earth/

TU Delft

# Preface

A project is never completed alone, and I would like to take this opportunity to thank the people who have supported me and have contributed to the completion of my thesis.
First of all, I would like to thank the committee members for dedicating their time to review and evaluate my thesis. I would like to express my gratitude to Domenico for his enthusiasm, his assistance in learning a new programming language, and his creative ideas. His support enabled me to gain understanding of Julia and successfully overcome numerous implementation challenges. Next, I would like to thank Aad for his guidance throughout the project. I am grateful for the honest, personal involvement, which has empowered me to complete the project smoothly. His support and flexibility, especially in the final phase, were really valuable. I am also thankful to Eugene for his guidance and sharp insights during our discussions. His detailed input and work were essential to raise this research to a higher level. It was a pleasure to have supervisors who focus on opportunities rather than problems, creating an inspiring environment for learning.
Finally, I would like to thank my friends and family for their encouragement and motivation.

*Lisette de Bruin*
*The Hague, July 2024*

# Abstract

Magnetostatics play a crucial role in the detection and localisation of naval vessels. Also, minimising a vessel's magnetic signature is essential to reduce the risk posed by naval mines, which often rely on magnetic detection. This research aims to improve the calculation of magnetic signatures using the Method of Moments (MoM) by implementing it in Julia, a high-performance programming language. A simplified version of TNO's current MATLAB-based approach is implemented in Julia to establish a baseline for the accuracy and efficiency. Linear basis functions and automatic differentiation (AD) are incorporated into the methodology to explore potential improvements. These extended methods are compared to the baseline to evaluate their performance.

Results show that Julia can be of great value, since it significantly improves the assembly time of the interaction matrix. Point matching is not a suitable approach when using linear basis functions. The Galerkin method shows promising results, though its computational performance remains a significant drawback. Also, using AD shows potential to simplify the implementation of the MoM by eliminating the need for analytical integral expressions. However, AD disappoints in terms of computational performance. Moreover, the AD implementation relies on a mesh-dependent parameter.

# Contents

# List of Figures

# List of Tables

# Nomenclature

## Symbols

| Symbol | Definition | Unit |
|---|---|---|
| $\mathbf{A}$ | Magnetic vector potential | [A·m] |
| $\mathbf{B}$ | Magnetic flux density | T |
| $\mathbf{B}_{\text{ext}}$ | External magnetic flux density | T |
| $\mathbf{B}_{\text{red}}$ | Reduced magnetic flux density | T |
| $\tilde{\mathbf{B}}_{\text{red}}$ | Approximated reduced magnetic flux density | T |
| $\mathbf{H}$ | Magnetic field | A/m |
| $\mathbf{H}_{\text{ext}}$ | External magnetic field | A/m |
| $\mathbf{H}_{\text{red}}$ | Reduced magnetic field | A/m |
| $\mathbf{J}$ | Electric current density | A/m$^2$ |
| $\mathbf{J}_b$ | Bound current density | A/m$^2$ |
| $\mathbf{J}_f$ | Electric current density of free charges | A/m$^2$ |
| $\mathbf{M}$ | Magnetisation | A/m |
| $\tilde{\mathbf{M}}$ | Approximated magnetisation | A/m |
| $\mathbf{u}$ | Unit vector | - |
| $\epsilon_{shift}$ | Parameter to quantify the extent of evaluation point shift | Dimensionless |
| $\chi_m$ | Magnetic susceptibility | - |
| $\mu_0$ | Permeability of free space | H/m |

# 1 Introduction

## 1.1. Research Motivation

Magnetic fields are widely used for detection and localization of naval vessels [3]. In 1920, Germany started the development of sea mines that would remain inactive until it detected the magnetic field of a target ship. When certain requirements have been met, it would detonate [p. 2][4]. Ships often operate in conflict areas where the threat of naval mines is high. Naval vessels are mainly constructed of steel, which leads to a disturbance in the Earth's magnetic field. The magnetic distortion field surrounding a naval vessel inside the Earth's magnetic field is called the magnetic signature [5]. Thus, minimizing the ship's magnetic signature reduces the risk of naval mines and being detected. One way to reduce the magnetic signature is using a degaussing system [6]. A degaussing system is a collection of coils installed onboard the vessel, where the current in each of the coil is controlled. When these coils are energised with proper currents, it generates a magnetic field opposite to the magnetic field of the vessel to reduce the magnetic signature [3]. To be able to determine the values of the coil currents, the vessel's magnetic signature has to be estimated as accurately as possible. The Method of Moments (MoM) is a well known method to describe the inter-coupling effects within a magnetic structure [6]. A description of the MoM can be found in [7], [8]. TNO implemented this method to model naval vessel's magnetic signature.
Currently, TNO's implementation of MoM is done in MATLAB. Due to long run times and big memory uses, it is worth to investigate alternative implementations and programming languages, such as Julia. Julia is a relatively new language for scientific computing. It is a dynamic language, which performs comparable to C/C++ and Fortran and is significantly faster than MATLAB and Python [9].

## 1.2. Research Goals

The primary goal of this research is to calculate magnetic signatures using the MoM in Julia, aiming for the highest accuracy and efficiency. Efficiency here means faster computation, less memory usage, and simpler implementation. The research question is defined to be as follows:

> *How can can TNO's current implementation of the assembly step in the MoM for modeling magnetic signatures be improved in terms of accuracy and efficiency using Julia?*

To be able to answer this research question, the following sub-questions are investigated:

1. What is the optimal implementation of a simplified version of the assembly step in TNO's current MoM in terms of accuracy, memory use, and computation speed using Julia?
2. What is the optimal choice of weighting functions using linear basis functions?
3. What is the effect on the accuracy of estimating magnetic signatures using linear basis functions?
4. How to use automatic differentiation such that the assembly step in the modeling process will be simplified?

The research is structured in three main phases, First, TNO's current approach will be implemented and tested to establish a baseline. This involves reproducing TNO's method and verifying its performance in Julia. Next, linear basis functions will be introduced to improve the accuracy of the magnetic signature estimations. Lastly, the focus will be on using automatic differentiation (AD) to make the computational process simpler. Moreover, the use of AD has the potential to lead to a general framework for the MoM in magnetostatics.

## 1.3. Outline

The research starts with an introductory chapter. This chapter provides an overview of the theory of magnetostatics. Thereafter, an introduction to the MoM is given in Chapter 3. The approach described in this chapter is comparable to TNO's current implementation. The method described in Chapter 3 is extended in Chapter 4 by introducing linear basis functions. Chapter 5 gives an introduction to the Julia programming language and elaborates on performance tips to optimise Julia code. In Chapter 6, an analysis is conducted between TNO's current implementation and the newly implemented MoM in Julia. This chapter also evaluates the new approaches in terms of computational speed and memory usage. The implementation of the different methods is verified in Chapter 7. Chapter 8 provides a brief introduction to AD and its relevance to the research. Chapter 9 presents the results of applying AD in the MoM. The implementation is compared to the previously described implementation that does not include AD. Finally, Chapter 10 concludes the research and gives recommendations for future work.

# 2 Magnetostatics

This research focuses on the theory of magnetic fields that are constant in time, also known as magnetostatics [10, p. 24]. This chapter introduces the fundamentals of magnetostatics.

## 2.1. Maxwell's Equations

In the early 1860s, James Clerk Maxwell laid the foundation for what are now known as Maxwell's equations. These four differential equations describe the relation and interaction of the magnetic induction field intensity $\mathbf{B}$ $[\mathrm{T}]$ and the electric field intensity $\mathbf{E}$ $[\mathrm{V/m}]$. The equations are given by [10, p. 3]

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0}, \tag{Gauss' law}$$

$$\nabla \cdot \mathbf{B} = 0, \tag{Gauss' law}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}, \tag{Faraday's law of induction}$$

$$\nabla \times \mathbf{B} = \mu_0 \left( \mathbf{J} + \epsilon_0 \frac{\partial \mathbf{E}}{\partial t} \right). \tag{Ampère's law}$$

Here, $\epsilon_0$ and $\mu_0$ are constants. $\epsilon_0$ denotes the electric constant $[\mathrm{F/m}]$ and $\mu_0$ denotes the permeability of free space $[\mathrm{H/m}]$. The electric charge density $[\mathrm{C/m^3}]$ is denoted by $\rho$ and the electric current density $[\mathrm{A/m^2}]$ is denoted by $\mathbf{J}$. In magnetostatic problems there is no time dependency [10, p. 24]. This implies that the time derivatives equal zero. The magnetostatic equations simplify to

$$\nabla \cdot \mathbf{B} = 0, \tag{2.1}$$

$$\nabla \times \mathbf{B} = \mu_0 \mathbf{J}. \tag{2.2}$$

### 2.1.1. Magnetisation

When a magnetic field is present, matter may become magnetised [11, p. 262]. Magnetisation is the magnetic dipole moment per unit volume of material and denoted by $\mathbf{M}$ $[\mathrm{A/m^2}]$ [10, p. 8]. The magnetisation contributes to a bound current $\mathbf{J}_b$ by

$$\mathbf{J}_b = \nabla \times \mathbf{M}. \tag{2.3}$$

In any situation, total current can be written as

$$\mathbf{J} = \mathbf{J}_f + \mathbf{J}_b, \tag{2.4}$$

where $\mathbf{J}_f$ is the electric current density of free charges [11, p. 279]. Substituting the expressions for the current into Equation 2.2 gives

$$\begin{aligned} \nabla \times \mathbf{B} &= \mu_0 \mathbf{J}, && (2.5)\\ &= \mu_0(\mathbf{J}_f + \mathbf{J}_b), && (2.6)\\ &= \mu_0(\mathbf{J}_f + \nabla \times \mathbf{M}). && (2.7) \end{aligned}$$

In order to obtain Ampère's law in terms of only free current, a new auxiliary field $\mathbf{H}$ $[\mathrm{A/m}]$ is defined as

$$\mathbf{H} = \frac{1}{\mu_0}\mathbf{B} - \mathbf{M}. \tag{2.8}$$

In terms of $\mathbf{H}$, Ampère's law is written as

$$\nabla \times \mathbf{H} = \mathbf{J}_f. \tag{2.9}$$

Assuming that there are no free currents present, the following equations hold

$$\nabla \cdot \mathbf{B} = 0, \tag{2.10}$$
$$\nabla \times \mathbf{H} = \mathbf{0}, \tag{2.11}$$
$$\mathbf{B} = \mu_0(\mathbf{H} + \mathbf{M}). \tag{2.12}$$

Observe that in free space, $\mathbf{M} = \mathbf{0}$ and Equation 2.12 reduces to

$$\mathbf{B} = \mu_0\mathbf{H}. \tag{2.13}$$

### 2.1.2. Constitutive Relations

Constitutive relations are used to describe the material properties of the media involved. The relations are also necessary to solve Equation 2.10 - 2.12. In this research, the material is linear, isotropic and homogeneous. A material is isotropic if its properties are the same in all directions. A material is homogeneous if its properties are the same at every point within the material [11, p. 189-190]. When an object of this material is placed in a uniform magnetic background field $\mathbf{B}_{\text{ext}} = \mu_0\mathbf{H}_{\text{ext}}$, the external field induces a magnetisation, $\mathbf{M}$. The induced magnetisation causes a disturbance in $\mathbf{H}$, known as the reduced magnetic field. The total magnetic field and total magnetic flux density equal

$$\mathbf{H} = \mathbf{H}_{\text{ext}} + \mathbf{H}_{\text{red}}, \tag{2.14}$$
$$\mathbf{B} = \mathbf{B}_{\text{ext}} + \mathbf{B}_{\text{red}}. \tag{2.15}$$

Linear, isotropic and homogeneous materials are characterised by a linear relation between the magnetisation and the magnetic flux density and reads

$$\mathbf{M} = \chi_m\mathbf{H}, \tag{2.16}$$

where $\chi_m$ is a dimensionless quantity known as the magnetic susceptibility [11, p. 285]. Substituting this into Equation 2.12, gives the following relation between $\mathbf{B}$ and $\mathbf{H}$

$$\mathbf{B} = \mu_0(\mathbf{H} + \mathbf{M}) = \mu_0(\mathbf{H} + \chi_m\mathbf{H}) = \mu_0(1 + \chi_m)\mathbf{H} = \mu\mathbf{H}. \tag{2.17}$$

The permeability of the material is described by $\mu = \mu_0(1 + \chi_m)$ and has unit $[\mathrm{H/m}]$ [12, p. 27]. Typical values for $\chi_m$ range from 1 to $10^5$.

### 2.1.3. Solution Based on a Vector Potential

Since $\mathbf{B}$ is a divergence-less field, the following theorem states that there exists a vector potential $\mathbf{A}$, such that $\mathbf{B} = \nabla \times \mathbf{A}$ [11, p. 54].

**Theorem 1** *The following conditions are equivalent:*

*(a)* $\nabla \cdot \mathbf{F} = 0$ *everywhere.*

*(b)* $\mathbf{F}$ *is the curl of some vector function:* $\mathbf{F} = \nabla \times \mathbf{A}$.

The solution of finding a vector potential $\mathbf{A}$ that corresponds to a given $\mathbf{B}$ is not unique. This is called the gauge freedom of $\mathbf{A}$ [13]. Suppose that $\mathbf{A}$ is a solution of $\nabla \times \mathbf{A} = \mathbf{B}$ and let $\phi$ be a scalar field. Then

$$\nabla \times (\mathbf{A} + \nabla\phi) = \nabla \times \mathbf{A} + \nabla \times \nabla\phi, \tag{2.18}$$
$$= \nabla \times \mathbf{A} + \mathbf{0}, \tag{2.19}$$
$$= \mathbf{B}. \tag{2.20}$$

As a result, $\mathbf{A} + \nabla\phi$ is also a solution of the problem. To obtain a unique solution for $\mathbf{A}$, the Coulomb gauge is used. This gauge assumes that the vector potential is also divergence free [13]

$$\nabla \cdot \mathbf{A} = 0. \tag{2.21}$$

From this assumption and Maxwell's equations for magnetostatic problems, the following can be derived

$$\begin{aligned}
\nabla \times \mathbf{B} &= \nabla \times (\nabla \times \mathbf{A}), && (2.22)\\
\nabla \times (\mu_0(\mathbf{H} + \mathbf{M})) &= \nabla(\nabla \cdot \mathbf{A}) - \nabla^2\mathbf{A}, && (2.23)\\
\nabla \cdot \mathbf{A} = 0 \quad \text{and} \quad \nabla \times \mathbf{H} = \mathbf{0} \Rightarrow \nabla \times \mu_0\mathbf{M} &= -\nabla^2\mathbf{A}, && (2.24)\\
\nabla^2\mathbf{A} &= -\mu_0\nabla \times \mathbf{M}. && (2.25)
\end{aligned}$$

Theorem 1 can be applied again on $\mathbf{A}$, because of the assumption $\nabla \cdot \mathbf{A} = 0$. Thus, there exists a vector function $\mathbf{C}$ such that $\mathbf{A} = \nabla \times \mathbf{C}$. As a result,

$$\begin{aligned}
\nabla^2\mathbf{A} &= -\mu_0\nabla \times \mathbf{M}, && (2.26)\\
\nabla^2\left(\nabla \times \mathbf{C}\right) &= -\mu_0\nabla \times \mathbf{M}, && (2.27)\\
\nabla^2\mathbf{C} &= -\mu_0\mathbf{M}. && (2.28)
\end{aligned}$$

This equation is known as Poisson's equation. A solution for $\mathbf{C}$ equals [11, p. 85]

$$\mathbf{C}(\mathbf{r}) = \frac{\mu_0}{4\pi}\iiint\limits_{\mathcal{V}_\infty} \frac{\mathbf{M}(\mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||}\,\mathrm{d}\mathbf{r}', \tag{2.29}$$

where $\mathcal{V}_\infty$ represents the entire 3D space. As a result, the vector potential can be written as

$$\begin{aligned}
\mathbf{A}(\mathbf{r}) &= \left(\nabla \times \mathbf{C}\right)(\mathbf{r}), && (2.30)\\
&= \frac{\mu_0}{4\pi}\nabla \times \iiint\limits_{\mathcal{V}_\infty} \frac{\mathbf{M}(\mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||}\,\mathrm{d}\mathbf{r}'. && (2.31)
\end{aligned}$$

The curl of the product of some scalar function $\phi$ and vector field $\mathbf{G}$ reads [11, p. 21]

$$\nabla \times (\phi\mathbf{G}) = \phi(\nabla \times \mathbf{G}) - \mathbf{G} \times \nabla\phi. \tag{2.32}$$

Using this identity, the following can be obtained

$$\nabla \times \left(\frac{\mathbf{M}(\mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||}\right) = \frac{1}{||\mathbf{r} - \mathbf{r}'||}\left(\nabla \times \mathbf{M}\right)(\mathbf{r}') - \mathbf{M}(\mathbf{r}') \times \nabla\frac{1}{||\mathbf{r} - \mathbf{r}'||}. \tag{2.33}$$

Since $\nabla$ operates on $\mathbf{r}$, $\left(\nabla \times \mathbf{M}\right)(\mathbf{r}') = 0$ for all $\mathbf{r}' \in \mathbb{R}^3$. As a result, the expression reads

$$\begin{aligned}
\nabla \times \left(\frac{\mathbf{M}(\mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||}\right) &= -\mathbf{M}(\mathbf{r}') \times \nabla\frac{1}{||\mathbf{r} - \mathbf{r}'||}, && (2.34)\\
\nabla\frac{1}{||\mathbf{r} - \mathbf{r}'||} = -\frac{\mathbf{r} - \mathbf{r}'}{||\mathbf{r} - \mathbf{r}'||^3} \Rightarrow \quad &= \frac{\mathbf{M}(\mathbf{r}') \times (\mathbf{r} - \mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||^3}. && (2.35)
\end{aligned}$$

Substituting this result into Equation 2.31 and using the fact that $\mathbf{B} = \nabla \times \mathbf{A}$, the magnetic flux density can be written as

$$\mathbf{B}(\mathbf{r}) = \frac{\mu_0}{4\pi}\nabla \times \iiint\limits_{\mathcal{V}_\infty} \frac{\mathbf{M}(\mathbf{r}') \times (\mathbf{r} - \mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||^3}\,\mathrm{d}\mathbf{r}'. \tag{2.36}$$

There is also another approach to find an expression for $\mathbf{B}$. Denote $\nabla'$ as the nabla operator acting on $\mathbf{r}'$. Using the identity from Equation 2.32, the result in Equation 2.31 can be written as

$$\nabla' \times \left(\frac{\mathbf{M}(\mathbf{r}')}{||\mathbf{r}-\mathbf{r}'||}\right) = \frac{1}{||\mathbf{r}-\mathbf{r}'||}\left(\nabla' \times \mathbf{M}\right)(\mathbf{r}') - \mathbf{M}(\mathbf{r}') \times \nabla' \frac{1}{||\mathbf{r}-\mathbf{r}'||}, \tag{2.37}$$

$$= \frac{\left(\nabla' \times \mathbf{M}\right)(\mathbf{r}')}{||\mathbf{r}-\mathbf{r}'||} + \mathbf{M}(\mathbf{r}') \times \nabla \frac{1}{||\mathbf{r}-\mathbf{r}'||}, \tag{2.38}$$

$$= \frac{\left(\nabla' \times \mathbf{M}\right)(\mathbf{r}')}{||\mathbf{r}-\mathbf{r}'||} - \nabla \times \left(\frac{\mathbf{M}(\mathbf{r}')}{||\mathbf{r}-\mathbf{r}'||}\right), \tag{2.39}$$

$$\nabla \times \left(\frac{\mathbf{M}(\mathbf{r}')}{||\mathbf{r}-\mathbf{r}'||}\right) = \frac{\left(\nabla' \times \mathbf{M}\right)(\mathbf{r}')}{||\mathbf{r}-\mathbf{r}'||} - \nabla' \times \left(\frac{\mathbf{M}(\mathbf{r}')}{||\mathbf{r}-\mathbf{r}'||}\right). \tag{2.40}$$

Hence, the following expression for $\mathbf{B}$ can be obtained

$$\mathbf{B}(\mathbf{r}) = \nabla \times \left(\frac{\mu_0}{4\pi} \iiint\limits_{\mathcal{V}_\infty} \frac{\left(\nabla' \times \mathbf{M}\right)(\mathbf{r}')}{||\mathbf{r}-\mathbf{r}'||} - \nabla' \times \left(\frac{\mathbf{M}(\mathbf{r}')}{||\mathbf{r}-\mathbf{r}'||}\right) \mathrm{d}\mathbf{r}'\right). \tag{2.41}$$

# 3 Derivation of Method of Moments for Magnetostatics

The MoM is a magnetisation-based formulation of the volume integral equations method for 3-D magnetostatics. This chapter shows how this method is derived, primarily based on [7]. Section 3.2 focuses on deriving the weak formulation. Consequently, two different sets of weighting functions are used to obtain linear systems. Thereafter, the focus is on deriving the interaction matrix and how the reduced magnetic flux density can be computed.

Consider a compact 3D-domain $\tau_M$ made of magnetic material exposed to an external magnetic flux density. As mentioned in Section 2.1.2, the total magnetic flux density at a point $\mathbf{r}$ equals

$$\mathbf{B}(\mathbf{r}) = \mathbf{B}_{\text{red}}(\mathbf{r}) + \mathbf{B}_{\text{ext}}(\mathbf{r}). \tag{3.1}$$

When estimating magnetic signatures, $\mathbf{B}_{\text{ext}}$ equals the Earth's magnetic flux density. Therefore, the external magnetic field in this study is assumed to be uniform in the vicinity of $\tau_M$. The derivation of the reduced magnetic flux density is given in Section 2.1.3 and is expressed in terms of the curl of the vector potential

$$\mathbf{B}_{\text{red}}(\mathbf{r}) = \frac{\mu_0}{4\pi}\nabla \times \iiint_{\tau_M} \frac{\mathbf{M}(\mathbf{r}') \times (\mathbf{r}-\mathbf{r}')}{||\mathbf{r}-\mathbf{r}'||^3}\,\mathrm{d}\mathbf{r}'. \tag{3.2}$$

## 3.1. Meshing the Domain

The first step in the MoM is to create a mesh by discretising the magnetisable domain. The process of meshing decomposes the domain $\tau_M \in \mathbb{R}^3$ in a finite number of non-overlapping, open polyhedra, $\tau_h$, $h = 1, \ldots, N$. Thus,

$$\tau_M = \bigcup_{h=1}^{N} \tau_h \tag{3.3}$$

This partition is called the mesh and the polyhedra are called the mesh elements.

## 3.2. Derivation of Weak Formulation

When pulse functions are used as basis functions, the assumption is made that the magnetisation is uniform in each element. The pulse functions are defined as [8, p. 45]

$$\phi^h(\mathbf{r}) = \begin{cases} 1 & \text{if} \quad \mathbf{r} \in \tau_h, \\ 0 & \text{elsewhere,} \end{cases} \qquad \text{for} \quad h = 1, \ldots, N. \tag{3.4}$$

The function for the estimated magnetisation of the object is now defined as

$$\tilde{\mathbf{M}}(\mathbf{r}) = \sum_{h=1}^{N} \tilde{\mathbf{M}}^h \phi_h(\mathbf{r}) \tag{3.5}$$

$$= \sum_{h=1}^{N} (\tilde{M}_x^h \mathbf{u}_x + \tilde{M}_y^h \mathbf{u}_y + \tilde{M}_z^h \mathbf{u}_z)\phi^h(\mathbf{r}) \tag{3.6}$$

Using this expression, the total flux density can be approximated at any point

$$\tilde{\mathbf{B}}(\mathbf{r}) = \mathbf{B}_{\text{red}}(\mathbf{r}) + \mathbf{B}_{\text{ext}}(\mathbf{r}), \tag{3.7}$$

$$= \nabla \times \left( \frac{\mu_0}{4\pi} \iiint_{\tau_M} \frac{\tilde{\mathbf{M}}(\mathbf{r}') \times (\mathbf{r} - \mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||^3} \, \mathrm{d}\mathbf{r}' \right) + \mathbf{B}_{\text{ext}}(\mathbf{r}), \tag{3.8}$$

$$= \nabla \times \sum_{h=1}^{N} \frac{\mu_0}{4\pi} \iiint_{\tau_h} \frac{\tilde{\mathbf{M}}(\mathbf{r}') \times (\mathbf{r} - \mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||^3} \, \mathrm{d}\mathbf{r}' + \mathbf{B}_{\text{ext}}(\mathbf{r}). \tag{3.9}$$

The vector potential produced by the estimated magnetisation of an arbitrary element $\tau_k$ is worked out below.

$$\frac{\mu_0}{4\pi} \iiint_{\tau_k} \frac{\tilde{\mathbf{M}}(\mathbf{r}') \times (\mathbf{r} - \mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||^3} \, \mathrm{d}\mathbf{r}' = \frac{\mu_0}{4\pi} \iiint_{\tau_k} \frac{(\tilde{M}_x^k \mathbf{u}_1 + \tilde{M}_y^k \mathbf{u}_2 + \tilde{M}_z^k \mathbf{u}_3) \times (\mathbf{r} - \mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||^3} \, \mathrm{d}\mathbf{r}', \tag{3.10}$$

$$= \frac{\mu_0}{4\pi} \left( \iiint_{\tau_k} \frac{\mathbf{u}_1 \times (\mathbf{r} - \mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||^3} \, \mathrm{d}\mathbf{r}' \tilde{M}_x^k + \iiint_{\tau_k} \frac{\mathbf{u}_2 \times (\mathbf{r} - \mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||^3} \, \mathrm{d}\mathbf{r}' \tilde{M}_y^k + \iiint_{\tau_k} \frac{\mathbf{u}_3 \times (\mathbf{r} - \mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||^3} \, \mathrm{d}\mathbf{r}' \tilde{M}_z^k \right), \tag{3.11}$$

$$= \sum_{i=1}^{3} \left( \frac{\mu_0}{4\pi} \iiint_{\tau_k} \frac{\mathbf{u}_i \times (\mathbf{r} - \mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||^3} \, \mathrm{d}\mathbf{r}' \right) \mathbf{u}_i^T \begin{pmatrix} \tilde{M}_x^k \\ \tilde{M}_y^k \\ \tilde{M}_z^k \end{pmatrix}. \tag{3.12}$$

Substituting this expression into Equation 3.9 gives

$$\tilde{\mathbf{B}}(\mathbf{r}) = \nabla \times \left[ \sum_{h=1}^{N} \sum_{i=1}^{3} \left( \frac{\mu_0}{4\pi} \iiint_{\tau_h} \frac{\mathbf{u}_i \times (\mathbf{r} - \mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||^3} \, \mathrm{d}\mathbf{r}' \right) \mathbf{u}_i^T \begin{pmatrix} \tilde{M}_x^h \\ \tilde{M}_y^h \\ \tilde{M}_z^h \end{pmatrix} \right] + \mathbf{B}_{\text{ext}}(\mathbf{r}), \tag{3.13}$$

$$= \sum_{h=1}^{N} \sum_{i=1}^{3} \nabla \times \left( \frac{\mu_0}{4\pi} \iiint_{\tau_h} \frac{\mathbf{u}_i \times (\mathbf{r} - \mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||^3} \, \mathrm{d}\mathbf{r}' \right) \mathbf{u}_i^T \begin{pmatrix} \tilde{M}_x^h \\ \tilde{M}_y^h \\ \tilde{M}_z^h \end{pmatrix} + \mathbf{B}_{\text{ext}}(\mathbf{r}), \tag{3.14}$$

$$= \sum_{h=1}^{N} [\mathbf{C}(\mathbf{r})]_h \tilde{\mathbf{M}}^h + \mathbf{B}_{\text{ext}}(\mathbf{r}), \tag{3.15}$$

$$\text{where} \quad [\mathbf{C}(\mathbf{r})]_h = \sum_{i=1}^{3} \nabla \times \left( \frac{\mu_0}{4\pi} \iiint_{\tau_h} \frac{\mathbf{u}_i \times (\mathbf{r} - \mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||^3} \, \mathrm{d}\mathbf{r}' \right) \mathbf{u}_i^T$$

The weak formulation now reads

$$\iiint_{\tau_M} w_k(\mathbf{r}) \left( \tilde{\mathbf{B}}(\mathbf{r}) - \sum_{h=1}^{N} [\mathbf{C}(\mathbf{r})]_h \tilde{\mathbf{M}}^h - \mathbf{B}_{\text{ext}}(\mathbf{r}) \right) \mathrm{d}\mathbf{r} = \mathbf{0} \quad \text{for} \quad k = 1, \ldots, N, \tag{3.16}$$

where $w_k$ is a proper scalar weighting function defined within each element $k$.

## 3.3. Point Matching

In TNO's current implementation, Equation 3.15 and the constitutive law of the material are satisfied at the centres of each element of the mesh. In other words, the centres of the elements are the evaluation points. This is equivalent to using Dirac delta functions centered at the centres of the elements as weighting functions. Thus the weighting functions can be defined as

$$w_k(\mathbf{r}) = \delta(\mathbf{r} - \mathbf{r}_k) \quad \text{for} \quad k = 1, \dots, N, \tag{3.17}$$

where $\mathbf{r}_k$ is the centre of element $k$. This formulation is known as point matching formulation and is widely mentioned in literature [14, p. 158] [8] [15]. The Dirac delta function satisfies the following relation [16]

$$\iiint\limits_{\tau_M} f(\mathbf{r})\delta(\mathbf{r} - \mathbf{r}_k)\,\mathrm{d}\mathbf{r} = \begin{cases} f(\mathbf{r}_k) & \text{if} \quad \mathbf{r}_k \in \tau_M, \\ 0 & \text{otherwise.} \end{cases} \tag{3.18}$$

As a result, the integral of the weak formulation becomes trivial and there is no integral over the range of the weighting function required [17]. The weak formulation becomes

$$\iiint\limits_{\tau_M} \delta(\mathbf{r} - \mathbf{r}_k) \left( \tilde{\mathbf{B}}(\mathbf{r}) - \sum_{h=1}^{N} [\mathbf{C}(\mathbf{r})]_h \tilde{\mathbf{M}}^h - \mathbf{B}_{\text{ext}}(\mathbf{r}) \right) \mathrm{d}\mathbf{r} = \mathbf{0}, \tag{3.19}$$

$$\tilde{\mathbf{B}}^k = \sum_{h=1}^{N} [\mathbf{C}_k]_h \tilde{\mathbf{M}}^h + \mathbf{B}^k_{\text{ext}}, \tag{3.20}$$

where $[\mathbf{C}_k]_h$ equals $[\mathbf{C}(\mathbf{r}_k)]_h$ and $\mathbf{B}^k_{\text{ext}} = \mathbf{B}_{\text{ext}}(\mathbf{r_k})$. The assumption is made that the magnetisation of the element is related to the magnetic flux density at the centre of the element. Assembling for all $N$ elements, the system of the discretised problem equals

$$\tilde{\mathbf{B}} = [\mathbf{C}]\tilde{\mathbf{M}} + \mathbf{B}_{\text{ext}}, \tag{3.21}$$

where the matrix $\mathbf{C}$, and vectors $\tilde{\mathbf{M}}$ and $\mathbf{B}_{\text{ext}}$ are as follows

$$[\mathbf{C}] = \begin{pmatrix} [\mathbf{C}(\mathbf{r}_1)]_1 & [\mathbf{C}(\mathbf{r}_1)]_2 & \dots & [\mathbf{C}(\mathbf{r}_1)]_N \\ [\mathbf{C}(\mathbf{r}_2)]_1 & [\mathbf{C}(\mathbf{r}_2)]_2 & \dots & [\mathbf{C}(\mathbf{r}_2)]_N \\ \vdots & \vdots & \dots & \vdots \\ [\mathbf{C}(\mathbf{r}_N)]_1 & [\mathbf{C}(\mathbf{r}_N)]_2 & \dots & [\mathbf{C}(\mathbf{r}_N)]_N \end{pmatrix}, \quad \tilde{\mathbf{M}} = \begin{pmatrix} \tilde{\mathbf{M}}^1 \\ \tilde{\mathbf{M}}^2 \\ \vdots \\ \tilde{\mathbf{M}}^N \end{pmatrix}, \quad \mathbf{B}_{\text{ext}} = \begin{pmatrix} \mathbf{B}^1_{\text{ext}} \\ \mathbf{B}^2_{\text{ext}} \\ \vdots \\ \mathbf{B}^N_{\text{ext}} \end{pmatrix}. \tag{3.22}$$

Note that $[\mathbf{C}]$ is a $3N \times 3N$ matrix, where $N$ is the number of mesh elements. From this point forward, $[\mathbf{C}]$ will be referred to as the interaction matrix. Since $\tilde{\mathbf{B}} = \mu_0(\tilde{\mathbf{H}} + \tilde{\mathbf{M}})$ and $\tilde{\mathbf{M}} = \chi_m \tilde{\mathbf{H}}$, a linear system is obtained

$$\tilde{\mathbf{B}} = [\mathbf{C}]\tilde{\mathbf{M}} + \mathbf{B}_{\text{ext}}, \tag{3.23}$$

$$\mu_0 \left( \tilde{\mathbf{H}} + \tilde{\mathbf{M}} \right) = [\mathbf{C}]\tilde{\mathbf{M}} + \mathbf{B}_{\text{ext}}, \tag{3.24}$$

$$\mu_0 \left( \frac{1}{\chi_m} \tilde{\mathbf{M}} + \tilde{\mathbf{M}} \right) = [\mathbf{C}]\tilde{\mathbf{M}} + \mathbf{B}_{\text{ext}}, \tag{3.25}$$

$$\tilde{\mathbf{M}} = \frac{\chi_m}{\mu_0(1+\chi_m)}[\mathbf{C}]\tilde{\mathbf{M}} + \frac{\chi_m}{\mu_0(1+\chi_m)}\mathbf{B}_{\text{ext}}, \tag{3.26}$$

$$\left( [\mathbf{I_N}] - \frac{\chi_m}{\mu_0(1+\chi_m)}[\mathbf{C}] \right) \tilde{\mathbf{M}} = \frac{\chi_m}{\mu_0(1+\chi_m)}\mathbf{B}_{\text{ext}}, \tag{3.27}$$

$$[\mathbf{A}(\chi_m)]\tilde{\mathbf{M}} = \frac{\chi_m}{\mu_0(1+\chi_m)}\mathbf{B}_{\text{ext}}, \tag{3.28}$$

where $[\mathbf{A}(\chi_m)] = [\mathbf{I_N}] - \frac{\chi_m}{\mu_0(1+\chi_m)}[\mathbf{C}]$. Observe that $[\mathbf{C}]$ is a non-symmetric matrix for finite discretisations, which implies that $[\mathbf{A}(\chi_m)]$ is also a non-symmetric matrix for finite discretisations.

## 3.4. Average Formulation

Alternatively, the weighting functions can be chosen as element-wise uniform functions, defined by the reciprocal of the element's volume. Taking the average represents a Galerkin formulation of the MoM [18]. The weighting functions are mathematically expressed as

$$w_k(\mathbf{r}) = \begin{cases} \frac{1}{V_k} & \text{if} \quad \mathbf{r} \in \tau_k, \\ 0 & \text{otherwise}, \end{cases} \qquad \text{for} \quad k = 1, \dots, N, \tag{3.29}$$

where $V_k$ equals the volume of element $\tau_k$. Substituting this weighting function for an arbitrary $k$ into Equation 3.16 results

$$\iiint\limits_{\tau_k} \frac{1}{V_k} \left( \tilde{\mathbf{B}}(\mathbf{r}) - \sum_{h=1}^{N} [\mathbf{C}(\mathbf{r})]_h \tilde{\mathbf{M}}^h - \mathbf{B}_{\text{ext}}(\mathbf{r}) \right) \mathrm{d}\mathbf{r} = \mathbf{0}, \tag{3.30}$$

$$\langle \tilde{\mathbf{B}} \rangle^k = \sum_{h=1}^{N} [\langle \mathbf{C} \rangle_k]_h \tilde{\mathbf{M}}^h + \langle \mathbf{B}_{\text{ext}} \rangle^k, \tag{3.31}$$

where $\langle \tilde{\mathbf{B}} \rangle^k$ is the average of the total magnetic flux density in element $k$, $\langle \mathbf{B}_{\text{ext}} \rangle^k$ is the average magnetic flux density of the external field, and $[\langle \mathbf{C} \rangle_k]_h$ is defined as

$$[\langle \mathbf{C} \rangle_k]_h = \frac{1}{V_k} \iiint\limits_{\tau_k} [\mathbf{C}(\mathbf{r})]_h \, \mathrm{d}\mathbf{r}. \tag{3.32}$$

Now, the assumption is made that the magnetisation within the element is related to the average magnetic flux density within the element. Assembling for all $N$ elements, the system of the discretised problem is obtained

$$\tilde{\mathbf{B}} = [\langle \mathbf{C} \rangle] \tilde{\mathbf{M}} + \langle \mathbf{B}_{\text{ext}} \rangle, \tag{3.33}$$

where the matrix $[\langle \mathbf{C} \rangle]$, and vectors $\tilde{\mathbf{M}}$ and $\langle \mathbf{B}_{\text{ext}} \rangle$ are as follows

$$[\langle \mathbf{C} \rangle] = \begin{pmatrix} [\langle \mathbf{C}(\mathbf{r}) \rangle_1]_1 & [\langle \mathbf{C}(\mathbf{r}) \rangle_1]_2 & \cdots & [\langle \mathbf{C}(\mathbf{r}) \rangle_1]_N \\ [\langle \mathbf{C}(\mathbf{r}) \rangle_2]_1 & [\langle \mathbf{C}(\mathbf{r}) \rangle_2]_2 & \cdots & [\langle \mathbf{C}(\mathbf{r}) \rangle_2]_N \\ \vdots & \vdots & \cdots & \vdots \\ [\langle \mathbf{C}(\mathbf{r}) \rangle_N]_1 & [\langle \mathbf{C}(\mathbf{r}) \rangle_N]_2 & \cdots & [\langle \mathbf{C}(\mathbf{r}) \rangle_N]_N \end{pmatrix}, \quad \tilde{\mathbf{M}} = \begin{pmatrix} \tilde{\mathbf{M}}^1 \\ \tilde{\mathbf{M}}^2 \\ \vdots \\ \tilde{\mathbf{M}}^N \end{pmatrix}, \quad \langle \mathbf{B}_{\text{ext}} \rangle = \begin{pmatrix} \langle \mathbf{B}_{\text{ext}} \rangle^1 \\ \langle \mathbf{B}_{\text{ext}} \rangle^2 \\ \vdots \\ \langle \mathbf{B}_{\text{ext}} \rangle^N \end{pmatrix} \tag{3.34}$$

Again, the $3N \times 3N$ matrix $[\langle \mathbf{C} \rangle]$ is not symmetric for finite discretisations.

## 3.5. Derivation of Interaction Matrix

For both point matching and the average formulation a function for the magnetic flux density reduced by a uniformly magnetised prism has to be found. Details on the calculations can be found in Appendix B. Suppose the uniform magnetisation of element $\tau$ is denoted by $\tilde{\mathbf{M}}$. As mentioned in Chapter 2, the magnetic vector potential produced can be calculated as follows

$$\mathbf{A}(\mathbf{r}) = \frac{\mu_0}{4\pi} \iiint\limits_{\tau} \frac{\tilde{\mathbf{M}}^k \times (\mathbf{r} - \mathbf{r}')}{||\mathbf{r} - \mathbf{r}||^3} \, \mathrm{d}\mathbf{r}'. \tag{3.35}$$

Since the assumption is made that $\tilde{\mathbf{M}}$ is uniform, $\tilde{\mathbf{M}}$ can be taken out of the integral, which gives

$$\mathbf{A}(\mathbf{r}) = \frac{\mu_0}{4\pi} \tilde{\mathbf{M}} \times \iiint\limits_{\tau} \nabla' \frac{1}{||\mathbf{r} - \mathbf{r}||} \, \mathrm{d}\mathbf{r}'. \tag{3.36}$$

By applying Gauss' theorem, the integral over the volume can be written as

$$\mathbf{A}(\mathbf{r}) = \frac{\mu_0}{4\pi} \sum_{S_f \in \partial\tau} \tilde{\mathbf{M}} \times \mathbf{n_f} W_f(\mathbf{r}), \tag{3.37}$$

where $\partial\tau$ is the boundary of $\tau$, $S_f$ is any of its faces, $\mathbf{n_f}$ is the outward normal unit vector of face $S_f$, and the function $W_f(\mathbf{r})$ is given by

$$W_f(\mathbf{r}) = \iint\limits_{S_f} \frac{1}{||\mathbf{r}-\mathbf{r}'||}\,\mathrm{d}\mathbf{r}'. \tag{3.38}$$

To obtain the induced magnetic flux density, the curl of Equation 3.37 is taken. This results in

$$\tilde{\mathbf{B}}_{\text{red}}(\mathbf{r}) = -\frac{\mu_0}{4\pi}\sum_{S_f\in\partial\tau}(\tilde{\mathbf{M}}\times\mathbf{n_f})\times\nabla W_f(\mathbf{r}). \tag{3.39}$$

The gradient in the above expression can be found by using vector identities and Stokes' theorem and is defined as

$$\nabla W_f(\mathbf{r}) = \sum_{l_e\in\partial S_f}\mathbf{n_f}\times\mathbf{t_e}w_e(\mathbf{r}) + \mathbf{n_f}\Omega_f(\mathbf{r}). \tag{3.40}$$

where $\mathbf{t_e}$ is the unit vector tangent to edge $l_e$ oriented accordingly by means of the right-hand rule. The function $\Omega_f(\mathbf{r})$ is the solid angle subtended by the face $S_f$, as seen from point $\mathbf{r}$. When $S_f$ is a triangular face, the solid angle can be calculated as follows

$$\Omega_f(\mathbf{r}) = 2\,\mathrm{arctan2}\,\frac{(\mathbf{r}_1-\mathbf{r})\cdot(\mathbf{r}_2-\mathbf{r})\times(\mathbf{r_3}-\mathbf{r})}{D(\mathbf{r})}, \tag{3.41}$$

$$\text{with}\quad D(\mathbf{r}) = ||\mathbf{r}_1-\mathbf{r}||\,||\mathbf{r}_2-\mathbf{r}||\,||\mathbf{r_3}-\mathbf{r}|| + ||\mathbf{r}_1-\mathbf{r}||(\mathbf{r}_2-\mathbf{r})\cdot(\mathbf{r_3}-\mathbf{r}) \\ + ||\mathbf{r}_2-\mathbf{r}||(\mathbf{r}_1-\mathbf{r})\cdot(\mathbf{r_3}-\mathbf{r}) + ||\mathbf{r_3}-\mathbf{r}||(\mathbf{r}_1-\mathbf{r})\cdot(\mathbf{r}_2-\mathbf{r}). \tag{3.42}$$

The function $w_e(\mathbf{r})$ is given by

$$w_e(\mathbf{r}) = \int_{l_e}\frac{1}{||\mathbf{r}-\mathbf{r}'||}\,\mathrm{d}r', \tag{3.43}$$

When the endpoints of the edge $l_e$ are defined as $\mathbf{r}_1$ and $\mathbf{r}_2$, the solution of this integral is as follows

$$w_e(\mathbf{r}) = \ln\frac{||\mathbf{r}_2-\mathbf{r}||+||\mathbf{r}_1-\mathbf{r}||+||\mathbf{r}_2-\mathbf{r}_1||}{||\mathbf{r}_2-\mathbf{r}||+||\mathbf{r}_1-\mathbf{r}||-||\mathbf{r}_2-\mathbf{r}_1||} \tag{3.44}$$

Note that both expressions for $\Omega_f$ and $w_e$ are singular for specific values of $\mathbf{r}$. The solid angle becomes singular when $\mathbf{r}$ coincides with one of the vertices of the face. Similarly, the function $w_e$ becomes singular when $\mathbf{r}$ lies on the edge $l_e$. The singularities do not lead to issues for point matching, as the functions are evaluated at the centres of the elements. However, when using the average formulation, the singularities may become problematic. For computing $[\langle[\mathbf{C}]\rangle_k]_h$, the average flux density within element $k$ produced by uniform magnetisation in element $h$, integration over the volume of element $k$ is done. This integration is done numerically. When $h = k$, the numerical integration must be done in a way that avoids evaluating the integrand at the boundary edges to prevent any singularities. The following section elaborates on the implementation of the numerical integration using Julia.

The derivations of the sub-blocks are independent of each other for both the point matching and the average formulation. Therefore, the process can be speed up by computing the sub-blocks of the interaction matrix in parallel. This is done with help of the package `Polyester.jl`. Details how this package is used, can be found in Chapter 5.

## 3.5.1. Numerical Integration in Julia

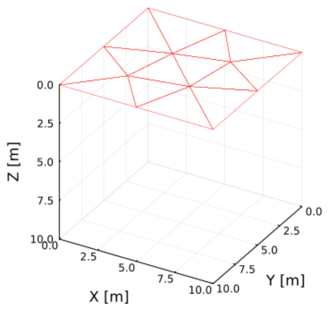
The integration is performed numerically with help of the Julia package `HCubature.jl`. This package is designed for "h-adaptive" multidimensional numerical integration, and is based on the algorithm described by A.C. Genz and A.A. Malikin [19]. The `hcubature()` function calculates the integral by continuously subdividing the integration domain into smaller sections in an adaptive manner until it reaches convergence. The function ensures that the integrand is never evaluated at the boundaries of the integration domain. As a result, it is possible to calculate the integral of a functions that have singularities at the boundaries. However, it is important to note that this approach may lead to slow convergence. Among other variables, users can specify the relative tolerances [20]. A good choice of the relative tolerance depends on the mesh. The default value equals the square root of the precision of the input arguments [20]. If this default setting results in `NaN` values, the relative tolerance is adjusted to the smallest possible value that avoids `NaN` values and does not lead to a memory error.

## 3.6. Obtain $\tilde{\mathbf{B}}_{\mathbf{red}}$

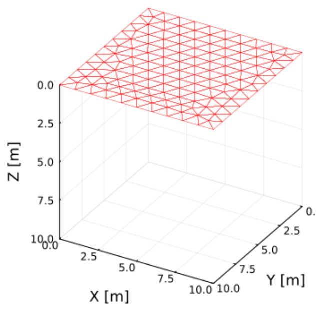
With help of the approximated magnetisation $\tilde{\mathbf{M}}$, an estimation for the reduced magnetic flux density can be calculated at any point $\mathbf{r}$. The reduced magnetic flux density by an arbitrary element $k$ is calculated by substituting $\tilde{\mathbf{M}}_{\mathbf{k}}$ in Equation 3.39. By taking the sum over all the elements, the total magnetic flux density in point $\mathbf{r}$ can be found.

## 3.7. Visualisation of $\tilde{\mathbf{M}}$ and $\tilde{\mathbf{B}}_{\mathbf{red}}$

In this section, the magnetisation and reduced magnetic flux density of a linearly reacting plate of $10 \times 10 \times 0.002\ \mathrm{m}$, with magnetic susceptibility $\chi_m = 100$, is visualised. Suppose the plate is placed in an external uniform background field of $50\ \mathrm{\mu T}$, which points in the positive x-direction. To create a 2D mesh, Gmsh is used. Thereafter, the mesh is extruded by $0.001\ \mathrm{m}$ in both the positive and negative z-direction. Thus, the mesh consists of a single layer of elements in the z-direction. Gmsh is an open-source three-dimensional finite element mesh generator designed to be fast, light, and user-friendly [21]. When a point of the geometry is defined, there is an optional parameter $lc$. This parameter defines the average edge length around that point [22]. By decreasing this parameter, the size of the mesh elements decreases and the mesh gets more refined. In the figure below, the mesh is visualised for two different values of $lc$.



**(a)** $lc = 5$.



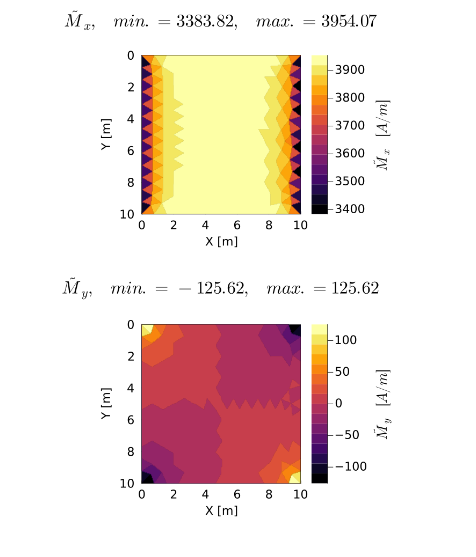
**(b)** $lc = 1$.

**Figure 3.1:** Visualisations of the mesh of the plate for two different values of $lc$.

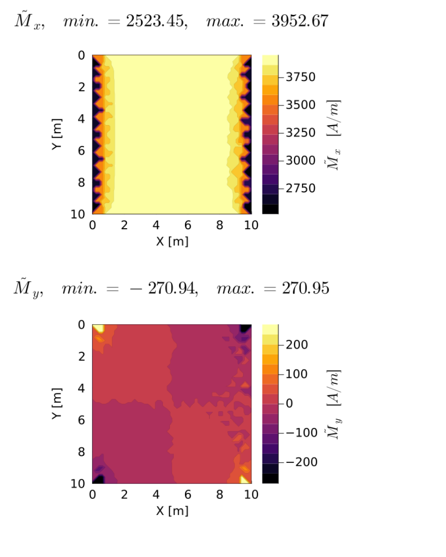
The parameter $lc$ is set to $0.8$. This results in a mesh consisting of 402 mesh elements and 228 nodes. In the average formulation, the relative tolerance in the function `hcubature()` is set to $10^{-6}$.

### 3.7.1. Visualisation of $\tilde{\mathbf{M}}$

In this subsection, the magnetisation at $z = 0\ \mathrm{m}$ within the plate is visualised through a top-down view. For this geometry and background field, the x-component of the magnetisation is expected to be symmetric in $x = 5\ \mathrm{m}$ and $y = 5\ \mathrm{m}$. The y-component of the magnetisation is expected to show 180-degree rotational symmetry x-y plane. Additionally, the y-component of the magnetisation is expected to be positive in the top-left and bottom-right regions of the plate, and negative in the remaining regions. The magnetisation in the x- and y-direction are visualised in separate figures. The z-component of the magnetisation is not shown because it is approximately zero throughout the plate, as expected. A total of 200 points are uniformly distributed along both the x- and y-axes, resulting in 40,000 points. The minimum and maximum value of the magnetisation at these points is calculated and displayed above the visualisations.



**(a)** Approximated magnetisation using point matching.



**(b)** Approximated magnetisation using the average formulation.

**Figure 3.2:** The approximated magnetisation using uniform basis functions within the plate.

From these figures, it can be concluded that the assumption of uniform magnetisation within each mesh element is made. The magnetisation within the plate shows a tile-like structure. By comparing the two figures, it can be concluded that the magnetisation in the x-direction is lower near the boundaries of the plate when using the average formulation compared to point matching. Also, the absolute minimum and maximum values of the y-component of the magnetisation, occurring at the corners of the plate, are higher with the average formulation than with point matching. Figure 3.2b shows an unexpected pattern in $\tilde{M}_y$ near the right boundary of the plate.

### 3.7.2. Visualisation of $\tilde{\mathrm{B}}_{\mathrm{red}}$

In this subsection, the reduced magnetic flux density due to the magnetisation of the plate is visualised. The reduced magnetic flux density is calculated in the x-y plane at $z = 5\ \mathrm{m}$. Both $x$ and $y$ range from $-10\ \mathrm{m}$ to $20\ \mathrm{m}$. 40 points are uniformly distributed along both the x- and y-direction, which results in a total of 1600 points. The points are visualised in the figure below.

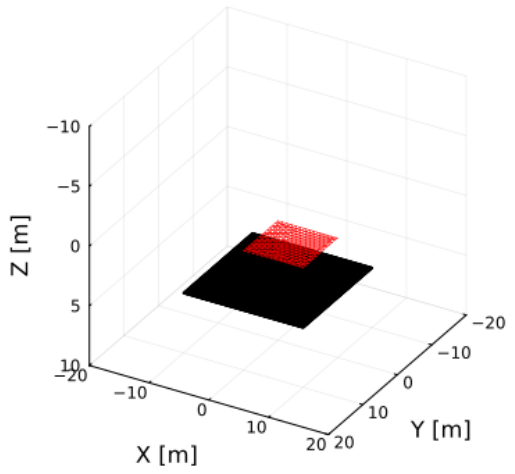
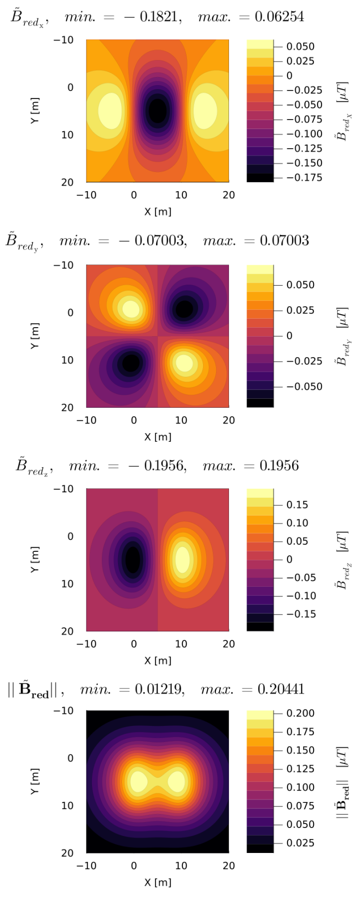
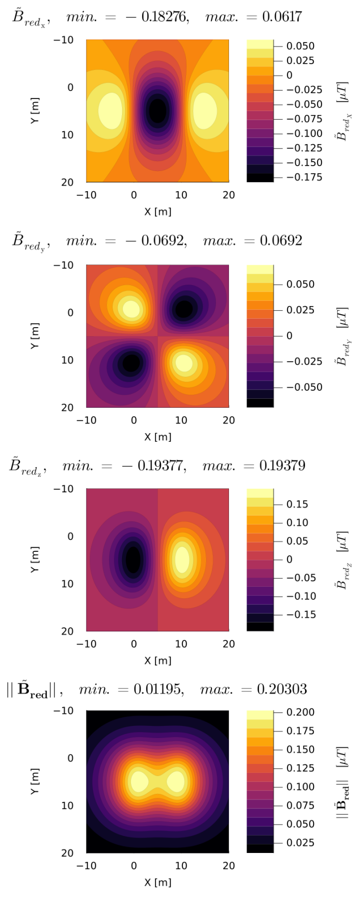


**Figure 3.3:** Visualisation of the points in which $\tilde{\mathbf{B}}_{\mathrm{red}}$ is calculated.

The x-, y-, and z-components of the reduced magnetic flux density are presented in separate plots. Moreover, the norm of the reduced magnetic flux density, denoted as $||\tilde{\mathbf{B}}_{\text{red}}||$, is shown. It is expected



**(a)** Approximated reduced magnetic flux density using point matching.



**(b)** Approximated reduced magnetic flux density using average formulation.

**Figure 3.4:** The approximated reduced magnetic flux density using uniform basis functions.

In both Figure 3.4a and 3.4b, the expected symmetries can be recognised.

# 4 Method of Moments Using Linear Basis Functions

In practice, the magnetisation is not constant in a volume. Therefore, using linear basis functions instead of uniform basis functions may improve the approximation of the magnetisation of the object [8, p. 45]. In this chapter, MoM is extended to the use of linear basis functions. Firstly, it is explained how the meshing is done. Thereafter, the weak formulation is derived for a mesh consisting of one mesh element. Two different sets of weighting functions are used to obtain a linear system. Lastly, it is explained how the interaction matrix is derived when a mesh consisting of more than one element is used. This chapter is primarily based on [1].

## 4.1. Meshing the Domain

When using uniform basis functions, any polyhedron shape can be selected as mesh elements. However, the use of linear basis functions restricts the mesh elements to right triangular prisms. In other words, prisms with two parallel and congruent triangular faces and three rectangular faces. A sketch of one single element, denoted as $\mathcal{P}$, is depicted below.

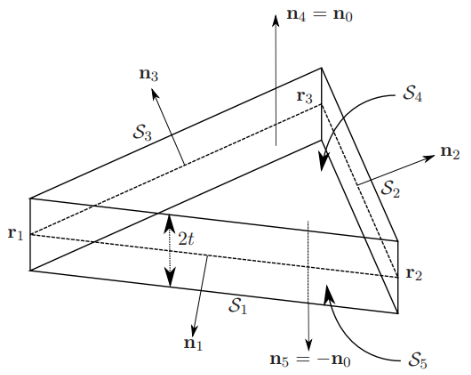


**Figure 4.1:** Triangular Prism $\mathcal{P}$ [1].

The points $\mathbf{r_1}$, $\mathbf{r_2}$ and $\mathbf{r_3}$ are referred to as the vertices of element $\mathcal{P}$. Unit length tangential vectors along the sides $\mathcal{S}_i$, $i = 1, 2, 3$ are introduced

$$\tau_1 = \frac{\mathbf{r_2} - \mathbf{r_1}}{||\mathbf{r_2} - \mathbf{r_1}||}, \quad \tau_2 = \frac{\mathbf{r_3} - \mathbf{r_2}}{||\mathbf{r_3} - \mathbf{r_2}||}, \quad \tau_3 = \frac{\mathbf{r_1} - \mathbf{r_3}}{||\mathbf{r_1} - \mathbf{r_3}||}. \tag{4.1}$$

The outward pointing normal vector $\mathbf{n_0}$ can be calculated as follows

$$\mathbf{n_0} = \frac{(\mathbf{r_2} - \mathbf{r_1}) \times (\mathbf{r_3} - \mathbf{r_2})}{||(\mathbf{r_2} - \mathbf{r_1}) \times (\mathbf{r_3} - \mathbf{r_2})||}. \tag{4.2}$$

The remaining three normal vectors are defined as

$$\mathbf{n}_i = \tau_i \times \mathbf{n}_0 \quad \text{for} \quad i = 1, 2, 3. \tag{4.3}$$

The derivation of the MoM using linear basis functions is explained using this element $\mathcal{P}$ as reference. The same notation as shown Figure 4.1 is used throughout the chapter.

## 4.2. Derivation of the Weak Formulation

To get a better understanding how the weak formulation is derived, a mesh is considered with only one mesh element $\mathcal{P}$. The assumption is made that the magnetisation is uniform in the thickness direction of $\mathcal{P}$. Using the notation introduced in the previous section, Lepelaars defines three basis functions on element $\mathcal{P}$ as

$$\phi_1(\mathbf{r}) = \begin{cases} \frac{(\mathbf{r}-\mathbf{r_2})\cdot\mathbf{n_2}}{(\mathbf{r_1}-\mathbf{r_2})\cdot\mathbf{n_2}} & \text{if} \quad \mathbf{r} \in \mathcal{P}, \\ 0 \quad \text{elsewhere}, \end{cases} \tag{4.4}$$

$$\phi_2(\mathbf{r}) = \begin{cases} \frac{(\mathbf{r}-\mathbf{r_3})\cdot\mathbf{n_3}}{(\mathbf{r_2}-\mathbf{r_3})\cdot\mathbf{n_3}} & \text{if} \quad \mathbf{r} \in \mathcal{P}, \\ 0 \quad \text{elsewhere}, \end{cases} \tag{4.5}$$

$$\phi_3(\mathbf{r}) = \begin{cases} \frac{(\mathbf{r}-\mathbf{r_1})\cdot\mathbf{n_1}}{(\mathbf{r_3}-\mathbf{r_1})\cdot\mathbf{n_1}}, & \text{if} \quad \mathbf{r} \in \mathcal{P}, \\ 0 \quad \text{elsewhere}. \end{cases} \tag{4.6}$$

The basis functions are linear on the triangle. Moreover, it can be verified that

$$\phi_i(\mathbf{r_j}) = \begin{cases} 1 & \text{if} \quad i = j, \\ 0 & \text{otherwise}, \end{cases} \qquad \text{for} \quad i, j = 1, 2, 3. \tag{4.7}$$

Note that the basis functions are invariant under a translation of $\mathbf{r}$ in the $\mathbf{n_0}$-direction. The estimated magnetisation of $\mathcal{P}$ can be written as

$$\tilde{\mathbf{M}}(\mathbf{r}) = \sum_{i=1}^{3} \tilde{\mathbf{M}}_i \phi_i(\mathbf{r}), \tag{4.8}$$

$$= \sum_{i=1}^{3} (\tilde{M}_{ix}\mathbf{u}_1 + \tilde{M}_{iy}\mathbf{u}_2 + \tilde{M}_{iz}\mathbf{u}_3)\phi_i(\mathbf{r}). \tag{4.9}$$

To find the total magnetic flux density, the steps from Section 3.2 are repeated. By substituting the expression for $\tilde{\mathbf{M}}$ in Equation 2.41, the total magnetic flux density can be approximated at any point

$$\tilde{\mathbf{B}}(\mathbf{r}) = \mathbf{B}_{\text{red}}(\mathbf{r}) + \mathbf{B}_{\text{ext}}(\mathbf{r}), \tag{4.10}$$

$$= \nabla \times \left( \frac{\mu_0}{4\pi} \iiint_{\mathcal{P}} \frac{(\nabla' \times \mathbf{M})(\mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||} - \nabla' \times \left( \frac{\mathbf{M}(\mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||} \right) \mathrm{d}\mathbf{r}' \right) + \mathbf{B}_{\text{ext}}(\mathbf{r}), \tag{4.11}$$

$$= \nabla \times \frac{\mu_0}{4\pi} \left( \iiint_{\mathcal{P}} \frac{(\nabla' \times \mathbf{M})(\mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||} \mathrm{d}\mathbf{r}' - \iiint_{\mathcal{P}} \nabla' \times \left( \frac{\mathbf{M}(\mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||} \right) \mathrm{d}\mathbf{r}' \right) + \mathbf{B}_{\text{ext}}(\mathbf{r}), \tag{4.12}$$

$$\tag{4.13}$$

Consider the first integral. Observe that inside $\mathcal{P}$

$$(\nabla' \times \mathbf{M})(\mathbf{r}') = \nabla' \times \sum_{i=1}^{3} \tilde{\mathbf{M}}_{\mathbf{i}} \phi_i(\mathbf{r}'), \tag{4.14}$$

$$= -\sum_{i=1}^{3} \tilde{\mathbf{M}}_{\mathbf{i}} \times \nabla' \phi_i(\mathbf{r}'), \tag{4.15}$$

$$= \frac{-\mathbf{M}_1 \times \mathbf{n_2}}{(\mathbf{r_1} - \mathbf{r_2}) \cdot \mathbf{n_2}} - \frac{\mathbf{M}_2 \times \mathbf{n_3}}{(\mathbf{r_2} - \mathbf{r_3}) \cdot \mathbf{n_3}} - \frac{\mathbf{M}_3 \times \mathbf{n_1}}{(\mathbf{r_3} - \mathbf{r_1}) \cdot \mathbf{n_1}}, \tag{4.16}$$

$$= \frac{\mathbf{n_2} \times \mathbf{M}_1}{(\mathbf{r_1} - \mathbf{r_2}) \cdot \mathbf{n_2}} + \frac{\mathbf{n_3} \times \mathbf{M}_2}{(\mathbf{r_2} - \mathbf{r_3}) \cdot \mathbf{n_3}} + \frac{\mathbf{n_1} \times \mathbf{M}_3}{(\mathbf{r_3} - \mathbf{r_1}) \cdot \mathbf{n_1}} = \mathbf{A_1}. \tag{4.17}$$

Thus, the first integral can be simplified

$$\iiint_{\mathcal{P}} \frac{(\nabla' \times \mathbf{M})(\mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||} \,\mathrm{d}\mathbf{r}' = \mathbf{A_1} \iiint_{\mathcal{P}} \frac{1}{||\mathbf{r} - \mathbf{r}'||} \,\mathrm{d}\mathbf{r}'. \tag{4.18}$$

By applying Gauss' theorem on the second integral, the integral can be written as

$$\iiint_{\mathcal{P}} \nabla' \times \left( \frac{\mathbf{M}(\mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||} \right) \mathrm{d}\mathbf{r}' = \iint_{\partial\mathcal{P}} \mathbf{n}(\mathbf{r}') \times \frac{\mathbf{M}(\mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||} \,\mathrm{d}\mathbf{r}'. \tag{4.19}$$

Remember that the boundary of $\mathcal{P}$ consists of five surfaces, denoted by $\mathcal{S}_j$ for $j = 1, \dots, 5$. The integral equals

$$\iint_{\partial\mathcal{P}} \mathbf{n}(\mathbf{r}') \times \frac{\mathbf{M}(\mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||} \,\mathrm{d}\mathbf{r}' = \sum_{j=1}^{5} \iint_{\mathcal{S}_j} \mathbf{n_j} \times \frac{\mathbf{M}(\mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||} \,\mathrm{d}\mathbf{r}', \tag{4.20}$$

$$= \sum_{j=1}^{5} \mathbf{n_j} \times \iint_{\mathcal{S}_j} \frac{\sum_{i=1}^{3} \tilde{\mathbf{M}}_i \phi_i(\mathbf{r})}{||\mathbf{r} - \mathbf{r}'||} \,\mathrm{d}\mathbf{r}', \tag{4.21}$$

$$= \sum_{j=1}^{5} \sum_{i=1}^{3} (\mathbf{n_j} \times \tilde{\mathbf{M}}_i) \iint_{\mathcal{S}_j} \frac{\phi_i(\mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||} \,\mathrm{d}\mathbf{r}'. \tag{4.22}$$

The reduced magnetic flux density equals

$$\mathbf{B}_{\text{red}}(\mathbf{r}) = \nabla \times \frac{\mu_0}{4\pi} \left( \mathbf{A_1} \iiint_{\mathcal{P}} \frac{1}{||\mathbf{r} - \mathbf{r}'||} \,\mathrm{d}\mathbf{r}' - \sum_{j=1}^{5} \sum_{i=1}^{3} (\mathbf{n_j} \times \tilde{\mathbf{M}}_i) \iint_{\mathcal{S}_j} \frac{\phi_i(\mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||} \,\mathrm{d}\mathbf{r}' \right), \tag{4.23}$$

$$= -\frac{\mu_0}{4\pi} \mathbf{A_1} \times \nabla \iiint_{\mathcal{P}} \frac{1}{||\mathbf{r} - \mathbf{r}'||} \,\mathrm{d}\mathbf{r}' + \frac{\mu_0}{4\pi} \sum_{j=1}^{5} \sum_{i=1}^{3} (\mathbf{n_j} \times \tilde{\mathbf{M}}_i) \times \nabla \iint_{\mathcal{S}_j} \frac{\phi_i(\mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||} \,\mathrm{d}\mathbf{r}'. \tag{4.24}$$

Again using Gauss' theorem on the first integral, expression simplifies

$$\mathbf{B}_{\text{red}}(\mathbf{r}) = \frac{\mu_0}{4\pi} \mathbf{A_1} \times \sum_{j=1}^{5} \mathbf{n}_j I_j(\mathbf{r}) + \frac{\mu_0}{4\pi} \sum_{j=1}^{5} \sum_{i=1}^{3} (\mathbf{n}_j \times \tilde{\mathbf{M}}_i) \times \nabla I_{ij}(\mathbf{r}), \tag{4.25}$$

$$\text{where} \quad I_j(\mathbf{r}) = \iint_{\mathcal{S}_j} \frac{1}{||\mathbf{r} - \mathbf{r}'||} d\mathbf{r}' \quad \text{and} \quad I_{ij}(\mathbf{r}) = \iint_{\mathcal{S}_j} \frac{\phi_i(\mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||} d\mathbf{r}'. \tag{4.26}$$

From Figure 4.1 can be concluded, that the surfaces $\mathcal{S}_j$ for $j = 1, 2, 3$ are rectangular, while the surfaces $\mathcal{S}_j$ for $j = 4, 5$ are triangular. The analytical expressions of the solution of $I_j$ and $I_{ij}$ depend on the shape of the surface $\mathcal{S}_j$. The following notation is introduced

$$L_T(\mathbf{r},\mathbf{r_1},\mathbf{r_2},\mathbf{r_3}) = \iint_{\mathcal{T}} \frac{1}{||\mathbf{r}-\mathbf{r}'||}\,\mathrm{d}\mathbf{r}', \quad L_R(\mathbf{r},\mathbf{r_1},\mathbf{r_2},\mathbf{r_3},\mathbf{r_4}) = \iint_{\mathcal{R}} \frac{1}{||\mathbf{r}-\mathbf{r}'||}\,\mathrm{d}\mathbf{r}', \tag{4.27}$$

$$K_T(\mathbf{r},\mathbf{r_1},\mathbf{r_2},\mathbf{r_3}) = \iint_{\mathcal{T}} \frac{\phi_1(\mathbf{r}')}{||\mathbf{r}-\mathbf{r}'||}\,\mathrm{d}\mathbf{r}', \quad K_R(\mathbf{r},\mathbf{r_1},\mathbf{r_2},\mathbf{r_3},\mathbf{r_4}) = \iint_{\mathcal{R}} \frac{\phi_{12}(\mathbf{r}')}{||\mathbf{r}-\mathbf{r}'||}\,\mathrm{d}\mathbf{r}', \tag{4.28}$$

where $\mathcal{T}$ is a triangle with vertices $\mathbf{r_1}$, $\mathbf{r_2}$ and $\mathbf{r_3}$ and $\mathcal{R}$ is a rectangle with vertices $\mathbf{r_1}$, $\mathbf{r_2}$, $\mathbf{r_3}$ and $\mathbf{r_4}$. In this notation, the second argument of $K_T$ denotes where the linear function $\phi$ assumes the value 1. For $K_R$ the second and third argument indicate the vertices where the linear function $\phi$ assumes the value 1. Analytical expressions for these integrals can be found in Appendix C.
As a result, the approximated total magnetic flux density equals

$$\tilde{\mathbf{B}}(\mathbf{r}) = [\mathbf{C}(\mathbf{r})] \begin{pmatrix} \tilde{\mathbf{M}}_1 \\ \tilde{\mathbf{M}}_2 \\ \tilde{\mathbf{M}}_3 \end{pmatrix} + \mathbf{B_{ext}}(\mathbf{r}), \tag{4.29}$$

$$= \big([\mathbf{C_1}(\mathbf{r})] \quad [\mathbf{C_2}(\mathbf{r})] \quad [\mathbf{C_3}(\mathbf{r})]\big) \begin{pmatrix} \tilde{\mathbf{M}}_1 \\ \tilde{\mathbf{M}}_2 \\ \tilde{\mathbf{M}}_3 \end{pmatrix} + \mathbf{B_{ext}}(\mathbf{r}), \tag{4.30}$$

$$\text{where} \quad [\mathbf{C_i}(\mathbf{r})] = \begin{pmatrix} \nabla\phi_i(\mathbf{r}) \times \mathbf{u}_1 \times \sum_{j=1}^5 \mathbf{n_j} I_j(\mathbf{r}) + \sum_{j=1}^5 (\mathbf{n_j} \times \mathbf{u}_1 \times \nabla I_{ij}(\mathbf{r})) \\ \nabla\phi_i(\mathbf{r}) \times \mathbf{u}_2 \times \sum_{j=1}^5 \mathbf{n_j} I_j(\mathbf{r}) + \sum_{j=1}^5 (\mathbf{n_j} \times \mathbf{u}_2 \times \nabla I_{ij}(\mathbf{r})) \\ \nabla\phi_i(\mathbf{r}) \times \mathbf{u}_3 \times \sum_{j=1}^5 \mathbf{n_j} I_j(\mathbf{r}) + \sum_{j=1}^5 (\mathbf{n_j} \times \mathbf{u}_3 \times \nabla I_{ij}(\mathbf{r})) \end{pmatrix}^T \tag{4.31}$$

Note that $[\mathbf{C}(\mathbf{r})]$ is a $3 \times 9$ matrix.
Now suppose there are $N$ mesh elements. For an arbitrary element $k$, the basis functions are denoted as $\phi_1^k, \phi_2^k$ and $\phi_3^k$. The estimated magnetisation on the object can be written as

$$\tilde{\mathbf{M}}(\mathbf{r}) = \sum_{h=1}^{N} \sum_{i=1}^{3} \tilde{\mathbf{M}}_i^h \phi_i^h(\mathbf{r}). \tag{4.32}$$

As a result, the total magnetic flux density is written as

$$\tilde{\mathbf{B}}(\mathbf{r}) = \sum_{h=1}^{N} [\mathbf{C}(\mathbf{r})]_h \begin{pmatrix} \tilde{\mathbf{M}}_1^h \\ \tilde{\mathbf{M}}_2^h \\ \tilde{\mathbf{M}}_3^h \end{pmatrix} + \mathbf{B_{ext}}(\mathbf{r}), \tag{4.33}$$

$$= \sum_{h=1}^{N} [\mathbf{C}(\mathbf{r})]_h \tilde{\mathbf{M}}^h + \mathbf{B_{ext}}(\mathbf{r}), \tag{4.34}$$

$$\tag{4.35}$$

where $[\mathbf{C}(\mathbf{r})]_h$ is derived as in Equation 4.31 The weak formulation can be obtained

$$\iiint_{\mathcal{P}_M} w_k(\mathbf{r}) \left( \tilde{\mathbf{B}}(\mathbf{r}) - \sum_{h=1}^{N} [\mathbf{C}(\mathbf{r})]_h \tilde{\mathbf{M}}^h - \mathbf{B}_{\text{ext}}(\mathbf{r}) \right) \mathrm{d}\mathbf{r} = \mathbf{0} \quad \text{for} \quad k = 1, \dots, 3N. \tag{4.36}$$

## 4.3. Point Matching

In this section, a point-matching formulation is considered, which implies that the weighting functions are Dirac delta functions. First, the evaluation points used in this formulation are identified. Thereafter, a linear system is derived from these evaluation points.

### 4.3.1. Evaluation Points

From Equation 4.8 can be derived that there are three unknown vectors per element. When the same weighting functions are used as described in Chapter 3, the number of unknowns would exceed the number of weighting functions, which gives an infinite number of solutions. Thus, three evaluation points per element have to be chosen. In this section, two different sets of evaluation points are considered, denoted by $E_1$ and $E_2$.

### Evaluation Points $E_1$

The first set that is considered is $E_1 = \{\mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}\}$, where $\mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}$ are as shown in Figure 4.1. However, calculating $K_R$ is not always possible when considering the set $E_1$ as evaluation points. Details on the calculations that identify the causes of the problems can be found in Appendix C. To overcome these challenges, the evaluation points are shifted to the center of one of the adjacent elements. A parameter $\epsilon_{shift}$ is defined to quantify the extent to which the evaluation point is shifted towards the center. $\epsilon_{shift}$ is a value between 0 and 1. For $\epsilon_{shift} = 1$, evaluation point equals the center of the element and for $\epsilon_{shift} = 0$ the evaluation point is not shifted. To get a better understanding of the evaluation points, a top view of a simple mesh, consisting of one mesh element, is depicted. Suppose that the vertices of element 1 equal $\mathbf{r_1}, \mathbf{r_2}$ and $\mathbf{r_3}$. The evaluation points are indicated by the red crosses. On the left-hand side the original evaluation points are shown, while on the right-hand side the evaluation points are shifted with $\epsilon_{shift} = 0.1$.

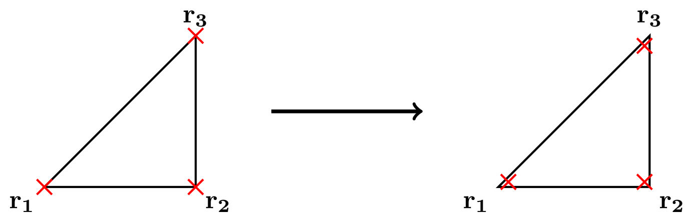


**Figure 4.2:** A visualization of the evaluation points $E_1$ with a shift of $\epsilon_{shift} = 0.1$ for one mesh element.

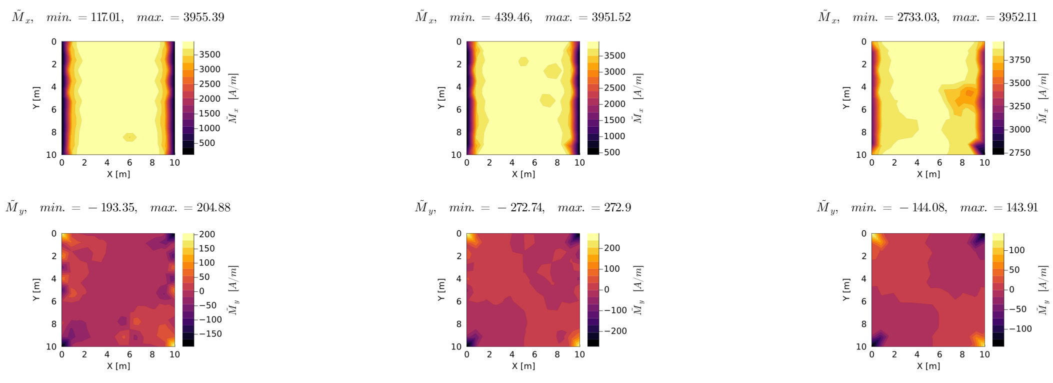
Considering these evaluation points, the three weighting functions equal

$$w_1(\mathbf{r}) = \delta(\mathbf{r} - (1 + \epsilon_{shift})\mathbf{r_1} - \epsilon_{shift}\mathbf{r_c}), \tag{4.37}$$

$$w_2(\mathbf{r}) = \delta(\mathbf{r} - (1 + \epsilon_{shift})\mathbf{r_2} - \epsilon_{shift}\mathbf{r_c}), \tag{4.38}$$

$$w_3(\mathbf{r}) = \delta(\mathbf{r} - (1 + \epsilon_{shift})\mathbf{r_3} - \epsilon_{shift}\mathbf{r_c}), \tag{4.39}$$

where $\mathbf{r_c}$ is the center of element $k$ and $\mathbf{r_1}, \mathbf{r_2}$ and $\mathbf{r_3}$ are the vertices of the element.
A disadvantage of this approach is the dependency of a parameter $\epsilon_{shift}$. Different values of $\epsilon_{shift}$ result in different outcomes for the magnetisation in the object. To explore the impact of $\epsilon_{shift}$, the x- and y-component at $z = 0.0$ is computed for the same geometry as described in Section 3.7. The mesh contains 244 mesh elements. The background field is chosen to be $50$ μT in the x-direction. A total of 200 points are uniformly distributed along both the x- and y-axes, resulting in 40,000 points. The minimum and maximum value of the magnetisation at these points is calculated and displayed above the visualisations.



**(a)** $\epsilon_{shift} = 0.001$ **(b)** $\epsilon_{shift} = 0.01$ **(c)** $\epsilon_{shift} = 0.1$

**Figure 4.3:** The x- and y-component of the magnetisation within a plate using linear basis functions with $E_1$ as evaluation points for various values of $\epsilon_{shift}$

From Figure 4.3 can be derived that for $\epsilon_{shift} = 0.01$ the x-component of the magnetisation satisfies the expected symmetries. For a larger value of $\epsilon_{shift}$ the magnetisation deviates from the expected behaviour. For a smaller value of $\epsilon_{shift}$, the y-component of the magnetisation becomes positive in regions where it should be negative.

### Evaluation Points $E_2$

To get rid of this dependency a second set of evaluation points is introduced, $E_2 = \{\frac{\mathbf{r_1}+\mathbf{r_2}}{2}, \frac{\mathbf{r_2}+\mathbf{r_3}}{2}, \frac{\mathbf{r_3}+\mathbf{r_1}}{2}\}$. The calculations in Appendix C show that no problems arise for evaluating the integrals, when this set of evaluation points is used. Figure 4.4 provides a top view of the evaluation points $E_2$ of the previously introduced simple mesh.

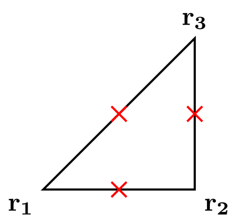


**Figure 4.4:** A visualization of the evaluation points $E_2$ for one mesh element.

Using the evaluation points from $E_2$, the weighting functions equal

$$w_1(\mathbf{r}) = \delta\left(\mathbf{r} - \frac{(\mathbf{r_1}+\mathbf{r_2})}{2}\right), \tag{4.40}$$

$$w_2(\mathbf{r}) = \delta\left(\mathbf{r} - \frac{(\mathbf{r_2}+\mathbf{r_3})}{2}\right), \tag{4.41}$$

$$w_3(\mathbf{r}) = \delta\left(\mathbf{r} - \frac{(\mathbf{r_3}+\mathbf{r_1})}{2}\right). \tag{4.42}$$

Observe that when using $E_2$ as evaluation points, the resulting linear system is not always a square system of equation. An example of such a mesh is given in the sketch below.

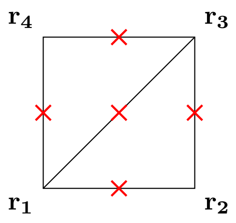


**Figure 4.5:** A visualization of the evaluation points $E_2$ for a mesh consisting of two mesh elements.

The system for this mesh results in a $15 \times 12$ system. A least squares solution is obtained. If there are multiple solutions, the one with smallest norm is selected. For the same mesh and background field as described in Section 4.3.1, the resulting magnetisation using $E_2$ as evaluation points are shown in the figure below.

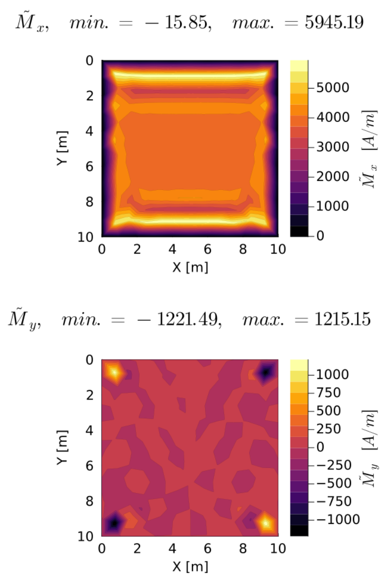


**Figure 4.6:** The x- and y-component of the magnetisation within a plate using linear basis functions with $E_2$ as evaluation points

This figure shows that the x-component of the approximated magnetisation is the highest close to the upper and lower boundary of the plate. Also, the minimum value is negative, while the magnetisation in the x-component is always positive for this background field. Moreover, the y-component shows an unexpected pattern. Based on Figure 4.3 and Figure 4.6, the set $E_1$ is chosen as set of evaluation points with $\epsilon_{shift} = 0.01$.

## 4.3.2. Obtain Linear System

In this section, it is shown how the linear system is derived, considering one mesh element. The decision is made to use the vertices of the prism as evaluation points, with $\epsilon_{shift} = 0.01$. Remember that the weight functions can be defined as follows

$$w_1(\mathbf{r}) = \delta(\mathbf{r} - (1 + \epsilon_{shift})\mathbf{r_1} - \epsilon_{shift}\mathbf{r_c}), \tag{4.43}$$

$$w_2(\mathbf{r}) = \delta(\mathbf{r} - (1 + \epsilon_{shift})\mathbf{r_2} - \epsilon_{shift}\mathbf{r_c}), \tag{4.44}$$

$$w_3(\mathbf{r}) = \delta(\mathbf{r} - (1 + \epsilon_{shift})\mathbf{r_3} - \epsilon_{shift}\mathbf{r_c}). \tag{4.45}$$

where $\mathbf{r}_\text{c}$ is the center of element $\mathcal{P}$ and $\mathbf{r_1}, \mathbf{r_2}$ and $\mathbf{r_3}$ are the vertices of the element. The weak formulation is worked out for a mesh with one element. The weak formulation for an arbitrary $i = 1, 2, 3$ simplifies to

$$\iiint_{\mathcal{P}} \delta(\mathbf{r} - (1 + \epsilon_{shift})\mathbf{r_i} - \epsilon_{shift}\mathbf{r}_\text{c}) \left(\tilde{\mathbf{B}}(\mathbf{r}) - [\mathbf{C}(\mathbf{r})]\tilde{\mathbf{M}} - \mathbf{B}_\text{ext}(\mathbf{r})\right) \mathrm{d}\mathbf{r} = \mathbf{0}, \tag{4.46}$$

$$\tilde{\mathbf{B}}^i = [\mathbf{C}_i]\tilde{\mathbf{M}} + \mathbf{B}^i_\text{ext}, \tag{4.47}$$

where $[\mathbf{C}_i]$ equals $[\mathbf{C}(\mathbf{r_i})]$. Observe that $\tilde{\mathbf{B}}^i$ and $\mathbf{B}^i_\text{ext}$ represent the total and external magnetic flux density at the point $\mathbf{r} - (1 + \epsilon_{shift})\mathbf{r_i} - \epsilon_{shift}\mathbf{r}_\text{c}$. Assembling for all three evaluation points, the system of the discretised problem equals

$$\tilde{\mathbf{B}} = [\mathbf{C}]\tilde{\mathbf{M}} + \mathbf{B}_\text{ext}, \tag{4.48}$$

where the matrix $[\mathbf{C}]$, and vectors $\tilde{\mathbf{M}}$ and $\mathbf{B}_\text{ext}$ are as follows

$$[\mathbf{C}] = \begin{pmatrix} [\mathbf{C}(\mathbf{r_1})] \\ [\mathbf{C}(\mathbf{r_2})] \\ [\mathbf{C}(\mathbf{r_3})] \end{pmatrix}, \quad \tilde{\mathbf{M}} = \begin{pmatrix} \tilde{\mathbf{M}}_1 \\ \tilde{\mathbf{M}}_2 \\ \tilde{\mathbf{M}}_3 \end{pmatrix}, \quad \mathbf{B}_\text{ext} = \begin{pmatrix} \mathbf{B}^1_\text{ext} \\ \mathbf{B}^2_\text{ext} \\ \mathbf{B}^3_\text{ext} \end{pmatrix}. \tag{4.49}$$

Note that $[\mathbf{C}]$ is a $9 \times 9$ matrix and both $\tilde{\mathbf{M}}$ and $\mathbf{B}_{\text{ext}}$ are vectors of size $9$. In a similar way as in Chapter 3, the linear system is obtained

$$[\mathbf{A}(\chi_m)]\tilde{\mathbf{M}} = \frac{\chi_m}{\mu_0(1+\chi_m)}\mathbf{B}_{\text{ext}}, \tag{4.50}$$

where $[\mathbf{A}(\chi_m)] = [\mathbf{I_9}] - \frac{\chi_m}{\mu_0(1+\chi_m)}[\mathbf{C}]$.

## 4.4. Galerkin Method

In this section, a Galerkin method is introduced. A Galerkin method offers two advantages. Firstly, as shown in Section 4.3, the results on the approximated magnetisation depend on which evaluation points are chosen. Secondly, and most importantly, the Galerkin method produces a symmetric interaction matrix. This symmetry ensures that the resulting linear system is also symmetric, which simplifies the solution process. In this formulation, the basis functions are used as weighting functions. Thus the weighting functions equal

$$w_1(\mathbf{r}) = \begin{cases} \phi_1(\mathbf{r}) & \text{if} \quad \mathbf{r} \in \mathcal{P}, \\ 0 & \text{otherwise}, \end{cases} \tag{4.51}$$

$$w_2(\mathbf{r}) = \begin{cases} \phi_2(\mathbf{r}) & \text{if} \quad \mathbf{r} \in \mathcal{P}, \\ 0 & \text{otherwise}, \end{cases} \tag{4.52}$$

$$w_3(\mathbf{r}) = \begin{cases} \phi_3(\mathbf{r}) & \text{if} \quad \mathbf{r} \in \mathcal{P}, \\ 0 & \text{otherwise}, \end{cases} \tag{4.53}$$

The weak formulation for an arbitrary $i = 1, 2, 3$ results in

$$\iiint\limits_{\mathcal{P}} \phi_i(\mathbf{r}) \left(\tilde{\mathbf{B}}(\mathbf{r}) - [\mathbf{C}(\mathbf{r})]\tilde{\mathbf{M}} - \mathbf{B}_{\text{ext}}(\mathbf{r})\right) \mathrm{d}\mathbf{r} = \mathbf{0}, \tag{4.54}$$

$$\iiint\limits_{\mathcal{P}} \phi_i(\mathbf{r})\tilde{\mathbf{B}}(\mathbf{r})\, \mathrm{d}\mathbf{r} = \iiint\limits_{\mathcal{P}} \phi_i(\mathbf{r}) \left([\mathbf{C}(\mathbf{r})]\tilde{\mathbf{M}} - \mathbf{B}_{\text{ext}}(\mathbf{r})\right) \mathrm{d}\mathbf{r}. \tag{4.55}$$

Assembling for all three weighting functions, the discretised problem equals

$$\{\tilde{\mathbf{B}}\} = [\{\mathbf{C}\}]\tilde{\mathbf{M}} + \{\mathbf{B}_{\text{ext}}\}, \tag{4.56}$$

where the matrix $[\{\mathbf{C}\}]$, and vectors $\tilde{\mathbf{M}}$ and $\{\mathbf{B}_{\text{ext}}\}$ are as follows

$$[\{\mathbf{C}\}] = \begin{pmatrix} \iiint\limits_{\mathcal{P}} \phi_1(\mathbf{r})[\mathbf{C}(\mathbf{r})]\, \mathrm{d}\mathbf{r} \\ \iiint\limits_{\mathcal{P}} \phi_2(\mathbf{r})[\mathbf{C}(\mathbf{r})]\, \mathrm{d}\mathbf{r} \\ \iiint\limits_{\mathcal{P}} \phi_3(\mathbf{r})[\mathbf{C}(\mathbf{r})]\, \mathrm{d}\mathbf{r} \end{pmatrix}, \quad \tilde{\mathbf{M}} = \begin{pmatrix} \tilde{\mathbf{M}}_1 \\ \tilde{\mathbf{M}}_2 \\ \tilde{\mathbf{M}}_3 \end{pmatrix}, \quad \{\mathbf{B}_{\text{ext}}\} = \begin{pmatrix} \iiint\limits_{\mathcal{P}} \phi_1(\mathbf{r})\mathbf{B}_{\text{ext}}(\mathbf{r})\, \mathrm{d}\mathbf{r} \\ \iiint\limits_{\mathcal{P}} \phi_2(\mathbf{r})\mathbf{B}_{\text{ext}}(\mathbf{r})\, \mathrm{d}\mathbf{r} \\ \iiint\limits_{\mathcal{P}} \phi_3(\mathbf{r})\mathbf{B}_{\text{ext}}(\mathbf{r})\, \mathrm{d}\mathbf{r} \end{pmatrix}. \tag{4.57}$$

The obtained linear system equals

$$[\mathbf{A}(\chi_m)]\tilde{\mathbf{M}} = \frac{\chi_m}{\mu_0(1+\chi_m)}\{\mathbf{B}_{\text{ext}}\}, \tag{4.58}$$

where $[\mathbf{A}(\chi_m)] = [\mathbf{I_9}] - \frac{\chi_m}{\mu_0(1+\chi_m)}[\{\mathbf{C}\}]$.

## 4.5. Derivation of Interaction Matrix

There are two challenges in finding the interaction matrix using linear basis functions. First of all, analytical expressions must be derived for the integral equations $I_j$ and $\nabla I_{ij}$. The derivations from Lepelaars and Morandi are combined to obtain these analytical expressions [1] [7]. The analytical expressions can be found in Appendix C. Second of all, it is required for the magnetisation to be continuous within the object. Thus, a continuity constraint must be incorporated into the calculations.

### 4.5.1. Continuity Constraint

The derivation of the requirement for the continuity of the approximated magnetisation is explained using a small example mesh. Before introducing this mesh, a distinction is made between global and local nodes. The global node number is one which is unique to each node and differentiates it from all other nodes. Additionally, each element has its own local numbering system. The sketch below visualises a mesh with two mesh elements and four nodes. The global nodes are denoted by bold, unique numbers, while the local nodes are labeled with numbers from one to three at each node.

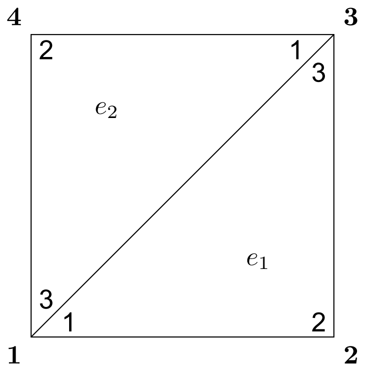


**Figure 4.7:** A visualization of a mesh consisting of two mesh elements and four nodes

When the interaction matrix is assembled in a straightforward manner, an $18 \times 18$ linear system is obtained

$$[\mathbf{A}(\chi_m)]\tilde{\mathbf{M}} = \frac{\chi_m}{\mu_0(1+\chi_m)}\mathbf{B}_{\text{ext}}, \tag{4.59}$$

$$\text{where} \quad \tilde{\mathbf{M}} = \begin{pmatrix} \tilde{\mathbf{M}}_1^1 \\ \tilde{\mathbf{M}}_2^1 \\ \tilde{\mathbf{M}}_3^1 \\ \tilde{\mathbf{M}}_1^2 \\ \tilde{\mathbf{M}}_2^2 \\ \tilde{\mathbf{M}}_3^2 \end{pmatrix}. \tag{4.60}$$

When solving this system, the function for the magnetisation may be discontinuous across the shared edge of the two mesh elements. However, the approximated magnetisation must be continuous within the object. For this mesh, continuity implies that $\tilde{\mathbf{M}}_1^1 = \tilde{\mathbf{M}}_3^2$ and $\tilde{\mathbf{M}}_3^1 = \tilde{\mathbf{M}}_1^2$. These equations must hold to achieve a continuous magnetisation throughout the object. Taking this into account, the objective is to find a solution to the following problem

$$[\mathbf{A}(\chi_m)]\tilde{\mathbf{M}} = \frac{\chi_m}{\mu_0(1+\chi_m)}\mathbf{B}_{\text{ext}}, \tag{4.61}$$

$$\text{s.t.} \quad [\mathbf{D}]\tilde{\mathbf{M}} = \mathbf{0}, \tag{4.62}$$

$$\text{where} \quad [\mathbf{D}] = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & -1 & 0 & 0 \end{pmatrix}. \tag{4.63}$$

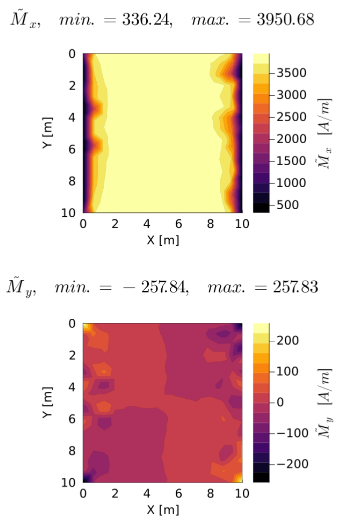
Adding this constraint to the system reduces the number of unknowns, which in turn decreases the number of weighting functions. The size of the resulting linear system is $12 \times 12$.

## 4.6. Visualisation of $\tilde{\mathbf{M}}$ and $\tilde{\mathbf{B}}_{\mathbf{red}}$

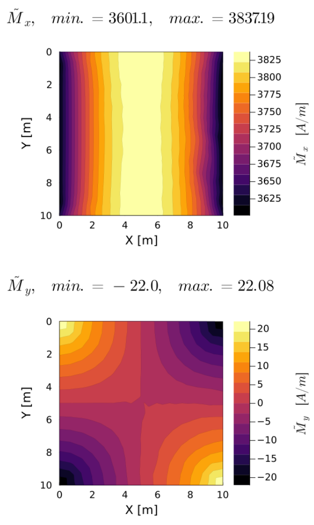
In this section, the magnetisation and reduced magnetic flux density of a linearly reacting plate of $10 \times 10 \times 0.002$ m, with magnetic susceptibility $\chi_m = 100$, is visualised. Suppose the plate is placed in an external uniform background field of $50\ \mu\text{T}$, which points in the positive x-direction. The same mesh is considered as described in Section 3.7, which means the mesh consists of 402 mesh elements and 228 nodes. In the Galerkin method, the relative tolerance in the function `hcubature()` is set to $10^{-6}$.

### 4.6.1. Visualisation of $\tilde{\mathrm{M}}$

In this subsection, the magnetisation at $z = 0$ m within the plate is visualised through a top-down view. The magnetisation is calculated in a similar way as described in Section 3.7.1.



**(a)** Approximated magnetisation using point matching.



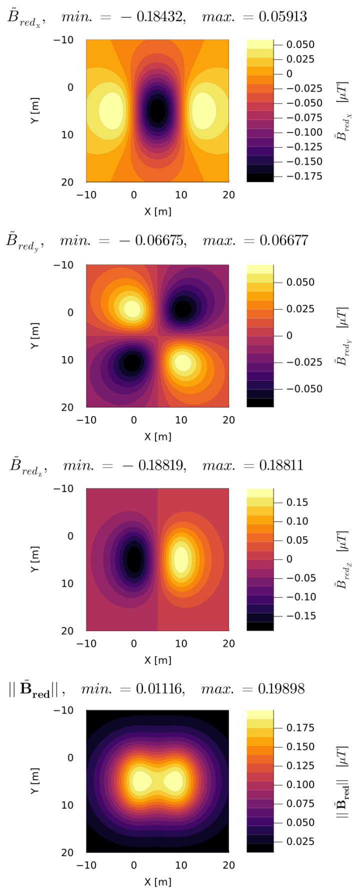
**(b)** Approximated magnetisation using the Galerkin method.

**Figure 4.8:** The approximated magnetisation using linear basis functions within the plate.

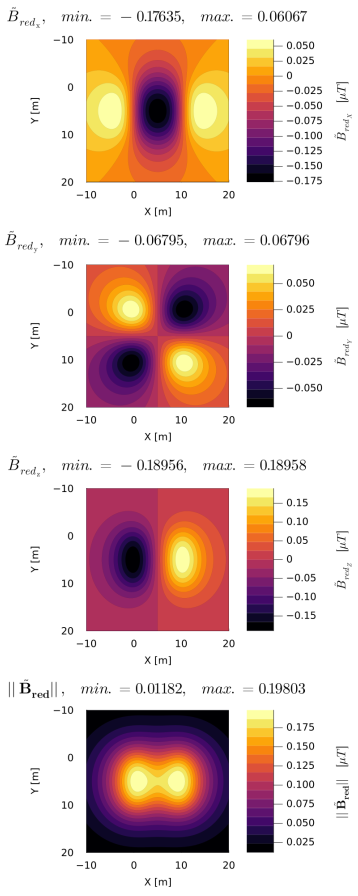
It is observed that the minimum value of the x-component of the approximated magnetisation using point matching is significantly lower than that obtained using the Galerkin method. Also, x-component is lower compared to the results shown in Figure 3.2. The expected symmetry in the y-component of the magnetisation is recognisable. However, there are regions where the magnetisation is estimated to be negative, while in practice it would be positive, or vice versa. The visualisations of the approximated magnetisation using the Galerkin method demonstrate the expected symmetries perfectly. However, the absolute value of the extreme values of the y-component of the magnetisation are substantially lower compared to the other methods.

### 4.6.2. Visualisation of $\tilde{\mathrm{B}}_{\mathrm{red}}$

In this subsection, the reduced magnetic flux density due to the magnetisation of the plate is visualised. The reduced magnetic flux density is calculated in a similar way as described in Section 3.7.2.



**(a)** Approximated reduced magnetic flux density using point matching.



**(b)** Approximated reduced magnetic flux density using the Galerkin method.

**Figure 4.9:** The approximated reduced magnetic flux density using linear basis functions.

# 5 Performance Tips for Julia

Programming languages can be categorised into two types. On the one hand, there are statically compiled languages such as Fortran and C/C++. These languages run quickly and efficiently, but are hard to learn and use. On the other hand, there are dynamic languages, such as Python and MATLAB. These language are more user-friendly and have more intuitive syntax. However, they often struggle with handling computationally intensive tasks efficiently. Therefore, developers often use a combination of a statically compiled language and a dynamic language, also known as the two-language problem [23]. Julia overcomes the two-language problem [24]. Julia is an open-source language created by Bezanson, Karpinski, Shah, and Edelman, which launched its initial version in 2012 [25]. It promises the ease of a dynamic language at the speed of a statically compiled language [26]. Julia achieves performance comparable to C/C++ and Fortran and is significantly faster than MATLAB and Python [9]. This is possible because Julia uses just-in-time (JIT) compilation [27]. Julia compiles code during the execution and generates efficient, type-specialised code based on the types encountered [28]. To make optimal use of Julia's performance, this chapter provides a few useful techniques that optimise Julia code in terms of computation speed.

## 5.1. Benchmarking

When writing code, it is interesting to benchmark the code. In this research, benchmarking will refer to testing the time spent to execute code and the memory usage of the code and compare it to alternative implementations. Julia has a built-in `@time` macro function, which returns the time it took to execute the code, the number of heap allocations, and the total number of bytes its execution caused to be allocated [27]. However, to get more accurate values it is recommended to use the package `BenchmarkTools`. On it, there is a macro called `@btime`. This function executes the code multiple times in order to reduce noise and to obtain more accurate measurements. Thereafter, it prints the minimum run-time and memory allocation [29]. The macro `@belapsed` returns the minimum time in seconds. The macro `@ballocated` is comparable to `@belapsed`, but it returns the total number of bytes allocated on the heap corresponding to the trial with the minimum elapsed time measured during the benchmark [30].

## 5.2. Reducing Heap Allocations

An operating system's memory consists of different segments, among others, the stack and heap. Stacks are regions of memory where data is stored in the order it gets them and removes the values in the opposite order, also known as Last In, First Out (LIFO) manner. Moreover, all data stored on the stack must have a predetermined, fixed size. As a result, stack allocation is very simple and typically faster than heap-based memory allocation [31]. The heap is dynamic memory, which implies that it can be allocated, resized and freed during program runtime. When data is put on the heap, the memory allocator finds an empty spot in the heap that is big enough, marks it as being in use, and returns a pointer. A pointer is a variable whose value is the address of another variable. This process is referred to as allocating [32]. The allocating makes the heap allocations costly [33]. Pushing to the stack is faster than allocating on the heap, since the data will always be placed at the top of the stack [32].

In general, new mutable values are stored in the heap, and new immutable values are stored in the stack. In some situations, the implementation can be adjusted such that the data is stored on the stack instead of the heap. An example from [33] is shown below.

```
using BenchmarkTools

A = rand(100,100)
B = rand(100,100)
C = rand(100,100)

function inner_alloc!(C,A,B)
  for j in 1:100, i in 1:100
    val = [A[i,j] + B[i,j]]
    C[i,j] = val[1]
  end
end
@btime inner_alloc!(C,A,B)

# 226.092 μs (10000 allocations: 625.00 KiB)

function inner_noalloc!(C,A,B)
  for j in 1:100, i in 1:100
    val = A[i,j] + B[i,j]
    C[i,j] = val[1]
  end
end
@btime inner_noalloc!(C,A,B)

# 6.540 μs (0 allocations: 0 bytes)
```

The arrays are created using `rand()`. When there is no type specified, the elements are of type Float64 by default [34]. Note that this code first loops over the rows and then over the columns. This approach is faster in Julia, which uses column-major order, similar to MATLAB. In contrast, Python's NumPy library uses row-major order [33]. The difference between the two functions lies in the type of the variable `val`. In the first function, `val` is an array, whereas in the second function, `val` is of the same type as the elements of the arrays `A` and `B`. Specifically, in this case, `val` is a `Float64`. Since an array is a mutable object, it is stored on the heap. As a result, in the first function, not only is the allocation inefficient, but the code is also not optimal in terms of computation speed. In contrast, the size of a Float64 number is known at compile-time, namely 64-bits. Therefore, in the second function, `val` is stored onto the stack, resulting in less memory usage and faster computation.

An alternative to heap allocations is the package `StaticArrays.jl`. `StaticArrays.jl` provides a framework for implementing statically sized arrays in Julia. Helpful macros from the package are `SVector`, `SMatrix` and `SArray` [35]. These arrays have their size known at compile-time and are immutable. As a result, these structures are stored on the stack [36]. An example from [33] illustrates how this package can be used.

```
using StaticArrays

function static_inner_alloc!(C,A,B)
  for j in 1:100, i in 1:100
    val = @SVector [A[i,j] + B[i,j]]
    C[i,j] = val[1]
  end
end
@btime static_inner_alloc!(C,A,B)

# 6.544 μs (0 allocations: 0 bytes)
```

Important to note is that arrays can only be static if they are sufficiently small. Once the array reaches a certain size, the array has to be allocated in the heap instead of the stack. It is recommended to not use static arrays if the system requires more than around 100 variables [35].

## 5.3. Type Stability

One key to performance in Julia is due to the fact that the compiler determines the concrete return type of any function call it encounters. An example of a concrete type is `Int64`. The compiler knows the value and the size of that type. Abstract types, such as `Number`, have no concrete objects or values of their own and the compiler has no information [37]. In Julia, a function can have multiple implementations, called methods, each one for a different combination of argument types. At run-time, the language will determine which specific method is most applicable to the types of the arguments. This is known as multiple dispatch [26]. In order to write high-performance Julia code, it is important to write type stable code. Type stable code means that the concrete type of its output is entirely determined by the concrete types of its arguments [37]. In other words, the type of the output cannot vary depending on the values of the inputs [27]. The code below shows an example of a function that is not type stable.

```
using BenchmarkTools

n = 10000
vfloat = rand(Float64, n)
vint = rand(Int64, n)

function sum(v, t)
    res = v[1]
    for i = 2:length(v)
        elm = v[i] < t ? v[i] : t
        res += elm
    end
    return res
end
@btime sum(vfloat, 0.5)
# 10.200 μs (1 allocation: 16 bytes)

@btime sum(vint, 0.5)
#52.900 μs (1 allocation: 16 bytes)
```

This function sums the elements of the vector `v`, but if any element exceeds the threshold `t`, it adds `t` instead. Note there is a large difference in computation time. The Julia compiler is not able to deduce a concrete return type of the method, since it can be either an Int64 or Float64. In more complex code, it may be hard to understand the type behaviour of a function. Fortunately, Julia has a built-in macro, `@code_warntype`, which shows types that are not concrete [37]. Although the macro returns extensive output, the colored parts are the most important. When running

```
    @code_warntype sum(vint, 0.5)
```

one of the output lines is `Body::Union{Float64, Int64}`, indicating that the return type may be either a Float64 or Int64. Moreover, the line `res::Union{Float64, Int64}` shows that the type of `res` changes. This type-related property will be referred to as type groundedness, indicating that the type of each variable depends only on the types of the arguments [37]. To better understand the distinction between these properties, an example is provided that is type stable, but not type grounded.

```
using BenchmarkTools

function sumofsins1(n)
    r = 0
    for i in 1:n
        r += sin(3.4)
    end
    return r
end
@btime sumofsins1(100000)

#   244.400 μs (0 allocations: 0 bytes)

function sumofsins2(n)
    r = 0.0
    for i in 1:n
        r += sin(3.4)
    end
    return r
end
@btime sumofsins2(100000)

#   72.900 μs (0 allocations: 0 bytes)
```

In the first function, the variable `r` is initialised as an integer. Thus, in the first iteration the type of `r` is transformed from an Int64 to a Float64. The type of `r` is not stable over time. As a result, the compiler cannot optimise the main loop, since it cannot guarantee that the type of the variable `r` will remain the same throughout the loop. The compiler must check the type of `r` in every iteration. In the second function, the variable is initialised as a float and the variable is type stable. Therefore, the execution time of the second function is significantly shorter than the execution time of the first function.

## 5.4. Parallel Computing

Another key feature of Julia is the built-in parallel processing support. To create a language for parallel computing, was one of the motivations to build Julia [38]. In this research, the package `Polyester.jl` is used. This package provides low-overhead multithreading [39]. Julia operates with a single thread for execution as its default configuration. When launching Julia via the Powershell on Windows, the environment variable `JULIA_NUM_THREADS` governs the number of threads. For example, when `JULIA_NUM_THREADS` is set as `$env:JULIA_NUM_THREADS=4`, there are four available threads after launching Julia. Important to note is that `JULIA_NUM_THREADS` must be defined before launching Julia. Configuring it within the `~/.julia/config/startup.jl` is ineffective, since it occurs too late in the startup process [40]. It is recommended to set the number of threads equal to the computer's logical processors, which can be found under system information. When the Julia extension in Studio Visual Code is used, the number of threads can be controlled in the settings. To enable the maximum number of threads available on the machine, the line `"julia.NumThreads": "auto"` should be added to the file. The macro `@batch` allows to evaluate a for-loop on multiple threads.
As an example of the application of `@batch`, the discretised Laplace operator is considered. In Julia, a two-dimensional implementation with a finite-difference formula is implemented.

```
function lap2d!(u, unew)
    M, N = size(u)
    for j in 2:N-1
        for i in 2:M-1
            unew[i,j] = 0.25 * (u[i+1,j] + u[i-1,j] + u[i,j+1] + u[i,j-1])
         end
    end
end

M = 4096
N = 4096
u = zeros(M, N)

%# set boundary conditions
u[1,:] = u[end,:] = u[:,1] = u[:,end] .= 10.0
unew = copy(u);
@btime lap2d!(u, unew)

# 22.967 ms (0 allocations: 0 bytes)
```

The function is multithreaded by adding the macro `@batch` to the most outer loop. The parallelisation results in a speed up as can be concluded from the following code.

```
function lap2d_parallel!(u, unew)
    M, N = size(u)
    @batch for j in 2:N-1
        for i in 2:M-1
            unew[i,j] = 0.25 * (u[i+1,j] + u[i-1,j] + u[i,j+1] + u[i,j-1])
         end
    end
end

M = 4096
N = 4096
u = zeros(M, N)

# set boundary conditions
u[1,:] = u[end,:] = u[:,1] = u[:,end] .= 10.0
unew = copy(u);
@btime lap2d_parallel!(u, unew)

# 15.868 ms (0 allocations: 0 bytes)
```

# 6 Computational Performance Analysis

This chapter compares the previously described methodologies in terms of computation speed and memory usage. Here, memory usage refers specifically to heap memory usage. Recall that the different approaches are:

1. Uniform basis functions with point matching at the centres of the mesh elements.
2. Uniform basis functions with the average formulation.
3. Linear basis functions with point matching at the vertices of the mesh elements, shifted to the centre by $\epsilon_{shift}$.
4. Linear basis functions with a Galerkin method.

Moreover, an in-depth analysis of the performance of the MoM implementation using uniform basis functions and point matching is conducted. This analysis aims to estimate the potential benefits of using Julia for TNO's numerical simulations. Comparable to MATLAB, Julia has a built-in operator to solve matrices, the '$\backslash$'-operator. Since this research does not dive into optimising the system-solving process, this macro is used to obtain the solution. A distinction is made between the time required to obtain the matrix and solution vector of the linear system and the time required to solve this system. From now on, there will be referred to these times as "assembly time" and "solve time", respectively. For the other methods, the assembly and solve time will be compared for one mesh.The same geometry is used as described in Section 3.7. Thus, a linearly reacting plate of $10 \times 10 \times 0.002\ \mathrm{m}$, with magnetic susceptibility $\chi_m = 100$ is considered. The next section explains how the external magnetic flux density is chosen. Following that, the results on the performance of the MoM implementation using uniform basis functions and point matching are shown and compared to TNO's current implementation. Lastly, the computation speed and memory usage of the remaining three methods are presented.

## 6.1. Determine External Magnetic Flux Density

As mentioned in Chapter 3, $\mathbf{B}_{\text{ext}}$ equals the Earth's magnetic flux density. In the previous chapter, the external flux density was chosen to be only nonzero in the x-component. To make the comparison as accurate as possible, $\mathbf{B}_{\text{ext}}$ equals Earth's magnetic flux density in the Dutch North Sea. The magnitude of the magnetic flux density in this region equals approximately $50\ \mu\mathrm{T}$ [41]. With help of the magnetic declination and inclination, denoted by $D$ and $I$ respectively, the magnetic flux density in the x-, y-, and z-direction can be found. The magnetic declination is the difference in direction between geographic and magnetic North [42]. Magnetic inclination is the angle at which the geomagnetic field is tilted relative to the surface of the Earth [2]. Figure 6.1 visualises these angles. In this figure, $F$ denotes the total magnetic flux intensity and $H$ denotes horizontal component of $F$.

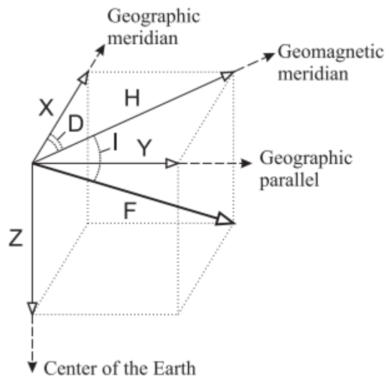


**Figure 6.1:** Decomposition of Earth's magnetic flux density [2].

At the Dutch North Sea, the magnetic declination and inclination equals approximately $2°$ and $68°$ respectively [43]. Using these angles and the fact that $F = 50$ µT, the external magnetic flux density in the x-, y- and z-components can be calculated as follows

$$\mathbf{B}_{\text{ext}} = \begin{pmatrix} B_{\text{ext}_x} \\ B_{\text{ext}_y} \\ B_{\text{ext}_z} \end{pmatrix}, \tag{6.1}$$

$$= \begin{pmatrix} H\cos(D) \\ H\sin(D) \\ F\sin(I) \end{pmatrix}, \tag{6.2}$$

$$H = F\cos(I) \Rightarrow = \begin{pmatrix} F\cos(I)\cos(D) \\ F\cos(I)\sin(D) \\ F\sin(I) \end{pmatrix}, \tag{6.3}$$

$$\approx \begin{pmatrix} 19 \\ 0.7 \\ 46 \end{pmatrix} \text{µT} \tag{6.4}$$

## 6.2. Julia Implementation vs. TNO's MATLAB Implementation

To evaluate the value of Julia for the MoM, TNO's existing implementation is compared to the implementation in Julia. Remember that currently TNO uses uniform basis function and point matching. Therefore, this section focuses on the implementation that uses uniform basis functions and point matching. Various meshes of the geometry are analysed. The table below shows the corresponding number of mesh elements for each value of $lc$.

| Mesh *k* | *lc* | **Number of mesh elements** |
|---|---|---|
| 1 | 10 | 4 |
| 2 | 1.0 | 244 |
| 3 | 0.70 | 544 |
| 4 | 0.60 | 690 |
| 5 | 0.50 | 936 |
| 6 | 0.40 | 1474 |
| 7 | 0.34 | 2122 |
| 8 | 0.32 | 2400 |
| 9 | 0.30 | 2738 |
| 10 | 0.28 | 3048 |
| 11 | 0.26 | 3536 |
| 12 | 0.24 | 4132 |
| 13 | 0.22 | 4912 |
| 14 | 0.20 | 5828 |
| 15 | 0.18 | 7322 |

**Table 6.1:** Table of the value of $lc$ and the number of mesh elements per mesh of the plate.

Remember that the size of the obtained linear system equals $3N \times 3N$, where $N$ is the number of mesh elements. The external magnetic flux density is as described in the previous section.

### 6.2.1. Obtained Solution for $\tilde{\mathrm{M}}$

Before the computational performance is analysed, it is verified that the solution for the magnetisation using the implementation in Julia give similar results as TNO's implementation in MATLAB. Two metrics are used to measure the difference between the two vectors. Denote the difference vector as $\mathbf{x} \in \mathbb{R}^n = [x_1, x_2, \dots, x_n]^T$ for some $n \in \mathbb{N}$. First of all, the $l^\infty$-norm is used. This norm is also called the maximum norm. It takes the form [44, p. 40]

$$||\mathbf{x}||_\infty = \max(|x_1|, |x_2|, \dots, |x_n|). \tag{6.5}$$

Observe that this error metric provides the worst-case scenario by focusing on the largest absolute value within the difference vector. To complement the maximum norm, the second error metric used is the Root Mean Square Error (RMSE). It provides an overall picture of the difference by considering all vector elements. The RMSE takes the form [45]

$$RMSE(\mathbf{x}) = \sqrt{\frac{\sum_{i=1}^{N}(x_i)^2}{N}}. \tag{6.6}$$

The figure below shows both the maximum error and the RMSE between the solution for the magnetisation using the implementation in Julia and TNO's implementation in MATLAB.

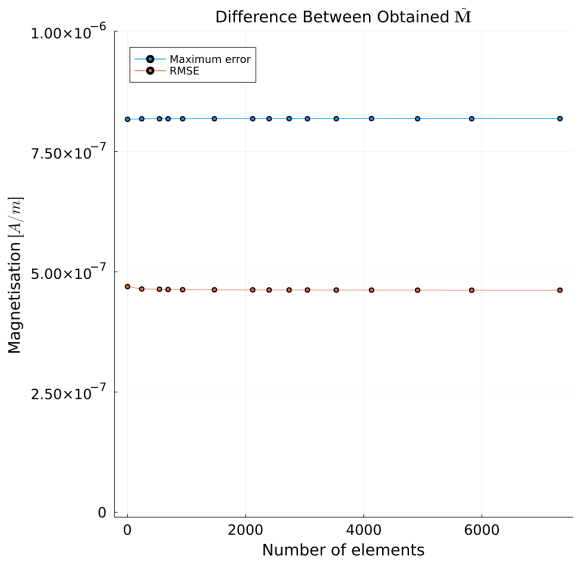
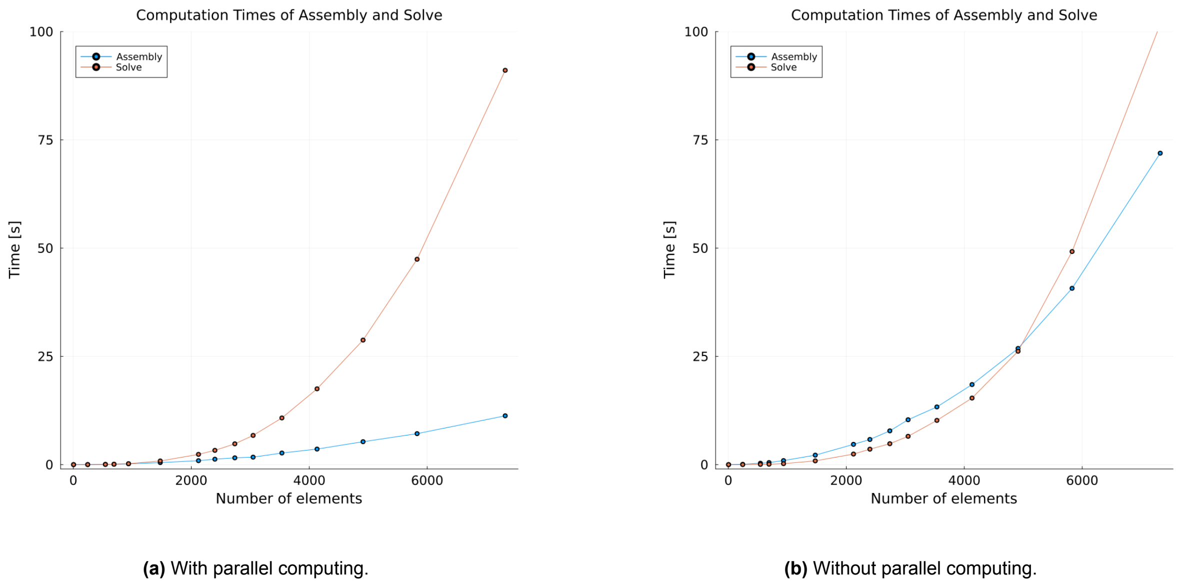


**Figure 6.2:** The difference between the obtained magnetisation using uniform basis functions and point matching.

Figure 6.2 shows that the difference between the solutions obtained using the Julia implementation and TNO's MATLAB implementation is negligibly small.

### 6.2.2. Computation Speed

The computation time of the assembly and solve step are found for various values of $lc$. Figure 6.3 shows the computation times of the implementation in Julia plotted against the number of elements in the mesh. Since TNO does not use parallel computing, the computation times are presented for both scenarios: with and without parallel computing. In Figure 6.3a, the sub-blocks of the interaction matrix are computed in parallel, whereas Figure 6.3b shows the results without parallel computing.



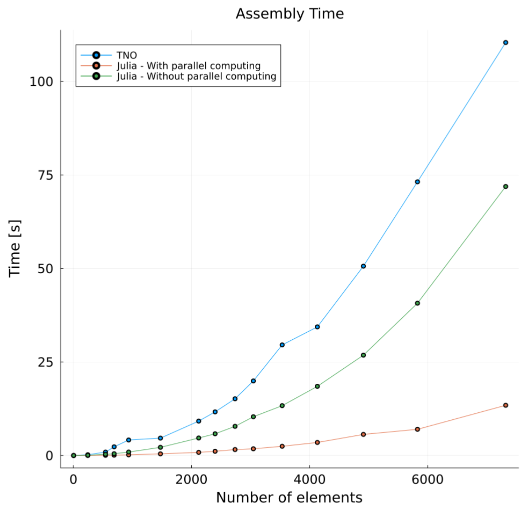
**(a)** With parallel computing.

**(b)** Without parallel computing.

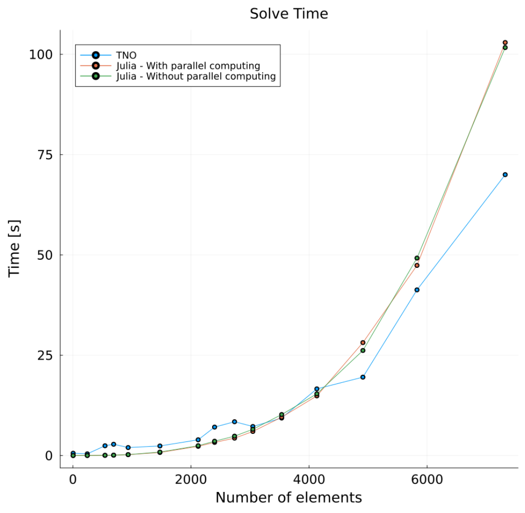
**Figure 6.3:** The assembly and solve time using uniform basis functions and point matching in Julia.

A significant difference in assembly times can be observed between the parallel and non-parallel computing scenarios. In both scenarios, there exists a point where the assembly of the matrix $[\mathbf{C}]$ becomes faster than solving the linear system with the '$\backslash$'-operator. The use of parallel computing or not should not have a significant influence on the solve time, since the macro `@batch` is not used to solve the linear system. Therefore, there is not much of a difference expected between the solve time with and without parallel computing, which is also shown in Figure 6.3.

The computation times for the Julia implementation are compared with TNO's MATLAB implementation. To minimise variability in TNO's computation times, the MATLAB code is executed four times, and the average times are recorded. The following figure shows the results.



**(a)** Assembly time.



**(b)** Solve time.

**Figure 6.4:** The assembly and solve time using uniform basis functions and point matching.

From Figure 6.4a, it can be concluded that the implementation in Julia outperforms TNO's current implementation in terms of the computation speed of the assembly step, regardless of whether parallel computing is used. For meshes up to approximately size 4000, the solve times are similar. However, for larger meshes, MATLAB solves the system faster.

### 6.2.3. Memory Usage

Also, the number of heap allocations is measured for the meshes. Unfortunately, MATLAB does not offer a straightforward method for benchmarking heap allocations during the simulation. Therefore, this section only focuses on the implementation in Julia. Figure 6.5 shows the heap memory use for the different meshes. The y-axis shows the memory use in kibibytes. Note that $1$ kibibyte equals $1024$ bit [46].

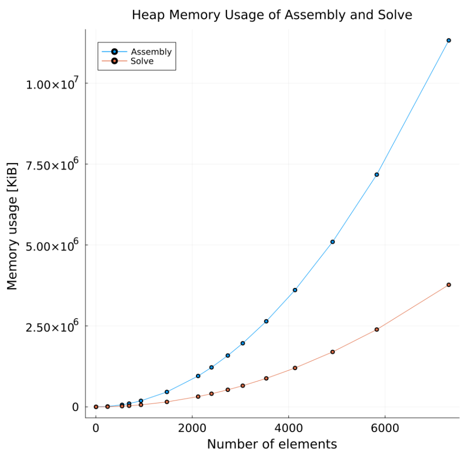


**Figure 6.5:** The heap memory usage using uniform basis functions and point matching in Julia

This graph shows that the assembly step takes more heap memory than the solve step.

## 6.3. Comparison of the Four Methods

In this section, the four different approaches, discussed in the previous chapters, are compared in terms of computation speed and memory use. To do so, only one mesh is considered, namely the plate as described before with $lc = 0.3$. This mesh consists of 1438 nodes and 2878 elements. The relative tolerance in the function `hcubature()` using uniform basis functions is set to $10^{-6}$ and using linear basis functions is set to $10^{-5}$. The table below shows the computation time and memory usage of the different methods for this mesh.

| | **Time [s]** | | **Memory Usage [KiB]** | |
|---|---|---|---|---|
| | **Assembly** | **Solve** | **Assembly** | **Solve** |
| Uniform - Point matching | $1.41$ | $4.95$ | $1.6 \cdot 10^6$ | $5.3 \cdot 10^5$ |
| Uniform - Average formulation | $955.17$ | $4.31$ | $3.0 \cdot 10^7$ | $5.3 \cdot 10^5$ |
| Linear - Point matching | $225.83$ | $0.68$ | $4.0 \cdot 10^7$ | $1.5 \cdot 10^5$ |
| Linear - Galerkin method | $4248.49$ | $0.66$ | $2.1 \cdot 10^9$ | $1.5 \cdot 10^5$ |

**Table 6.2:** Overview of the assembly and solve time of the different methods

The implementation of the assembly of the interaction matrix using uniform basis functions and point matching is the most efficient implementation in terms of computation speed and memory use. This efficiency is expected since avoiding allocation speeds up the implementation. With linear basis functions and point matching, heap memory allocation is affected by how the shift of evaluation points is handled. In the average formulation and the Galerkin method, the high heap allocation is due to the numerical integration process.
Note that the amount of memory use to solve the system, only depends on the size of the system. This explains the comparable solve times between the methods when the same set of basis functions is used. Since the number of nodes is less than the number of mesh elements, the linear system obtained by using linear basis functions is smaller than the linear system obtained by using uniform basis functions. This explains the difference in solve times. Due to the errors in the numerical integration process, the Galerkin method does not lead to a symmetric system yet. When rounding down to a specific number of significant figures, the interaction matrix, and consequently the linear system, becomes symmetric. This is advantageous as symmetric matrices possess properties that simplify the solving process.

# 7 Verification

In this chapter, the four different methods are verified. Verification is the process of answering the question whether the implementation is correct. For the implementation that uses uniform basis functions and point matching it is checked that the computed solution converges to the analytical solution of the spherical shell, which presented in Appendix A. For the remaining three methods, a mesh convergence analysis is conducted .

## 7.1. Accuracy

In this section, the performance in terms of accuracy of one approach is analysed, namely using uniform basis functions and point matching. The geometry is a spherical shell with outer radius 50 $\mathrm{m}$ and inner radius 49.998 $\mathrm{m}$. Denote the outer and inner radii as $b$ and $a$ respectively. In Appendix A, analytical solutions for $\mathbf{M}$ and $\mathbf{B}_{\text{red}}$ are given when a spherical shell is placed in a uniform magnetic field in the z-direction. The assumption is made that the external flux density equals

$$\mathbf{B}_{\text{ext}}(\mathbf{r}) = 50 \cdot 10^{-6}\mathbf{u}_z. \tag{7.1}$$

The analytical solutions for both $\mathbf{M}$ and $\mathbf{B}_{\text{red}}$ are compared to the numerical approximation for one geometry with various meshes.

### 7.1.1. Meshing of Spherical Shell

With help of Gmsh, a hollow sphere with radius 50 $\mathrm{m}$ is created. Thereafter, a 2D mesh is generated on the spherical geometry. Again, the parameter $lc$ is used to define the mesh size. The mesh is extruded towards the centre of the sphere by $h = b - a = 0.002\ \mathrm{m}$.

Note that this meshing approach does not produce prisms with congruent triangular faces as mesh elements. As a result, this complicates the integration over the volume. Additionally, the linear basis functions defined in Chapter 4 are not applicable to these elements, as the functions lose their invariance in the thickness direction of the element. Therefore, this section only focuses on the use of uniform basis functions and point matching. After the meshing is done, the volume of the mesh is calculated, which is denoted by $V_{mesh}$. Also, the volume of the shell is analytically calculated

$$\begin{aligned} V_{analytical} &= V_{outer} - V_{inner}, && (7.2)\\ &= \frac{4}{3}\pi b^3 - \frac{4}{3}\pi a^3, && (7.3)\\ &= \frac{4}{3}\pi(b^3 - (b-h)^3). && (7.4)\end{aligned}$$

To ensure an accurate comparison between the analytical and numerical solution, $b$ is transformed to $b'$, such that the volume of the mesh equals the volume of analytical spherical shell. $b'$ can be found as follows

$$V_{mesh} = \frac{4}{3}\pi(b'^3 - (b'-h)^3), \tag{7.5}$$

$$V_{mesh} = 4\pi h(b'^2 - b'h + \frac{h^2}{3}), \tag{7.6}$$

$$b'^2 - b'h + \frac{h^2}{3} - \frac{V_{mesh}}{4\pi h} = 0, \tag{7.7}$$

$$b' = \frac{h + \sqrt{\frac{V_{mesh}}{\pi h} - \frac{h^2}{3}}}{2} \quad \vee \quad b' = \frac{h - \sqrt{\frac{V_{mesh}}{\pi h} - \frac{h^2}{3}}}{2}, \tag{7.8}$$

$$b' \text{ must be positive and larger than } h \quad \Rightarrow b' = \frac{h + \sqrt{\frac{V_{mesh}}{\pi h} - \frac{h^2}{3}}}{2}. \tag{7.9}$$

Denote the transformed inner radius as $a' = b' - h$.

### 7.1.2. Comparison for $\tilde{\mathbf{M}}$

The approximated magnetisation at the centre of the mesh elements is compared to the analytical magnetisation at the middle of the shell's material. Suppose the mesh consists of $N$ mesh elements. First, the centres of the mesh elements are computed, denoted by $\mathbf{r}_k$ for $k = 1, \dots, N$. A vector pointing from the centre of the spherical shell to the centre of each mesh element is determined and normalised. Observe that the distance from the centre of the shell to the centre of the material is given by $a + \frac{h}{2}$. By multiplying this vector by $a + \frac{h}{2}$, the point inside the spherical shell is determined, denoted by $\mathbf{r}'_k$, where the magnetisation is estimated by $\tilde{\mathbf{M}}(\mathbf{r}_k)$ for $k = 1, \dots, N$. Repeating this for every element, two vectors of size $3N$ are obtained, $\tilde{\mathbf{M}}$ and $\mathbf{M}$. Figure 7.1 gives a clearer visualisation of the two points that are compared for an arbitrary mesh element $k$. A cross-section of the spherical shell is depicted. The red object represents the mesh element $k$.

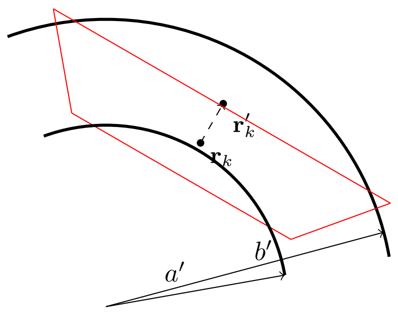


**Figure 7.1:** Two points $\mathbf{r}_k$ and $\mathbf{r}'_k$ where the magnetisation is compared for an arbitrary mesh element.

### 7.1.3. Comparison for $\tilde{\mathbf{B}}_{\text{red}}$

The estimated reduced magnetic flux density is compared to the analytical reduced magnetic flux density in the x-y plane at $z = 100\ \text{m}$. Both $x$ and $y$ range from $-60\ \text{m}$ to $60\ \text{m}$. 120 points are uniformly distributed along both the x- and y-direction, which results in a total of 14400 points. The points are visualised in the figure below.

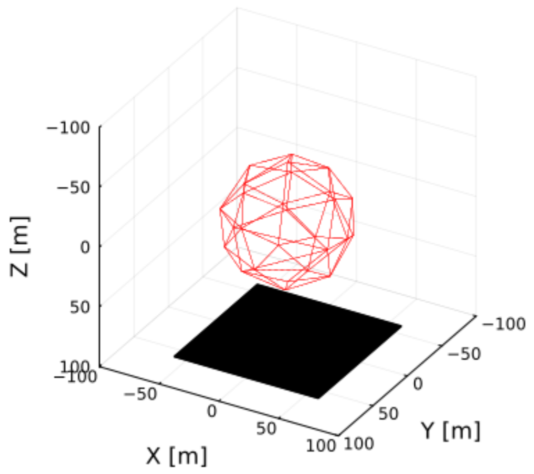


**Figure 7.2:** Visualization of the points that are used to compare $\tilde{\mathbf{B}}_{\text{red}}$

In this array of points, the reduced magnetic flux density resulting from the estimated magnetisation of the spherical shell is compared to the analytical solution for the reduced flux density.

### 7.1.4. Results

The approximated magnetisation and reduced magnetic flux density, calculated using uniform basis functions and point matching, is compared to the analytical solution for the magnetisation for multiple meshes. Table 7.1 provides an overview of the meshes.

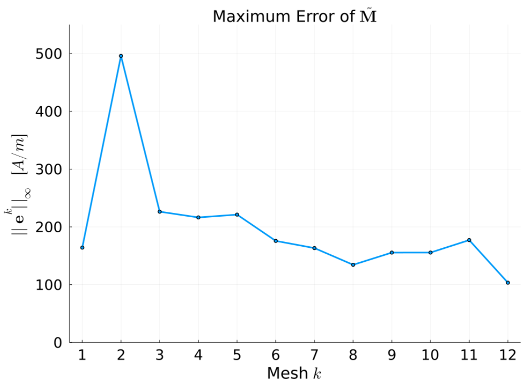
| Mesh *k* | *lc* | Number of mesh elements |
|---|---|---|
| 1 | 0.50 | 1138 |
| 2 | 0.40 | 1642 |
| 3 | 0.38 | 1792 |
| 4 | 0.36 | 2110 |
| 5 | 0.34 | 2258 |
| 6 | 0.32 | 2620 |
| 7 | 0.30 | 2970 |
| 8 | 0.28 | 3376 |
| 9 | 0.26 | 3790 |
| 10 | 0.24 | 4452 |
| 11 | 0.22 | 5452 |
| 12 | 0.20 | 6518 |

**Table 7.1:** Table of the value of $lc$ and the number of mesh elements for the mesh of the spherical shell.

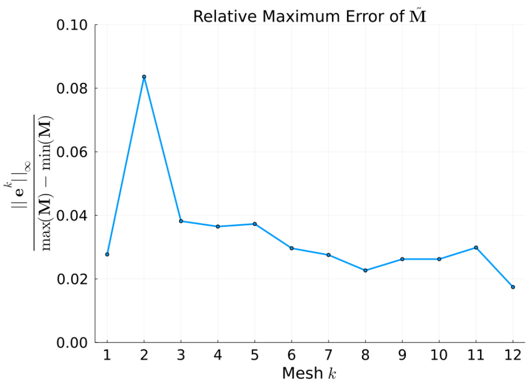
The vector $\mathbf{M} - \tilde{\mathbf{M}}^k$ is denoted by $\mathbf{e}^k$. The normalised RMSE (NRMSE) is calculated as follows

$$NRMSE(\mathbf{e^k}) = \frac{RMSE(\mathbf{e^k})}{\max(\mathbf{M}) - \min(\mathbf{M})}. \tag{7.10}$$

Figure 7.3 and Figure 7.4 show the results on the accuracy.

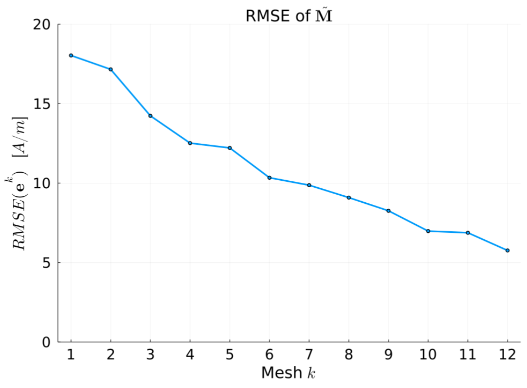


(a) Maximum error.

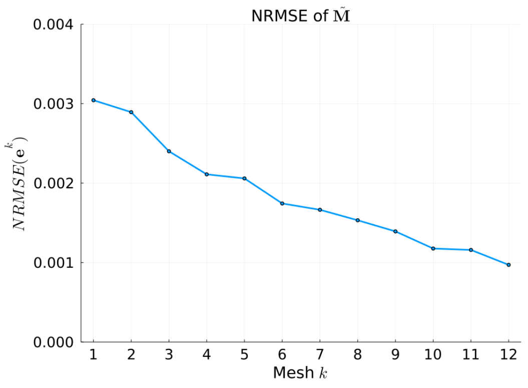


(b) Relative maximum error.

**Figure 7.3:** The maximum error of the approximated magnetisation using uniform basis functions and point matching.



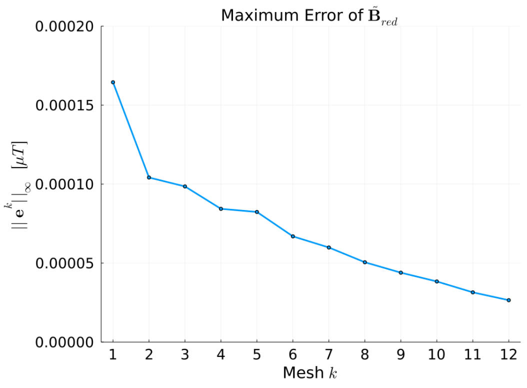
(a) RMSE.



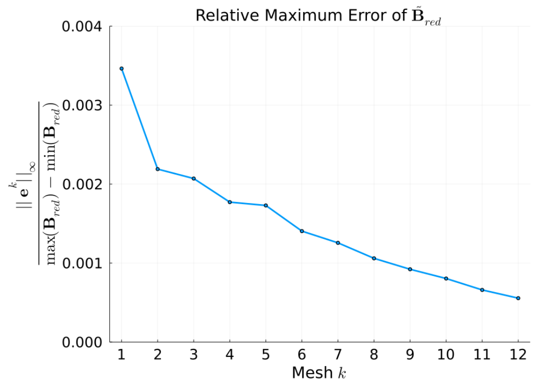
(b) NRMSE.

**Figure 7.4:** The RMSE of the approximated magnetisation using uniform basis functions and point matching.

Based on these figures, it can be concluded that the numerical solution converges to the analytical solution as the mesh becomes finer. The error, measured by the RMSE, shows a monotone decreasing trend. Similar graphs are made for the approximation of the reduced magnetic flux density. Now, $\mathbf{e^k}$ denotes the vector $\mathbf{B}_{\text{red}} - \tilde{\mathbf{B}}_{\text{red}}$.

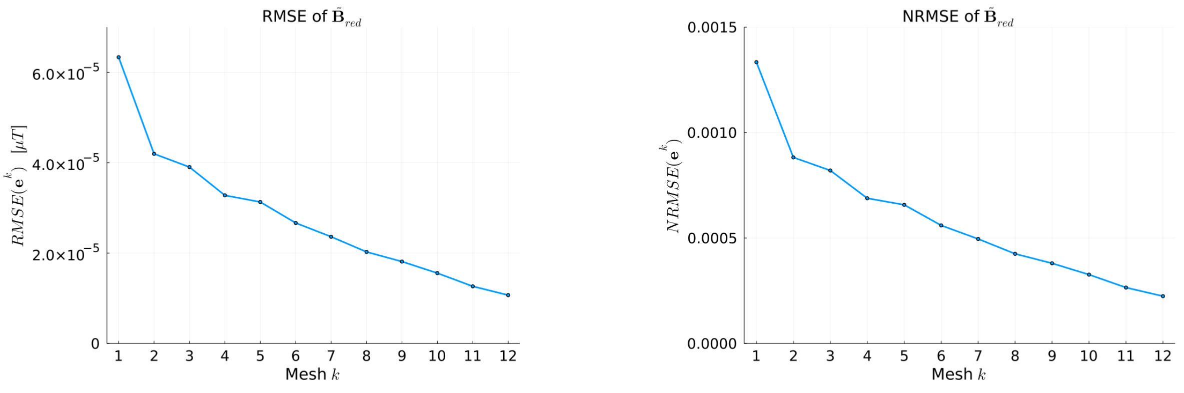


(a) Maximum error.



(b) Relative maximum error.

**Figure 7.5:** The maximum error of the approximated reduced magnetic flux density using uniform basis functions and point matching.



**(a)** RMSE. **(b)** NRMSE.

**Figure 7.6:** The RMSE of the estimated reduced magnetic flux density using uniform basis functions and point matching.

Figure 7.5 and Figure 7.6 demonstrate that both the maximum error and the RMSE show a monotonically decreasing trend, converging towards zero.
These results validate the implementation of the MoM using point matching in Julia.

## 7.2. Convergence

Since verifying the performance in terms of accuracy using the analytical solution of a spherical shell for all four methods was not possible, a mesh convergence analysis is conducted. Mesh convergence analysis helps verify the accuracy of the numerical solution. By refining the mesh and observing how the solution changes, it can be ensured that the results are approaching the true solution of the physical problem. The geometry is again the linearly reacting plate of $10 \times 10 \times 0.002$ m, with magnetic susceptibility $\chi_m = 100$. The external uniform magnetic flux density equals

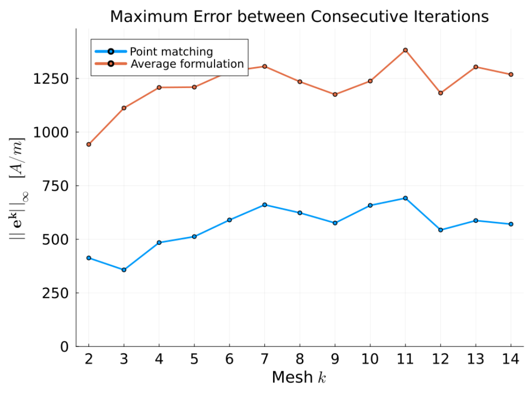
$$\mathbf{B}_{\text{ext}}(\mathbf{r}) = 50 \cdot 10^{-6}\mathbf{u}_x. \tag{7.11}$$

The mesh that is used to compute the approximation of the magnetisation in the plate is repeatedly refined. The same meshes are used as mentioned in Section 6.2, with the exception of the finest mesh, which is excluded from this analysis. Table 6.1 shows an overview of the iteration and the number of mesh elements. Figure 3.1 visualises meshes for two values of $lc$. After each refinement $k$, the magnetisation, denoted as $\tilde{\mathbf{M}}^k$, is recalculated. For $k > 1$, the newly computed magnetisation is compared to the approximated magnetisation obtained from the mesh of the previous iteration, which is denoted by $\tilde{\mathbf{M}}^{k-1}$. On the plate, 50 points are uniformly distributed along both the x- and y-direction, excluding the boundary. This results in a total of 2500 points, where the estimated magnetisation is computed. The vector $\tilde{\mathbf{M}}^k - \tilde{\mathbf{M}}^{k-1}$ is denoted as $\mathbf{e}^\mathbf{k}$. The difference between the estimated magnetisation from two consecutive refinements is calculated using the maximum error and the RMSE. In this section the RMSE is normalised as follows

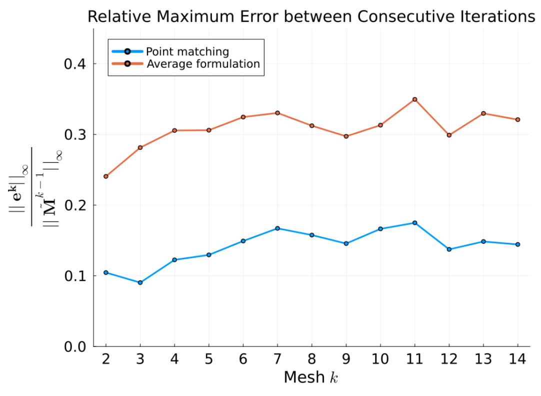
$$NRMSE(\mathbf{e}^\mathbf{k}) = \frac{RMSE(\mathbf{e}^\mathbf{k})}{\max(|\mathbf{e}^\mathbf{k}|) - \min(|\mathbf{e}^\mathbf{k}|)}. \tag{7.12}$$

### 7.2.1. Uniform Basis Functions

In this subsection, the convergence of the MoM for finding $\tilde{\mathbf{M}}$ using uniform basis functions is analysed. Figure 7.7 displays both the maximum and the relative maximum error of the approximated magnetisation between two consecutive meshes.



(a) Maximum error.



(b) Relative maximum error.

**Figure 7.7:** Maximum error and relative maximum error of the solutions for consecutive meshes using uniform basis functions.

From Figure 7.7, it can be concluded that the maximum norm of the difference in the solution between two consecutive meshes does not converge for these meshes. To gain insight into this issue, the positions on the plate where the maximum error occurs between two consecutive refinements are illustrated in the following figure.

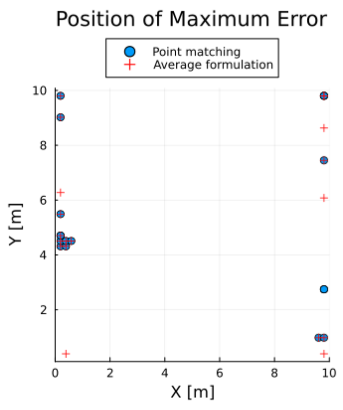
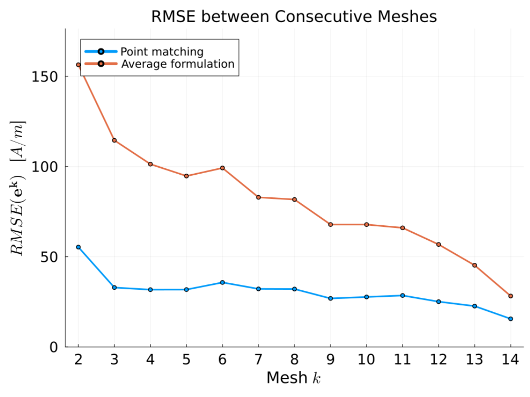
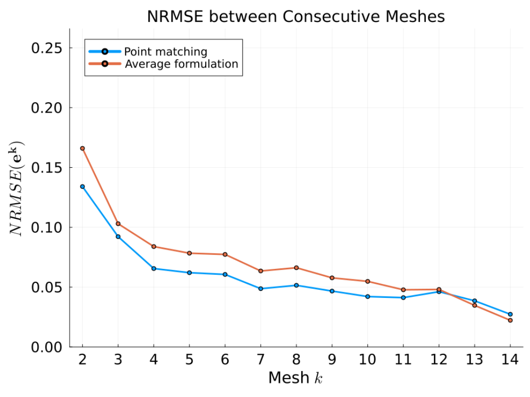


**Figure 7.8:** Positions within the plate where the maximum error occurs between two consecutive meshes using uniform basis functions.

Observe that the maximum error occurs close to the boundary of the geometry. Experiments show that, for this external field, the x-component of the magnetisation close to the boundary decreases as the mesh refines. Figure 7.9 show both the RMSE and NRMSE between two consecutive meshes.



(a) RMSE.



(b) NRMSE.

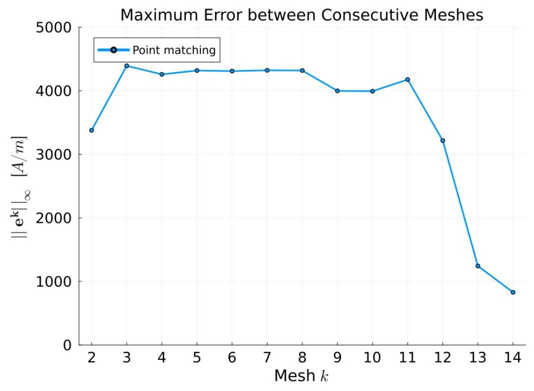
**Figure 7.9:** RMSE and NRMSE of the solutions for consecutive meshes using uniform basis functions.

Figure 7.9 demonstrates that both methods converge.

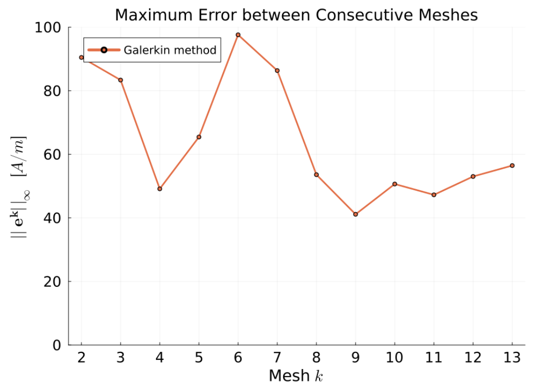
### 7.2.2. Linear Basis Functions

In this subsection, the convergence of the MoM for finding $\tilde{\mathbf{M}}$ using linear basis functions is analysed. The convergence of point matching and the Galerkin method are presented in separate figures. It is important to note that the convergence analysis for the Galerkin method does not include all the meshes.

Figure 7.10 and Figure 7.11 show the maximum error and relative maximum error, respectively, of the difference in the solution between two consecutive meshes using both point matching and the Galerkin method.

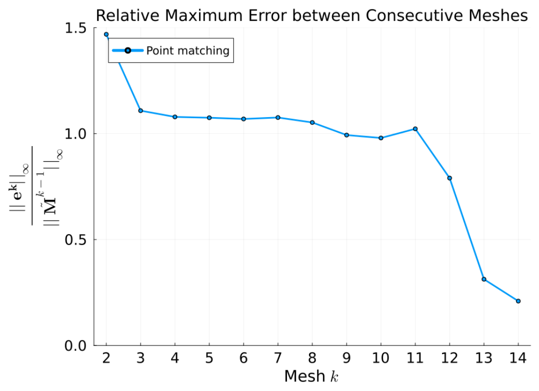


**(a)** Point matching.

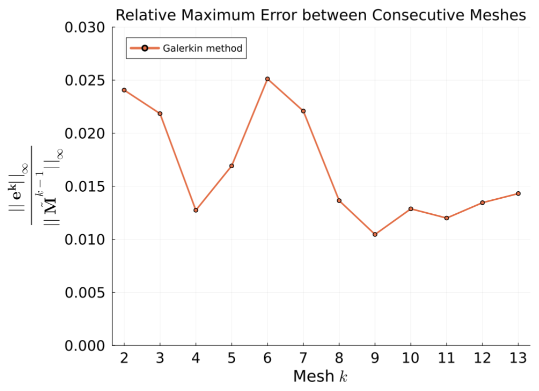


**(b)** Galerkin method.

**Figure 7.10:** Maximum error of the solutions for consecutive meshes using linear basis functions.



**(a)** Point matching.



**(b)** Galerkin method.

**Figure 7.11:** Relative maximum error of the solutions for consecutive meshes using linear basis functions.

Even though Figure 7.10a and Figure 7.11a show a decrease in both the maximum error and relative maximum error between two consecutive solutions as the mesh refines, the Galerkin method shows a significantly smaller maximum error and relative maximum error. However, Figure 7.10b and 7.11b show both errors do not converge to zero for these meshes. The positions on the plate where the maximum error occurs between two consecutive refinements are illustrated in the figure below.

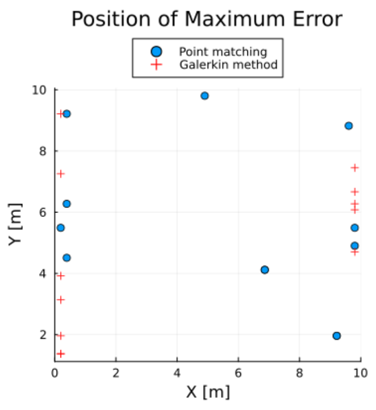
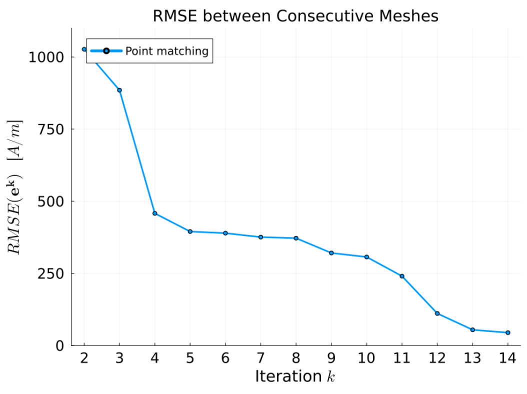
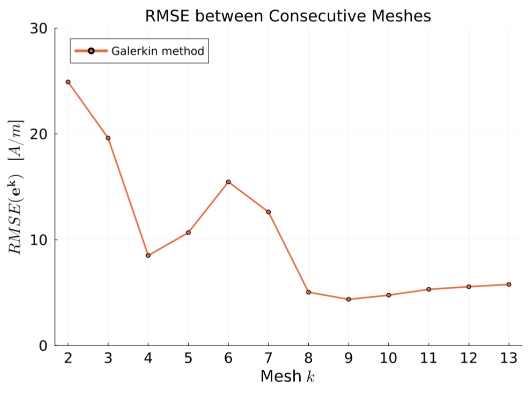


**Figure 7.12:** Positions within the plate where the maximum error occurs between two consecutive iterations using linear basis functions.

Observe in Figure 7.12 that the maximum difference of the solutions of two consecutive refinements using point matching also occurs occurs in the centre of the geometry. From a physical perspective, it is known that for this external magnetic flux density, the magnetisation in the internal part of the plate is uniform. Therefore, it is expected that the approximated magnetisation should not show significant differences when the mesh is refined. The maximum difference in the solutions of two consecutive refinements using the Galerkin method occurs near the boundary. Therefore, Figure 7.13 and Figure 7.14 show the RMSE and NRMSE, respectively, of the difference in the solution between two consecutive meshes using both point matching and the Galerkin method.

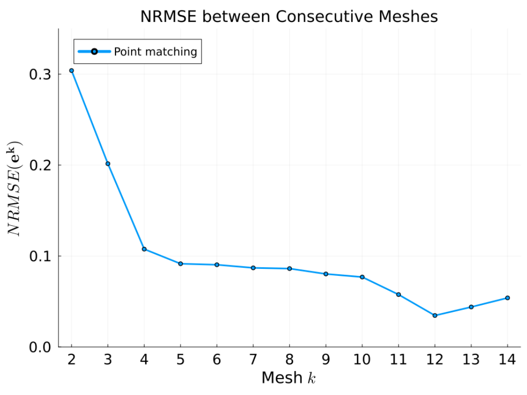


**(a)** Point matching.

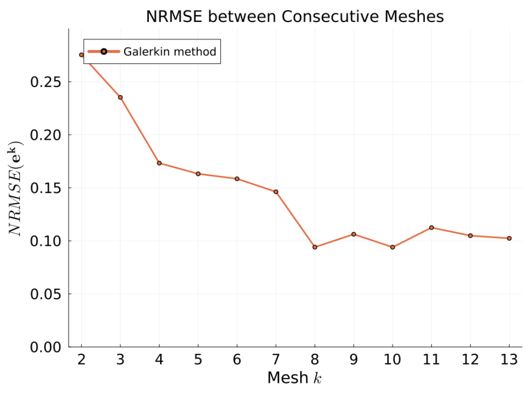


**(b)** Galerkin method.

**Figure 7.13:** RMSE of the solutions for consecutive meshes using linear basis functions.



**(a)** Point matching.



**(b)** Galerkin method.

**Figure 7.14:** NRMSE of the solutions for consecutive meshes using linear basis functions.

Figure 7.13 and Figure 7.14 show that both the RMSE and the NRMSE of the difference in the solution between two consecutive meshes using the Galerkin method is smaller than when using point matching. Based on this mesh convergence analysis, the Galerkin method is preferred over point matching.

# 8 The Use of Automatic Differentiation in Method of Moments

Consider a 3D-domain $\tau_M$ made of magnetic material exposed to an external magnetic flux density. To determine the elements of the interaction matrix in the MoM, it is essential to derive an expression for the following equation

$$\tilde{\mathbf{B}}_{\text{red}}(\mathbf{r}) = \nabla \times \left( \frac{\mu_0}{4\pi} \iiint\limits_{\tau_M} \frac{\tilde{\mathbf{M}}(\mathbf{r}') \times (\mathbf{r} - \mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||^3} \, \mathrm{d}\mathbf{r}' \right), \tag{8.1}$$

where $\tilde{\mathbf{M}}(\mathbf{r}) = \sum_{i=1}^{N} \tilde{\mathbf{M}}^i \phi_i(\mathbf{r})$ for a set of basis functions $\{\phi_i\}_{i=1}^N$ and coefficients $\{\tilde{\mathbf{M}}^i\}_{i=1}^N$. The methodologies discussed thus far involve interchanging the integration and differentiation, followed by finding an analytical expression. Differentiating the integrand increases its singularity. It is expected that first integrating with respect to $\mathbf{r}'$ and then differentiating with respect to $\mathbf{r}$ prevents an increase in the order of singularity of the integration kernel. This approach simplifies the numerical integration with respect to $\mathbf{r}'$ using adaptive numerical methods. Therefore, it is beneficial to first derive an expression for the integral and then determine the curl by taking its derivatives.
Three approaches can automate the calculation of derivatives: numerical differentiation using finite difference approximations, symbolic differentiation, and automatic differentiation (AD) [47]. Finite difference methods are vulnerable for truncation and rounding errors, and instability [48]. Symbolic differentiation may lead to costly evaluations of the derivative [49]. AD offers the benefits of precision and user-friendliness, while the computation speed is comparable to hand-coding derivatives [47] [50]. Another advantage of using AD in the MoM is the potential to establish a more general framework. This approach would make it easier to extend the MoM to different sets of basis functions and simplify the implementation process.
This chapter introduces AD with the help of dual numbers. To get a better understanding, an example of forward AD is provided. Next, the use of the Julia package `ForwardDiff.jl`, which enables automatic differentiation, is demonstrated. Finally, the derivation of the interaction matrix in the MoM using AD is presented.

## 8.1. Automatic Differentiation using Dual Numbers

One way to implement AD is using dual numbers. A dual number is a number of the form $a + \epsilon b$, where $a, b \in \mathbb{R}$, $\epsilon \neq 0$ and $\epsilon^2 = 0$ [33]. The addition, multiplication and division operation are straightforward

$$
\begin{aligned}
z_1 + z_2 &= (a + \epsilon b) + (c + \epsilon d), && (8.2)\\
&= a + c + \epsilon(b + d), && (8.3)\\
z_1 \cdot z_2 &= (a + \epsilon b) \cdot (c + \epsilon d), && (8.4)\\
&= ac + \epsilon(ad + bc) + \epsilon^2 bd, && (8.5)\\
&= ac + \epsilon(ad + bc), && (8.6)\\
\frac{z_1}{z_2} &= \frac{a + \epsilon b}{c + \epsilon d} \quad \text{(where } c \neq 0\text{)}, && (8.7)\\
&= \frac{(a + \epsilon b)(c - \epsilon d)}{(c + \epsilon d)(c - \epsilon d)}, && (8.8)\\
&= \frac{ac - \epsilon ad + \epsilon bc - \epsilon^2 bd}{c^2 - \epsilon d^2}, && (8.9)\\
&= \frac{a}{c} + \epsilon \frac{bc - ad}{c^2}. && (8.10)
\end{aligned}
$$

The relation between dual numbers and differentiation becomes clear when looking at the Taylor expansion. Given an arbitrary real function $f : \mathbb{R} \to \mathbb{R}$. The Taylor series for the function $f(z_1)$ at the point $a$ is given by [51, p. 159]

$$
\begin{aligned}
f(a + \epsilon b) &= \sum_{n=0}^{\infty} \frac{f^{(n)}(a)(a + \epsilon b - a)^n}{n!}, && (8.11)\\
&= \sum_{n=0}^{\infty} \frac{f^{(n)}(a)\epsilon^n b^n}{n!}, && (8.12)\\
&= f(a) + bf'(a)\epsilon. && (8.13)
\end{aligned}
$$

The same can be done for multi-variate functions. Given an arbitrary real function $f : \mathbb{R}^n \to \mathbb{R}$. Several $\epsilon$ are added to each component. These values are stored in $\epsilon \in \mathbb{R}$, where $\epsilon_i^2 = \epsilon_{ij} = 0$. The Taylor expansion at point $\mathbf{a}$ is given by [51, p. 160]

$$f(\mathbf{a} + \epsilon b) = f(\mathbf{a}) + b\nabla f(\mathbf{a})\epsilon. \tag{8.14}$$

For a function $f : \mathbb{R}^n \to \mathbb{R}^m$, instead of using a vector $\epsilon$, a matrix $\epsilon$ is used. For each row in the matrix, Equation 8.14 is applied [33]. The Taylor expansions allows to extend the elementary functions to the set of dual numbers. This equation allows to extend elementary functions to the set of dual numbers. For example,

$$\sin(z_1) = \sin(a + \epsilon b) = \sin(a) + b\cos(a)\epsilon. \tag{8.15}$$

### 8.1.1. Example of Forward Automatic Differentiation

Consider an arbitrary function $f : \mathbb{R}^n \to \mathbb{R}^m$for which all elements of the Jacobian matrix need to be computed. AD operates in two modes: forward mode and reverse mode. In [47] is mentioned, that forward mode is more efficient when $n \leq m$. Given that the integral shown in Equation 8.1 maps from $\mathbb{R}^3$ to $\mathbb{R}^3$, forward mode is considered in this research. This section illustrates forward AD with an example. Suppose that the gradient at $(0, 1)$ of the following multivariate function has to be computed

$$f(x, y) = xy + \sin(x) + 4. \tag{8.16}$$

A series of arithmetic operations gives an equivalent representation of the function into a code list. The code list for $f(x, y)$ is [52]

$$s_1 = x, \tag{8.17}$$
$$s_2 = y, \tag{8.18}$$
$$s_3 = s_1 \cdot s_2, \tag{8.19}$$
$$s_4 = \sin(s_1), \tag{8.20}$$
$$s_5 = s_3 + s_4, \tag{8.21}$$
$$s_6 = s_5 + 4. \tag{8.22}$$

In [53], a method is described in which the primary AD variables are defined as

$$x = 0 + \epsilon \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \tag{8.23}$$
$$y = 1 + \epsilon \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{8.24}$$

$x$ and $y$ are substituted for $s_1$ and $s_2$.

$$s_1 = 0 + \epsilon \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \tag{8.25}$$
$$s_2 = 1 + \epsilon \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \tag{8.26}$$
$$s_3 = 0 + \epsilon \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \tag{8.27}$$
$$s_4 = 0 + \epsilon \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \tag{8.28}$$
$$s_5 = 0 + \epsilon \begin{pmatrix} 2 \\ 0 \end{pmatrix}, \tag{8.29}$$
$$s_6 = 4 + \epsilon \begin{pmatrix} 2 \\ 0 \end{pmatrix}. \tag{8.30}$$

It can be concluded that $f(0, 1) = 4$ and $\nabla f(0, 1) = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$. By computing $f(0, 1)$ and $\nabla f(0, 1)$ analytically, it can be verified that the results are correct.

## 8.2. Forward Automatic Differentiation in Julia

`ForwardDiff.jl` is a Julia package that provides tools for calculating derivatives, gradients, Jacobians, Hessians, and higher-order derivatives using dual numbers as described in Section 8.1. For computing gradients and Jacobians, the package offers a version of vector-forward mode that eliminates the need for costly heap allocations [48]. The code below returns the gradient of the multivariate function displayed in Equation 8.16.

```julia
using ForwardDiff

function f(x)
    return x[1]*x[2] + sin(x[1]) + 4
end

x = @SVector[0,1]
ForwardDiff.gradient(f, x)
```

The output of the code is

```
2-element SVector{2, Float64} with indices SOneTo(2):
 2.0
 0.0
```

To obtain the interaction matrix in MoM, the curl of a a vector-valued integral has to be computed. First, the Jacobian of the integral is computed. Thereafter, the curl is derived using the entries of the Jacobian. To get a better understanding how this is done in Julia an example is provided. Consider a vector-valued function $g : \mathbb{R}^3 \to \mathbb{R}^3$ defined as follows

$$\mathbf{g}(x, y, z) = \begin{pmatrix} xy \\ yz \\ zx \end{pmatrix} . \tag{8.31}$$

For finding $\nabla \times \mathbf{g}(x, y, z)$ the following code can be used

```
using ForwardDiff

function g(x)
    g1 = x[1]*x[2]
    g2 = x[2]*x[3]
    g3 = x[3]*x[1]
    return @SVector[g1,g2,g3]
end

function compute_curl(J)
    curl_x = J[3,2] - J[2,3]
    curl_y = J[1,3] - J[3,1]
    curl_z = J[2,1] - J[1,2]

    return @SVector[curl_x, curl_y, curl_z]
end

x = @SVector[1.0, 1.0, 1.0]
J = ForwardDiff.jacobian(g, x)
compute_curl(J)
```

The output equals

```
3-element SVector{3, Float64} with indices SOneTo(3):
 -1.0
 -1.0
 -1.0
```

For this minimal working example, it can easily be verified that $\nabla \times \mathbf{g}(1, 1, 1) = \begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix}$.

## 8.3. Derivation of the Interaction Matrix Using Automatic Differentiation

This section describes the application of AD to determine the interaction matrix in the MoM. Observe that the coefficients $\{\tilde{\mathbf{M}}^i\}_{i=1}^N$ in Equation 8.1 are unknown, indicating that the function for the approximated reduced magnetic flux density is a function of $\tilde{\mathbf{M}}$ and $\mathbf{r}$. Considering point matching, it follows from the derivations in Chapter 3 that the interaction matrix $[\mathbf{C}]$ is defined such that

$$[\mathbf{C}]\tilde{\mathbf{M}} = \tilde{\mathbf{B}}_{\text{red}}, \tag{8.32}$$

$$[\mathbf{C}] \begin{pmatrix} \tilde{\mathbf{M}}^1 \\ \vdots \\ \tilde{\mathbf{M}}^N \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{B}}_{\text{red}}(\tilde{\mathbf{M}}, \mathbf{r_1}) \\ \vdots \\ \tilde{\mathbf{B}}_{\text{red}}(\tilde{\mathbf{M}}, \mathbf{r_N}) \end{pmatrix}. \tag{8.33}$$

$$\tag{8.34}$$

By taking the derivative with respect to $\tilde{\mathbf{M}}$, the following expression for the interaction matrix can be obtained

$$\frac{\partial[\mathbf{C}]\tilde{\mathbf{M}}}{\partial\tilde{\mathbf{M}}} = \frac{\partial\tilde{\mathbf{B}}_{\text{red}}}{\partial\tilde{\mathbf{M}}}, \tag{8.35}$$

$$[\mathbf{C}] = \begin{pmatrix} \frac{\partial\tilde{\mathbf{B}}_{\text{red}}(\tilde{\mathbf{M}},\mathbf{r_1})}{\partial\tilde{\mathbf{M}}^1} & \frac{\partial\tilde{\mathbf{B}}_{\text{red}}(\tilde{\mathbf{M}},\mathbf{r_1})}{\partial\tilde{\mathbf{M}}^2} & \cdots & \frac{\partial\tilde{\mathbf{B}}_{\text{red}}(\tilde{\mathbf{M}},\mathbf{r_1})}{\partial\tilde{\mathbf{M}}^N} \\ \frac{\partial\tilde{\mathbf{B}}_{\text{red}}(\tilde{\mathbf{M}},\mathbf{r_2})}{\partial\tilde{\mathbf{M}}^1} & \frac{\partial\tilde{\mathbf{B}}_{\text{red}}(\tilde{\mathbf{M}},\mathbf{r_2})}{\partial\tilde{\mathbf{M}}^2} & \cdots & \frac{\partial\tilde{\mathbf{B}}_{\text{red}}(\tilde{\mathbf{M}},\mathbf{r_2})}{\partial\tilde{\mathbf{M}}^N} \\ \vdots & \cdots & \cdots & \vdots \\ \frac{\partial\tilde{\mathbf{B}}_{\text{red}}(\tilde{\mathbf{M}},\mathbf{r_N})}{\partial\tilde{\mathbf{M}}^1} & \frac{\partial\tilde{\mathbf{B}}_{\text{red}}(\tilde{\mathbf{M}},\mathbf{r_N})}{\partial\tilde{\mathbf{M}}^2} & \cdots & \frac{\partial\tilde{\mathbf{B}}_{\text{red}}(\tilde{\mathbf{M}},\mathbf{r_N})}{\partial\tilde{\mathbf{M}}^N} \end{pmatrix}. \tag{8.36}$$

Observe that $\frac{\partial\tilde{\mathbf{B}}_{\text{red}}(\tilde{\mathbf{M}},\mathbf{r})}{\partial\tilde{\mathbf{M}}^k}$ for $k = 1, \dots, N$, is a $3 \times 3$ matrix. Writing out this expression for some $k = 1, \dots, N$ results in

$$\frac{\partial\tilde{\mathbf{B}}_{\text{red}}(\tilde{\mathbf{M}}, \mathbf{r})}{\partial\tilde{\mathbf{M}}^k} = \frac{\partial}{\partial\tilde{\mathbf{M}}^k}\left(\nabla \times \left(\frac{\mu_0}{4\pi}\iiint\limits_{\tau_M} \frac{\tilde{\mathbf{M}}(\mathbf{r}') \times (\mathbf{r} - \mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||^3}\,\mathrm{d}\mathbf{r}'\right)\right), \tag{8.37}$$

$$= \frac{\partial}{\partial\tilde{\mathbf{M}}^k}\left(\nabla \times \left(\frac{\mu_0}{4\pi}\iiint\limits_{\tau_M} \frac{\sum_{i=1}^N \tilde{\mathbf{M}}^i\phi_i(\mathbf{r}') \times (\mathbf{r} - \mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||^3}\,\mathrm{d}\mathbf{r}'\right)\right), \tag{8.38}$$

$$= \frac{\partial}{\partial\tilde{\mathbf{M}}^k}\left(\nabla \times \left(\frac{\mu_0}{4\pi}\iiint\limits_{\tau_k} \frac{\tilde{\mathbf{M}}^k\phi_k(\mathbf{r}') \times (\mathbf{r} - \mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||^3}\,\mathrm{d}\mathbf{r}'\right)\right). \tag{8.39}$$

An expression for the integral can be determined using the function `hcubature()`, as described in Section 3.5.1. Following this, AD comes into play. Firstly, the curl of the expression is taken with respect to $\mathbf{r}$. Secondly, the Jacobian with respect to $\tilde{\mathbf{M}}^k$ is computed. These final two steps use built-in functions from the Julia package `ForwardDiff.jl`. Observe that this approach simplifies the computation of the magnetic vector potential at any point $\mathbf{r}$, which gives more insight into the problem.

# 9 Method of Moments using Automatic Differentiation

The MoM using AD is implemented in Julia. The implementation assumes uniform basis functions and point matching. This implementation is compared to the implementation of the method as described in Chapter 3, hereafter referred to as Morandi's method, in terms of computation speed, memory use and estimated magnetisation.

## 9.1. Meshing

The same geometry is used as the one described in Section 3.7. Thus, a linearly reacting plate of $10 \times 10 \times 0.002$ m, with magnetic susceptibility $\chi_m = 100$, is considered. However, instead of meshing the 2D surface with triangles, squares are used. The geometry is uniformly meshed in both the x- and y-direction. The squares are extruded in both the positive and negative z-direction, resulting in box-shaped mesh elements. This approach simplifies the numerical integration over the mesh elements. The figure below visualises the mesh consisting of 25 elements.

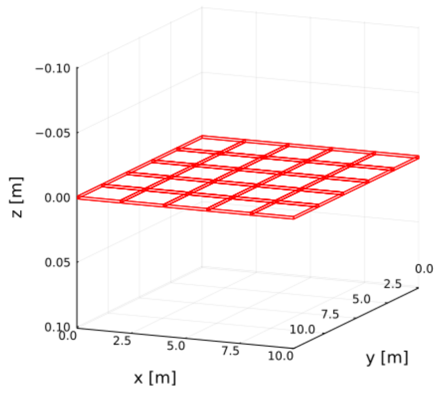


**Figure 9.1:** Visualisation of the mesh of the plate using AD, consisting of 25 elements.

## 9.2. Implementation

For now, only the approach that uses uniform basis functions and point matching is implemented. Remember that the weighting functions are the Dirac delta functions centered at the centres of mesh elements. Observe that when we want to calculate the submatrices on the diagonal of the interaction matrix, the following expressions are evaluated

$$\frac{\partial \tilde{\mathbf{B}}_{\text{red}}(\tilde{\mathbf{M}}, \mathbf{r_k})}{\partial \tilde{\mathbf{M}}^k} = \frac{\partial}{\partial \tilde{\mathbf{M}}^k}\left(\nabla \times \left(\frac{\mu_0}{4\pi}\iiint\limits_{\tau_k} \frac{\tilde{\mathbf{M}}^k \phi_k(\mathbf{r}') \times (\mathbf{r}-\mathbf{r}')}{||\mathbf{r_k}-\mathbf{r}'||^3}\,\mathrm{d}\mathbf{r}'\right)\right) \quad \text{for} \quad k = 1, \ldots, N, \tag{9.1}$$

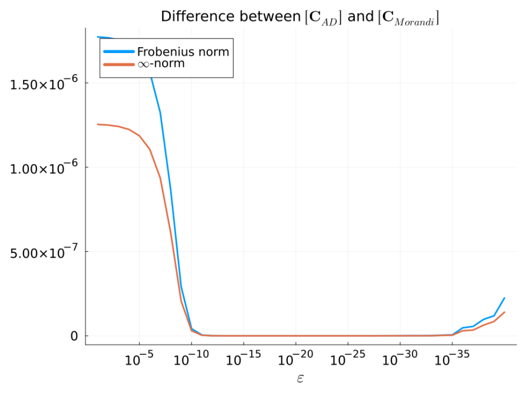
where $N$ is the number of mesh elements. Note that this is a singular integral. The function `hcubature()` can handle singular integrals, provided the singularity lies on the corner of the box-shaped integral domain. Otherwise, the function will return `NaN`. In this case, the singularity is located at the centre of the integration domain. Therefore, the domain is subdivided into eight smaller cubes, each with the singularity at one of its corners. The function `hcubature()` is now able to evaluate the integral on every subdomain and does not return `NaN`. However, test cases reveal that the results are not accurate. The resulting interaction matrix does not match the one computed using Morandi's method. Moreover, the solution for $\tilde{\mathbf{M}}$ is physically infeasible. A small value $\epsilon$ is added to the denominator of the integrand, which gives

$$\frac{\partial \tilde{\mathbf{B}}_{\text{red}}(\tilde{\mathbf{M}}, \mathbf{r_k})}{\partial \tilde{\mathbf{M}}^k} = \frac{\partial}{\partial \tilde{\mathbf{M}}^k}\left(\nabla \times \left(\frac{\mu_0}{4\pi}\iiint\limits_{\tau_k} \frac{\tilde{\mathbf{M}}^k \phi_k(\mathbf{r}') \times (\mathbf{r}-\mathbf{r}')}{||\mathbf{r_k}-\mathbf{r}'||^3+\epsilon}\,\mathrm{d}\mathbf{r}'\right)\right) \quad \text{for} \quad k = 1, \ldots, N. \tag{9.2}$$

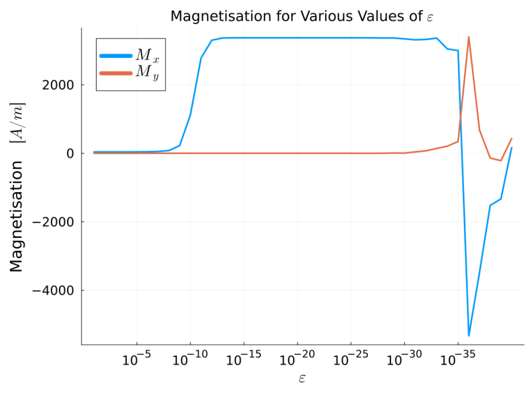
To determine the value of $\epsilon$, a geometry of a boxof $0.5 \times 0.5 \times 0.002\ \mathrm{m}$ is considered. Assuming the mesh consists of only one element, the calculations based on Chapter 3 and physical analysis indicate that the $3 \times 3$ interaction matrix is a diagonal matrix is a diagonal matrix with identical elements on the diagonal. From now on, the interaction matrix derived by Morandi'method will be denoted as $[\mathbf{C}_{Morandi}]$ and the interaction matrix derived by AD will be denoted as $[\mathbf{C}_{AD}]$. Remember that $[\mathbf{C}_{Morandi}]$ does not depend on a parameter $\epsilon$. The figure below shows the differences between the two matrices for various values of $\epsilon$ in both the Frobenius and the $\infty$-norm. The Frobenius norm and the $\infty$-norm of an arbitrary matrix $[\mathbf{A}] \in \mathbb{R}^{m \times n}$ equal [54, p. 127, p. 130]

$$||[\mathbf{A}]||_F = \left(\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2\right)^{1/2} \quad \text{and} \quad ||[\mathbf{A}]||_\infty = \max_{1 \le i \le m} \sum_{j=1}^{n} |a_{ij}|. \tag{9.3}$$

To assess the impact on the solution, the x- and y-components of the magnetisation for varying values $\epsilon$ are plotted when the object is exposed to a uniform external field of $50\ \mu\mathrm{T}$ in the positive x-direction.



**(a)** Difference between $[\mathbf{C}_{AD}]$ and $[\mathbf{C}_{Morandi}]$.



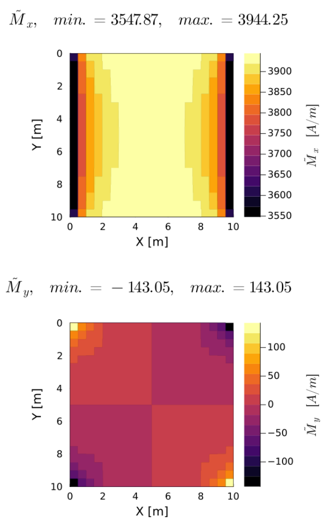
**(b)** The x- and y- component of the magnetisation.

**Figure 9.2:** The impact of $\epsilon$ on the interaction matrix and the obtained solution.

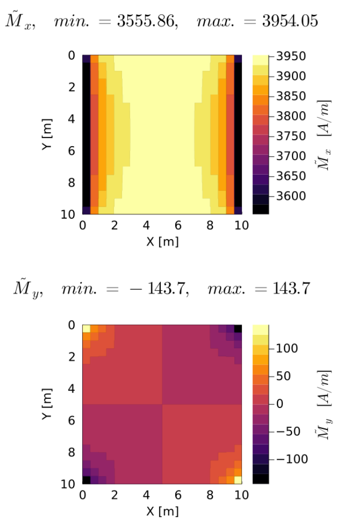
Results indicate that adding an $\epsilon$ to the integration kernel, within a specific range, is harmless. In this context, harmless means that the solution does not change significantly. Based on Figure 9.2 the decision is made to set $\epsilon = 10^{-13}$. Observe that a smaller value would not significantly change the solution for the magnetisation. However, when the value of $\epsilon$ is smaller than approximately $10^{-35}$, unexpected behaviour is observed in the interaction matrix $[\mathbf{A}_{AD}]$ and the solution.

## 9.3. Results

In this section, the results obtained using AD to find the interaction matrix are presented. The results are compared to the results using Morandi's method. The plate is uniformly meshed with 20 square elements in both the x- and y-directions, resulting in a total of 400 mesh elements. The uniform external magnetic flux density equals $50\ \mu\mathrm{T}$. The contour plots are made as explained in Section 3.7.



**(a)** Using AD.



**(b)** Using Morandi.

**Figure 9.3:** The assembly and solve time using uniform basis functions and point matching in Julia.

Since the interaction matrices using AD and using Morandi are approximately the same, the magnetisation within the plate is expected to be similar. Figure 9.3 confirms this. Both figures show the same pattern. Moreover, the minimum and maximum values of the x- and y-components of the magnetisation using AD are comparable to those obtained using Morandi.
The implementation using AD uses significantly more heap memory than Morandi's method due to the numerical integration, resulting in slower performance.

# 10 Conclusion and Recommendations

The primary goal of this research is to calculate magnetic signatures using the MoM in Julia, aiming for the highest accuracy and efficiency. Efficency is defined as faster computation, less memory usage, and simpler implementation. The research question is defined as follows

> *How can can TNO's current implementation of the assembly step in the MoM for modeling magnetic signatures be improved in terms of accuracy and efficiency using Julia?*

To answer this research question, the following sub-questions were investigated

1. What is the optimal implementation of a simplified version of the assembly step in TNO's current MoM in terms of accuracy, memory use, and computation speed using Julia?
2. What is the optimal choice of weighting functions using linear basis functions?
3. What is the effect on the accuracy of estimating magnetic signatures using linear basis functions?
4. How to use automatic differentiation such that the assembly step in the modeling process will be simplified?

The results from Chapter 6 indicate that using Julia instead of MATLAB can significantly improve the assembly time of the interaction matrix, regardless of whether parallel computing is used. Combined with the user-friendliness of Julia, this demonstrates its potential value for TNO in assembling the interaction matrix as efficient as possible. Note that for these systems, MATLAB's '$\backslash$'-operator is faster than Julia's '$\backslash$'-operator. Since MATLAB does not offer a straightforward method for benchmarking heap allocations during the simulation, it is not possible to determine whether memory usage is improved. Numerical experiment should be conducted to find out if using Julia improves the memory usage of the assembly step.
Chapter 4 shows that point matching is not a suitable approach when using linear basis functions. If the evaluation points are selected at the centre of the edges of the triangular surface, the resulting magnetisation is physically infeasible. When the shifted vertices of the mesh elements are used as evaluation points, the solution heavily depends on the parameter $\epsilon_{shift}$. It is necessary to investigate the optimal value of $\epsilon_{shift}$ for each mesh. Experiments show that determining the optimal value of the parameter is challenging. It can be concluded that the solution is highly sensitive to the chosen evaluation points, which is a significant drawback of point matching. The visualisations on the obtained magnetisation within a plate using a Galerkin method are promising. There is no dependency on a mesh-dependent parameter and, more important, when rounding down to a specific number of significant figures the linear system is symmetric. This would simplify the solving process. However, future research is needed to understand the effect of this rounding on the solution for the magnetisation. Also, the results in Chapter 6 show that the implementation uses more heap memory than the other described methods, which causes high computation times. It would be interesting to explore the possibility of developing a Galerkin method that uses less memory.
The different methods are verified in Chapter 7. The results show that the methods using linear basis functions converge for both point matching and the Galerkin method. However, the Galerkin method

shows faster converges than point matching. Based on the mesh convergence analysis, it is difficult to draw a final conclusion regarding the effect on accuracy when using linear basis functions. Future research is needed to provide more insights into the accuracy of the Galerkin method.
Chapter 9 demonstrates the potential of AD in the MoM. This research takes the first step by considering only uniform basis functions and point matching, showing that AD can simplify the implementation of the MoM by eliminating the need for analytical expressions for integrals. Note that the AD implementation also relies on a parameter $\epsilon$. While finding the optimal value of this parameter is not difficult, it is dependent on the mesh, which is a limitation. Also, the behaviour of the solution when $\epsilon$ converges to zero is unexpected and should be addressed in future work. Moreover, the AD implementation is suboptimal in terms of computational speed and memory usage. Even though using AD simplifies the implementation of the assembly step, its computational performance should be improved to be an alternative to the methods described in previous literature.

# References

[1] Eugene Lepelaars. personal communication.

[2] Alexei Gvishiani, Anatoly Soloviev, Roman Krasnoperov, and Renata Lukianova. Automated hardware and software system for monitoring the earth's magnetic environment. *Data Science Journal*, 15:18–18, 2016.

[3] RA Raveendra Varma. Design of degaussing system and demonstration of signature reduction on ship model through laboratory experiments. *Physics Procedia*, 54:174–179, 2014.

[4] John J Holmes. *Exploitation of a ship's magnetic field signatures*. Morgan & Claypool Publishers, 2006.

[5] Miroslaw Woloszyn and Jarosław Tarnawski. Magnetic signature reproduction of ferromagnetic ships at arbitrary geographical position, direction and depth using a multi-dipole model. *Scientific Reports*, 13(1):14601, 2023.

[6] A.R.P.J. Vijn. *Development of a Closed-loop degaussing system: Towards magnetic unobservable vessels*. Dissertation (tu delft), Delft University of Technology, 2021.

[7] Antonio Morandi, Massimo Fabbri, and Pier Luigi Ribani. A modified formulation of the volume integral equations method for 3-d magnetostatics. *IEEE transactions on magnetics*, 46(11):3848–3859, 2010.

[8] Walton C Gibson. *The method of moments in electromagnetics*. Chapman and Hall/CRC, 2021.

[9] Lei Xiao, Gang Mei, Ning Xi, and Francesco Piccialli. Julia language in computational mechanics: A new competitor. *Archives of Computational Methods in Engineering*, 29(3):1713–1726, 2022.

[10] John MD Coey. *Magnetism and magnetic materials*. Cambridge university press, 2010.

[11] David J Griffiths. Introduction to electrodynamics fourth edition. 2021.

[12] Richard Clinton Fernow. *Principles of magnetostatics*. Cambridge University Press, 2017.

[13] Zachary J Silberman, Thomas R Adams, Joshua A Faber, Zachariah B Etienne, and Ian Ruchlin. Numerical generation of vector potentials from specified magnetic fields. *Journal of Computational Physics*, 379:421–437, 2019.

[14] Anders Bondeson, Thomas Rylander, and Pär Ingelström. *Computational electromagnetics*. Springer, 2012.

[15] Jian-Ming Jin and Weng Cho Chew. Computational electromagnetics: The method of moments. *Electrical Engineering Handbook*, 1999.

[16] Sadri Hassani and Sadri Hassani. Dirac delta function. *Mathematical Methods: For Students of Physics and Related Fields*, pages 139–170, 2009.

[17] Roger F Harrington. The method of moments in electromagnetics. *Journal of Electromagnetic waves and Applications*, 1(3):181–200, 1987.

[18] Ross Howard and Steven Pekarek. Two-dimensional galerkin magnetostatic method of moments. *IEEE Transactions on Magnetics*, 53(12):1–6, 2017.

[19] A. C. Genz and A. A. Malik. Remarks on algorithm 006: An adaptive algorithm for numerical integration over an $n$-dimensional rectangular region. *Journal of Computational and Applied Mathematics*, 6:295–302, 1980. doi: 10.1016/0771-050x(80)90039-x.

[20] Steven G. Johnson. The HCubature.jl package for multi-dimensional adaptive integration in Julia. `https://github.com/JuliaMath/HCubature.jl`, 2017.

[21] Christophe Geuzaine and Jean-François Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International journal for numerical methods in engineering*, 79(11):1309–1331, 2009.

[22] Wu Wang and Naoshi Nishimura. Calculation of shape derivatives with periodic fast multipole method with application to shape optimization of metamaterials. *Progress In Electromagnetics Research*, 127:49–64, 2012.

[23] Elisabeth Roesch, Joe G Greener, Adam L MacLean, Huda Nassar, Christopher Rackauckas, Timothy E Holy, and Michael PH Stumpf. Julia for biologists. *Nature methods*, 20(5):655–664, 2023.

[24] Jonas Eschle, Tamás Gál, Mosè Giordano, Philippe Gras, Benedikt Hegner, Lukas Heinrich, Uwe Hernandez Acosta, Stefan Kluth, Jerry Ling, Pere Mato, et al. Potential of the julia programming language for high energy physics computing. *Computing and Software for Big Science*, 7(1):10, 2023.

[25] Soumen Pal, Manojit Bhattacharya, Snehasish Dash, Sang-Soo Lee, and Chiranjib Chakraborty. A next-generation dynamic programming languagejulia: its features and applications in biological science. *Journal of Advanced Research*, 2023.

[26] Jeff Bezanson, Jiahao Chen, Benjamin Chung, Stefan Karpinski, Viral B. Shah, Jan Vitek, and Lionel Zoubritzky. Julia: dynamism and performance reconciled by design. 2(OOPSLA), oct 2018. doi: 10.1145/3276490. URL `https://doi.org/10.1145/3276490`.

[27] Jeff Bezanson, Stefan Karpinski, Viral Shah, Alan Edelman, et al. Julia language documentation. *The Julia Manual*, pages 1–261, 2014.

[28] Francesc Verdugo and Santiago Badia. The software design of gridap: a finite element package based on the julia jit compiler. *Computer Physics Communications*, 276:108341, 2022.

[29] Jiahao Chen and Jarrett Revels. Robust benchmarking in noisy environments. *arXiv preprint arXiv:1608.04295*, 2016.

[30] Jiahao Chen and Jarrett Revels. Robust benchmarking in noisy environments. *arXiv e-prints*, art. arXiv:1608.04295, Aug 2016.

[31] Leo Ferres. Memory management in c: The heap and the stack. *Universidad de Concepcion*, 2010.

[32] Steve Klabnik and Carol Nichols. *The Rust programming language*. No Starch Press, 2023.

[33] Christopher Rackauckas. Parallel computing and scientific machine learning (sciml): Methods and applications, 2022. URL `https://github.com/SciML/SciMLBook`.

[34] Random numbers. URL `https://docs.julialang.org/en/v1/stdlib/Random/`. (accessed: 13-07-2024).

[35] Staticarrays. URL `https://github.com/JuliaArrays/StaticArrays.jl`. (accessed: 14-02-2024).

[36] Christopher Rackauckas and Qing Nie. Differentialequations.jl – a performant and feature-rich ecosystem for solving differential equations in julia. *The Journal of Open Research Software*, 5 (1), 2017. doi: 10.5334/jors.151. URL `https://app.dimensions.ai/details/publication/pub.1085583166andhttp://openresearchsoftware.metajnl.com/articles/10.5334/jors.151/galley/245/download/`.

[37] Artem Pelenitsyn, Julia Belyakova, Benjamin Chung, Ross Tate, and Jan Vitek. Type stability in julia: avoiding performance pathologies in jit compilation. *Proceedings of the ACM on Programming Languages*, 5(OOPSLA):1–26, 2021.

[38] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[39] Polyester. URL `https://github.com/JuliaSIMD/Polyester.jl`. (accessed: 30-05-2024).

[40] *The Julia Language*.

[41] A. Hermans and van der Kamp. Current state of knowledge electromagnetic fields, Sep 2022. URL `https://www.noordzeeloket.nl/publish/pages/204320/current-state-of-knowledge-electromagnetic-fields.pdf`.

[42] Nikita Chernetsov, Alexander Pakhomov, Alexander Davydov, Fedor Cellarius, and Henrik Mouritsen. No evidence for the use of magnetic declination for migratory navigation in two songbird species. *PLoS One*, 15(4):e0232136, 2020.

[43] Noaa magnetic field calculator. URL `https://www.ngdc.noaa.gov/geomag/calculators/mobileDeclination.shtml#WMM`. (accessed: 12-07-2024).

[44] Neal L Carothers. *Real analysis*. Cambridge University Press, 2000.

[45] Charalampos Bratsas, Kleanthis Koupidis, Josep Maria Salanova Grau, Konstantinos Giannakopoulos, Aristos Kaloudis, and Georgia Ayfantopoulou. A comparison of machine learning methods for the prediction of traffic speed in urban places. *Sustainability*, 12:142, 12 2019. doi: 10.3390/su12010142.

[46] Peter Glavič. Review of the international systems of quantities and units usage. *Standards*, 1(1): 2–16, 2021.

[47] Charles C Margossian. A review of automatic differentiation and its efficient implementation. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 9(4):e1305, 2019.

[48] Jarrett Revels, Miles Lubin, and Theodore Papamarkou. Forward-mode automatic differentiation in julia. *arXiv preprint arXiv:1607.07892*, 2016.

[49] Philipp HW Hoffmann. A hitchhiker's guide to automatic differentiation. *Numerical Algorithms*, 72 (3):775–811, 2016.

[50] Parth Nobel. auto_diff: an automatic differentiation package for python. In *2020 Spring Simulation Conference (SpringSim)*, pages 1–12. IEEE, 2020.

[51] Jerrold E Marsden and Anthony Tromba. *Vector calculus*. Macmillan, 2003.

[52] Louis B Rall. Perspectives on automatic differentiation: past, present, and future? In *Automatic Differentiation: Applications, Theory, and Implementations*, pages 1–14. Springer, 2006.

[53] Richard D Neidinger. Introduction to automatic differentiation and matlab object-oriented programming. *SIAM review*, 52(3):545–563, 2010.

[54] William Ford. *Numerical linear algebra with applications: Using MATLAB*. Academic Press, 2014.

[55] Eric W. Weisstein. Associated legendre differential equation. URL `https://mathworld.wolfram.com/AssociatedLegendreDifferentialEquation.html`.

[56] George B. Arfken and Hans J. Weber. *Mathematical Methods for Physicists*. Elsevier Academic Press, 2005.

[57] David A De Wolf. *Essentials of electromagnetics for engineering*. Cambridge University Press, 2001.

[58] Massimo Fabbri. Magnetic flux density and vector potential of uniform polyhedral sources. *IEEE Transactions on Magnetics*, 44(1):32–36, 2007.

[59] John David Jackson. *Classical electrodynamics*. John Wiley & Sons, 2021.

[60] Adriaan Van Oosterom and Jan Strackee. The solid angle of a plane triangle. *IEEE transactions on Biomedical Engineering*, (2):125–126, 1983.

# A

# Analytical Solution for a Spherical Shell

Consider a spherical shell made of material characterised by magnetic susceptibility $\chi_m$, with inner and outer radii $a$ and $b$, respectively. The permeability is defined as $\mu = \mu_0(1+\chi_m)$. The object is exposed to a uniform background magnetic field $\mathbf{B}_{\text{ext}} = B_{\text{ext}}\mathbf{u}_z$. The following three regions, each with homogeneous material properties, are distinguished:

1. The interior of the spherical shell.
2. The material of the spherical shell.
3. The exterior of the spherical shell.

Remember that in third region $\mathbf{B}_{\text{ext}} = \mu_0\mathbf{H}_{\text{ext}} = \mu_0 H_{\text{ext}}\mathbf{u}_z$ holds. Thus, $B_{\text{ext}} = \mu_0 H_{\text{ext}}$. A sketch of the situation is depicted below.

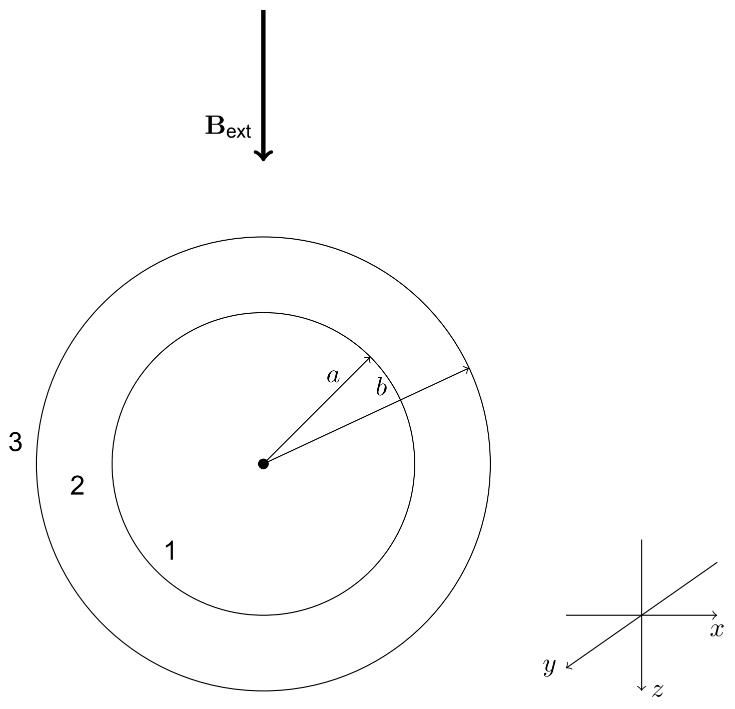


**Figure A.1:** Spherical shell in uniform background magnetic field.

First of all, magnetic scalar potentials for the three different regions are determined by solving Laplace's equation. With help of the boundary conditions, the coefficients for the scalar potentials in the different regions are computed. Finally, expressions for $\mathbf{M}$ and $\mathbf{B}_{\text{red}}$ are found.

## A.1. Laplace's Equation and Associated Legendre Functions

Remember that when no free currents are present in a region, the curl of the $\mathbf{H}$-field is zero. The following theorem states that there exists a scalar function $V$, such that $\mathbf{H} = -\nabla V$ [11, p. 54].

**Theorem 2** *The following conditions are equivalent:*

*(a)* $\nabla \times \mathbf{F} = 0$ *everywhere.*

*(b)* $\mathbf{F}$ *is the gradient of some scalar function:* $\mathbf{F} = -\nabla V$.

Using Maxwell's equations of magnetostatic problems, it follows that the scalar potential satisfies the Laplace equation [11, p. 84].

$$\mathbf{B} = \mu \mathbf{H}, \tag{A.1}$$

$$= -\nabla \mu V, \tag{A.2}$$

$$\nabla \cdot \mathbf{B} = \mu \nabla \cdot \nabla V, \tag{A.3}$$

$$\nabla \cdot \mathbf{B} = 0 \Rightarrow \quad \nabla^2 V = 0. \tag{A.4}$$

A spherical coordinate sytem is considered. The distance from the origin of the spherical shell to the position vector $\mathbf{r}$ is denoted by $r$. The polar angle $\theta$ is measured between the z-axis and $\mathbf{r}$. the azimuthal angle $\phi$ is the angle between the x-axis and the orthogonal projection of $\mathbf{r}$ onto the x-y-plane. Figure A.2 visualizes the angles and distance $r$.

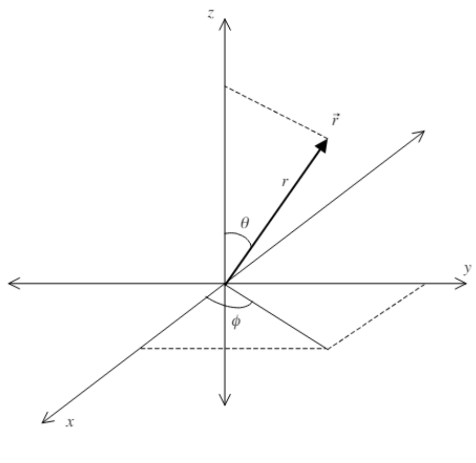


**Figure A.2:** Spherical coordinate system.

The relation between spherical coordinates and Cartesian coordinates is as follows

$$x = r\sin(\theta)\cos(\phi), \tag{A.5}$$

$$y = r\sin(\theta)\sin(\phi), \tag{A.6}$$

$$z = r\cos(\theta). \tag{A.7}$$

Laplace's equation yields

$$\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial V}{\partial r}\right) + \frac{1}{r^2\sin(\theta)}\frac{\partial}{\partial \theta}\left(\sin(\theta)\frac{\partial V}{\partial \theta}\right) + \frac{1}{r^2\sin^2(\theta)}\frac{\partial^2 V}{\partial \phi^2} = 0. \tag{A.8}$$

The assumption is made that the problem has azimuthal symmetry. This implies that $V$ is independent of $\phi$ and Equation A.8 reduces to

$$\frac{\partial}{\partial r}\left(r^2\frac{\partial V}{\partial r}\right)+\frac{1}{\sin(\theta)}\frac{\partial}{\partial\theta}\left(\sin(\theta)\frac{\partial V}{\partial\theta}\right)=0. \tag{A.9}$$

The assumption is made that the solution can be written as the sum over products of the following form

$$V(r,\theta)=R(r)\Theta(\theta). \tag{A.10}$$

Substituting this expression into Equation A.9 and dividing by $V$ gives

$$\frac{1}{R}\frac{d}{dr}\left(r^2\frac{dR}{dr}\right)+\frac{1}{\Theta\sin(\theta)}\frac{d}{d\theta}\left(\sin(\theta)\frac{d\Theta}{d\theta}\right)=0. \tag{A.11}$$

Note that the first term only depends on $r$, while the second term only depends on $\theta$. Since the equation holds for all $r$ and $\theta$, each term equals a constant

$$\frac{1}{R}\frac{d}{dr}\left(r^2\frac{dR}{dr}\right)=l(l+1),\qquad \frac{1}{\Theta\sin(\theta)}\frac{d}{d\theta}\left(\sin(\theta)\frac{d\Theta}{d\theta}\right)=-l(l+1), \tag{A.12}$$

where $l(l+1)$ is a way to write the separation constant. A solution of the form $R(r)=r^\alpha$ is substituted in the first equation. This yields

$$\begin{aligned} r^{-\alpha}\frac{d}{dr}\left(r^2\alpha r^{\alpha-1}\right)&=l(l+1), &\text{(A.13)}\\ r^{-\alpha}\alpha(\alpha+1)r^\alpha&=l(l+1), &\text{(A.14)}\\ \alpha(\alpha+1)&=l(l+1), &\text{(A.15)}\\ \alpha=-(l+1)&\vee\alpha=l. &\text{(A.16)}\end{aligned}$$

The general solution equals

$$R(r)=Ar^l+\frac{B}{r^{l+1}}, \tag{A.17}$$

for some constants $A$ and $B$. Now, a solution for $\Theta(\theta)$ is found. The second expression in Equation A.12 can be written as

$$\frac{d}{d\theta}\left(\sin(\theta)\frac{d\Theta}{d\theta}\right)+l(l+1)\Theta\sin(\theta)=0. \tag{A.18}$$

The following change of variables is introduced

$$\psi=\cos(\theta). \tag{A.19}$$

Recall that for any function $f(\psi)$

$$\begin{aligned}\frac{df}{d\theta}&=\frac{df}{d\psi}\frac{d\psi}{d\theta}, &\text{(A.20)}\\ &=-\sin(\theta)\frac{df}{d\psi}, &\text{(A.21)}\\ &=-\sqrt{1-\psi^2}\frac{df}{\psi}. &\text{(A.22)}\end{aligned}$$

Thus, Equation A.18 simplifies

$$\begin{aligned}\frac{d}{d\theta}\left(\sin(\theta)\frac{d\Theta}{d\theta}\right)+l(l+1)\Theta\sin(\theta)&=\frac{d}{d\theta}\left(-\sin(\theta)\sqrt{1-\psi^2}\frac{d\Theta}{d\psi}\right)+l(l+1)\Theta\sin(\theta), &\text{(A.23)}\\ &=-\sqrt{1-\psi^2}\frac{d}{d\theta}\left(\sin(\theta)\frac{d\Theta}{d\psi}\right)+l(l+1)\Theta\sin(\theta), &\text{(A.24)}\\ &=\frac{d}{d\psi}\left[(1-\psi^2)\frac{d\Theta}{d\psi}\right]+l(l+1)\Theta, &\text{(A.25)}\end{aligned}$$

$$\frac{d}{d\theta}\left(\sin(\theta)\frac{d\Theta}{d\theta}\right)+l(l+1)\Theta\sin(\theta)=0\Rightarrow\ \frac{d}{d\psi}\left[(1-\psi^2)\frac{d\Theta}{d\psi}\right]+l(l+1)\Theta=0. \tag{A.26}$$

This equation is known as the Legendre differential equation. Its solutions are polynomials in $\cos(\theta)$ and are known as the associated Legendre polynomials. The solution for $\Theta(\theta)$ is as follows [55]

$$\begin{aligned}\Theta(\theta) &= \hat{A}P_l(\psi) + \hat{B}Q_l(\psi), && \text{(A.27)}\\ &= \hat{A}P_l\left(\cos(\theta)\right) + \hat{B}Q_l\left(\cos(\theta)\right), && \text{(A.28)}\end{aligned}$$

for some constants $\hat{A}$ and $\hat{B}$. The functions $Q_l\left(\cos(\theta)\right)$ are unbounded when $\psi = 1$ or $\psi = -1$. Thus, the constant $\hat{B}$ equal zero. Since $l$ is an integer, the function $P_l\left(\cos(\theta)\right)$ reduces to the Legendre polynomial [55]. The Legendre polynomials are orthogonal to each other. They satisfy the orthonormality condition [56, p. 757]

$$\begin{aligned}\int_{-1}^{1} P_l(x)P_{l'}(x)\,\mathrm{d}x &= \int_{0}^{\pi} P_l(\cos(\theta))P_{l'}(\cos(\theta))\sin(\theta)\,\mathrm{d}\theta && \text{(A.29)}\\ &= \begin{cases} 0, & \text{if} \quad l \neq l', \\ \frac{2}{2l+1}, & \text{if} \quad l = l'. \end{cases} && \text{(A.30)}\end{aligned}$$

Also, the derivative of the Legendre polynomials can be written as

$$\frac{\mathrm{d}}{\mathrm{d}x}P_{l+1}(x) = \frac{2P_l(x)}{||P_l||^2} + \frac{2P_{l-2}(x)}{||P_{l-2}||^2} + \ldots, \tag{A.31}$$

where $||P_l|| = \sqrt{\frac{2}{2l+1}}$. The general solution for the potential is the linear combination of separable solution. As a result, a general solution equals

$$V(r,\theta) = \sum_{l=0}^{\infty}\left(A_l r^l + \frac{B_l}{r^{l+1}}\right)P_l\left(\cos(\theta)\right). \tag{A.32}$$

The potential for the problem must remain finite as $r \to \infty$. Therefore, in the third region, $A_l = 0$ for all $l$. Moreover, the potentials satisfy the superposition principle. This implies that the potential at any point is the sum of the potentials due to all the magnetic sources. Denote the potential of the uniform background field as $V_{\text{ext}}$. Remember that

$$\mathbf{B}_{\text{ext}} = B_{\text{ext}}\mathbf{u}_z \Rightarrow \mathbf{H}_{\text{ext}} = H_{\text{ext}}\mathbf{u}_z, \tag{A.33}$$

$$\mathbf{H}_{\text{ext}} = \begin{pmatrix} 0 \\ 0 \\ H_{\text{ext}} \end{pmatrix}, \tag{A.34}$$

$$-\nabla\left(-H_{\text{ext}}z\right) = \begin{pmatrix} 0 \\ 0 \\ H_{\text{ext}} \end{pmatrix} \Rightarrow V_{\text{ext}}(r,\theta) = -H_{\text{ext}}r\cos(\theta). \tag{A.35}$$

As a result, for $r > b$, the potential is of the form

$$V_3(r,\theta) = -H_{\text{ext}}r\cos(\theta) + \sum_{l=0}^{\infty}\frac{\alpha_l}{r^{l+1}}P_l\left(\cos(\theta)\right). \tag{A.36}$$

On the other hand, the potential must also remain finite as $r \to 0$. For $r < a$, the potential has the form

$$V_1(r,\theta) = \sum_{l=0}^{\infty}\delta_l r^l P_l\left(\cos(\theta)\right). \tag{A.37}$$

For $a < r < b$, the potential must be

$$V_2(r,\theta) = \sum_{l=0}^{\infty}\left(\beta_l r^l + \frac{\gamma_l}{r^{l+1}}\right)P_l\left(\cos(\theta)\right). \tag{A.38}$$

## A.2. Boundary Conditions

$H_\theta$ and $B_r$ should be continuous at $r = a$ and $r = b$. The boundary conditions can be written in terms of the potential $V$

$$\frac{\partial V_3}{\partial \theta}(b^+) = \frac{\partial V_2}{\partial \theta}(b^-), \quad \frac{\partial V_2}{\partial \theta}(a^+) = \frac{\partial V_1}{\partial \theta}(a^-), \tag{A.39}$$

$$\frac{\partial V_3}{\partial r}(b^+) = (1+\chi_m)\frac{\partial V_2}{\partial r}(b^-), \quad (1+\chi_m)\frac{\partial V_2}{\partial r}(a^+) = \frac{\partial V_1}{\partial r}(a^-). \tag{A.40}$$

With help of the boundary conditions, the coefficients for all $l$ can be found. The boundary condition at the surface $r = b$ for the continuity of $B_r$ results in

$$H_{\text{ext}}\cos(\theta) + \sum_{l=0}^{\infty} -(l+1)\frac{\alpha_l}{b^{l+2}}P_l\left(\cos(\theta)\right) = (1+\chi_m)\sum_{l=0}^{\infty}\left(l\beta_l b^{l-1} - (l+1)\frac{\gamma_l}{b^{l+2}}\right)P_l(\cos(\theta)), \tag{A.41}$$

$$H_{\text{ext}}P_1(\cos(\theta)) + \sum_{l=0}^{\infty} -(l+1)\frac{\alpha_l}{b^{l+2}}P_l\left(\cos(\theta)\right) = (1+\chi_m)\sum_{l=0}^{\infty}\left(l\beta_l b^{l-1} - (l+1)\frac{\gamma_l}{b^{l+2}}\right)P_l(\cos(\theta)), \tag{A.42}$$

$$\sum_{l=0}^{\infty}\left(-(l+1)\frac{\alpha_l}{b^{l+2}} - (1+\chi_m)l\beta_l b^{l-1} + (1+\chi_m)(l+1)\frac{\gamma_l}{b^{l+2}}\right)P_l(\cos(\theta)) = -H_{\text{ext}}P_1(\cos(\theta)). \tag{A.43}$$

Both sides are multiplied by $P_{l'}(\cos(\theta))\sin(\theta)$ and integrated from $0$ to $\pi$. Here two cases are distinguished, $l' = 1$ and $l' \neq 1$. This results in

$$\begin{cases} -2\frac{\alpha_1}{b^3} - (1+\chi_m)\beta_1 + 2(1+\chi_m)\frac{\gamma_1}{b^3} = -H_{\text{ext}}, & \text{if} \quad l' = 1 \\ -(l'+1)\frac{\alpha_{l'}}{b^{l'+2}} - (1+\chi_m)l'b^{l'-1}\beta_{l'} + (l'+1)(1+\chi_m)\frac{\gamma_{l'}}{b^{l'+2}} = 0, & \text{if} \quad l' \neq 1. \end{cases} \tag{A.44}$$

The continuity of $H_\theta$ is satisfied when

$$H_{\text{ext}}b\sin(\theta) + \sum_{l=0}^{\infty}\frac{\alpha_l}{b^{l+1}}\frac{\partial P_l(\cos(\theta))}{\partial\theta} = \sum_{l=0}^{\infty}\left(\beta_l b^l + \frac{\gamma_l}{b^{l+1}}\right)\frac{\partial P_l(\cos(\theta))}{\partial\theta}, \tag{A.45}$$

$$-H_{\text{ext}}b\frac{\partial P_1(\cos(\theta))}{\partial\theta} + \sum_{l=0}^{\infty}\frac{\alpha_l}{b^{l+1}}\frac{\partial P_l(\cos(\theta))}{\partial\theta} = \sum_{l=0}^{\infty}\left(\beta_l b^l + \frac{\gamma_l}{b^{l+1}}\right)\frac{\partial P_l(\cos(\theta))}{\partial\theta}, \tag{A.46}$$

$$\sum_{l=0}^{\infty}\left(\frac{\alpha_l}{b^{l+1}} + \beta_l b^l + \frac{\gamma_l}{b^{l+1}}\right)\frac{\partial P_l(\cos(\theta))}{\partial\theta} = H_{\text{ext}}b\frac{\partial P_1(\cos(\theta))}{\partial\theta}, \tag{A.47}$$

$$\sum_{l=0}^{\infty}\left(\frac{\alpha_l}{b^{l+1}} + \beta_l b^l + \frac{\gamma_l}{b^{l+1}}\right)\left(\frac{2P_{l-1}(\cos(\theta))}{||P_{l-1}||^2} + \frac{2P_{l-3}(\cos(\theta))}{||P_{l-3}||^2} + \dots\right) = H_{\text{ext}}b\frac{2P_0(\cos(\theta))}{||P_0||^2}. \tag{A.48}$$

Multiplying by $P_{l'}(\cos(\theta))\sin(\theta)$ and integrating from $0$ to $\pi$ gives

$$\begin{cases} \frac{\alpha_1}{b^2} + \beta_1 b + \frac{\gamma_1}{b^2} = H_{\text{ext}}b, & \text{if} \quad l' = 1, \\ \frac{\alpha_{l'}}{b^{l'+1}} + \beta_{l'}b^{l'} + \frac{\gamma_{l'}}{b^{l'+1}} = 0, & \text{if} \quad l' \neq 1 \end{cases} \tag{A.49}$$

These steps can be repeated for the boundary conditions for the surface $r = a$. It can be concluded that the coefficients $l' \neq 1$ satisfy the following equations

$$-(l'+1)\frac{\alpha_{l'}}{b^{l'+2}} - (1+\chi_m)l'b^{l'-1}\beta_{l'} + (l'+1)(1+\chi_m)\frac{\gamma_{l'}}{b^{l'+2}} = 0, \tag{A.50}$$

$$\frac{\alpha_{l'}}{b^{l'+1}} + \beta_{l'}b^{l'} + \frac{\gamma_{l'}}{b^{l'+1}} = 0, \tag{A.51}$$

$$a^{l'}\beta_{l'} + \frac{\gamma_{l'}}{a^{l'+1}} - a^{l'}\delta_{l'} = 0, \tag{A.52}$$

$$(1+\chi_m)l'a^{l'-1}\beta_{l'} - (1+\chi_m)(l'+1)\frac{\gamma_{l'}}{a^{l'+2}} - l'a^{l'-1}\delta_{l'} = 0. \tag{A.53}$$

By solving this system, it can be concluded that the four coefficients equal zero, regardless of the value of $l$. The coefficients $l = 1$ satisfy the following equations

$$\alpha_1 - b^3\beta_1 - \gamma_1 = b^3 H_{\text{ext}}, \tag{A.54}$$

$$2\alpha_1 + \mu' b^3\beta_1 - 2\mu'\gamma_1 = -b^3 H_{\text{ext}}, \tag{A.55}$$

$$a^3\beta_1 + \gamma_1 - a^3\delta_1 = 0, \tag{A.56}$$

$$\mu' a^3\beta_1 - 2\mu'\gamma_1 - a^3\delta_1 = 0, \tag{A.57}$$

where $\mu' = 1 + \chi_m$. The system is solved and the coefficients equal

$$\alpha_1 = \left(\frac{(2\mu'+1)(\mu'-1)}{(2\mu'+1)(\mu'+2) - 2\frac{a^3}{b^3}(\mu'-1)^2}\right)(b^3 - a^3)H_{\text{ext}}, \tag{A.58}$$

$$\beta_1 = -\left(\frac{3(2\mu'+1)}{(2\mu'+1)(\mu'+2) - 2\frac{a^3}{b^3}(\mu'-1)^2}\right)H_{\text{ext}}, \tag{A.59}$$

$$\gamma_1 = -\left(\frac{3a^3(\mu'-1)}{(2\mu'+1)(\mu'+2) - 2\frac{a^3}{b^3}(\mu'-1)^2}\right)H_{\text{ext}}, \tag{A.60}$$

$$\delta_1 = -\left(\frac{9\mu'}{(2\mu'+1)(\mu'+2) - 2\frac{a^3}{b^3}(\mu'-1)^2}\right)H_{\text{ext}}. \tag{A.61}$$

## A.3. Find $\mathbf{M}$ and $\mathbf{B}_{\text{red}}$

The analytical solutions for $\mathbf{M}$ and $\mathbf{B}_{\text{red}}$ are expressed in both spherical and Cartesian coordinates.

### A.3.1. $\mathbf{M}$ and $\mathbf{B}_{\text{red}}$ in Spherical Coordinates

To find the magnetisation inside the material, the magnetic potential for region 2 is used. Substituting the coefficients in Equation A.38, gives the following

$$V_2(r,\theta) = \beta_1 r\cos(\theta) + \frac{\gamma_1}{r^2}\cos(\theta). \tag{A.62}$$

For $a < r < b$, $\mathbf{H} = -\nabla V_2(r,\theta)$. Thus,

$$\mathbf{H}(r,\theta) = \begin{pmatrix} H_r(r,\theta) \\ H_\theta(r,\theta) \\ H_\phi(r,\theta) \end{pmatrix}, \tag{A.63}$$

$$= \begin{pmatrix} -\frac{\partial V_2}{\partial r} \\ -\frac{1}{r}\frac{\partial V_2}{\partial \theta} \\ -\frac{1}{r\sin(\theta)}\frac{\partial V_2}{\partial \phi} \end{pmatrix}, \tag{A.64}$$

$$= \begin{pmatrix} \cos(\theta)(-\beta_1 + 2\frac{\gamma_1}{r^3}) \\ \sin(\theta)(\beta_1 + \frac{\gamma_1}{r^3}) \\ 0 \end{pmatrix}. \tag{A.65}$$

Since $\mathbf{M} = \chi_m \mathbf{H}$, $\mathbf{M}$ is as follows

$$\mathbf{M}(r,\theta) = \begin{pmatrix} M_r(r,\theta) \\ M_\theta(r,\theta) \\ 0 \end{pmatrix} \tag{A.66}$$

$$= \chi_m \begin{pmatrix} \cos(\theta)(-\beta_1 + \frac{2\gamma_1}{r^3}) \\ \sin(\theta)(\beta_1 + \frac{\gamma_1}{r^3}) \\ 0 \end{pmatrix}, \tag{A.67}$$

$$\text{where} \quad \beta_1 = -\left( \frac{3(2\mu' + 1)}{(2\mu'+1)(\mu'+2) - 2\frac{a^3}{b^3}(\mu'-1)^2} \right) H_{\text{ext}},$$

$$\gamma_1 = -\left( \frac{3a^3(\mu' - 1)}{(2\mu'+1)(\mu'+2) - 2\frac{a^3}{b^3}(\mu'-1)^2} \right) H_{\text{ext}},$$

$$\text{and} \quad \mu' = \chi_m + 1. \tag{A.68}$$

Using the magnetic scalar potential of the first region, the auxiliary magnetic field $\mathbf{H}$ outside the shell is found. Using the fact that $\mathbf{B} = \mu_0 \mathbf{H}$ in air, results in the following expression for the reduced magnetic flux density in region 3 due to the magnetisation of the spherical shell

$$\mathbf{B}_{\text{red}}(r,\theta) = \begin{pmatrix} B_{\text{red}_r}(r,\theta) \\ B_{\text{red}_\theta}(r,\theta) \\ B_{\text{red}_\phi}(r,\theta) \end{pmatrix}, \tag{A.69}$$

$$= \mu_0 \begin{pmatrix} \cos(\theta)(H_{\text{ext}} + 2\frac{\alpha_1}{r^3}) \\ \sin(\theta)(-H_{\text{ext}} + \frac{\gamma_1}{r^3}) \\ 0 \end{pmatrix}, \tag{A.70}$$

$$\text{where} \quad \alpha_1 = \left( \frac{(2\mu'+1)(\mu'-1)}{(2\mu'+1)(\mu'+2) - 2\frac{a^3}{b^3}(\mu'-1)^2} \right) (b^3 - a^3) H_{\text{ext}} \tag{A.71}$$

$$\text{and} \quad \mu' = \chi_m + 1. \tag{A.72}$$

### A.3.2. $\mathbf{M}$ and $\mathbf{B}_{\text{red}}$ in Cartesian Coordinates

To obtain the magnetic field in Cartesian coordinates, the following relation is used [57, p. 30]

$$H_x = \sin(\theta)\cos(\phi)H_r + \cos(\theta)\cos(\phi)H_\theta - \sin(\phi)H_\phi, \tag{A.73}$$

$$H_y = \sin(\theta)\sin(\phi)H_r + \cos(\theta)\sin(\phi)H_\theta + \cos(\phi)H_\phi, \tag{A.74}$$

$$H_z = \cos(\theta)H_r - \sin(\theta)H_\theta. \tag{A.75}$$

Substituting $H_r$, $H_\theta$, and $H_\phi$ gives

$$\begin{pmatrix} H_x \\ H_y \\ H_z \end{pmatrix} = \begin{pmatrix} \sin(\theta)\cos(\phi)\cos(\theta)\left(-\beta_1 + 2\frac{\gamma_1}{r^3}\right) + \cos(\theta)\cos(\phi)\sin(\theta)\left(\beta_1 + \frac{\gamma_1}{r^3}\right) \\ \sin(\theta)\sin(\phi)\cos(\theta)\left(-\beta_1 + 2\frac{\gamma_1}{r^3}\right) + \cos(\theta)\sin(\phi)\sin(\theta)\left(\beta_1 + \frac{\gamma_1}{r^3}\right) \\ \cos^2(\theta)\left(-\beta_1 + 2\frac{\gamma_1}{r^3}\right) - \sin^2(\theta)\left(\beta_1 + \frac{\gamma_1}{r^3}\right) \end{pmatrix}, \tag{A.76}$$

$$= \begin{pmatrix} 3\frac{\gamma_1}{r^3}\sin(\theta)\cos(\phi)\cos(\theta) \\ 3\frac{\gamma_1}{r^3}\sin(\theta)\sin(\phi)\cos(\theta) \\ -\beta_1 + \cos^2(\theta)2\frac{\gamma_1}{r^3} - \sin^2(\theta)\frac{\gamma_1}{r^3} \end{pmatrix}, \tag{A.77}$$

$$= \begin{pmatrix} 3\frac{\gamma_1}{r^5}r\sin(\theta)\cos(\phi)r\cos(\theta) \\ 3\frac{\gamma_1}{r^5}r\sin(\theta)\sin(\phi)r\cos(\theta) \\ -\beta_1 + 2r^2\cos^2(\theta)\frac{\gamma_1}{r^5} - r^2\sin^2(\theta)\frac{\gamma_1}{r^5}, \end{pmatrix}, \tag{A.78}$$

$$= \begin{pmatrix} 3\frac{\gamma_1}{r^5}r\sin(\theta)\cos(\phi)r\cos(\theta) \\ 3\frac{\gamma_1}{r^5}r\sin(\theta)\sin(\phi)r\cos(\theta) \\ -\beta_1 + 2r^2\cos^2(\theta)\frac{\gamma_1}{r^5} + r^2\sin^2(\theta)\left(-\sin^2(\phi) - \cos^2(\phi)\right)\frac{\gamma_1}{r^5}, \end{pmatrix}, \tag{A.79}$$

$$= \begin{pmatrix} \frac{3\gamma_1 xz}{(x^2+y^2+z^2)^{5/2}} \\ \frac{3\gamma_1 yz}{(x^2+y^2+z^2)^{5/2}} \\ -\beta_1 + \frac{\gamma_1(2z^2-y^2-x^2)}{(x^2+y^2+z^2)^{5/2}} \end{pmatrix}, \tag{A.80}$$

$$\tag{A.81}$$

Therefore, $\mathbf{M}$ is as follows

$$\mathbf{M}(x,y,z) = \chi_m \begin{pmatrix} \frac{3\gamma_1 xz}{(x^2+y^2+z^2)^{5/2}} \\ \frac{3\gamma_1 yz}{(x^2+y^2+z^2)^{5/2}} \\ -\beta_1 + \frac{\gamma_1(2z^2-y^2-x^2)}{(x^2+y^2+z^2)^{5/2}} \end{pmatrix}, \tag{A.82}$$

$$\text{where} \quad \beta_1 = -\left(\frac{3(2\mu'+1)}{(2\mu'+1)(\mu'+2) - 2\frac{a^3}{b^3}(\mu'-1)^2}\right)H_{\text{ext}},$$

$$\gamma_1 = -\left(\frac{3a^3(\mu'-1)}{(2\mu'+1)(\mu'+2) - 2\frac{a^3}{b^3}(\mu'-1)^2}\right)H_{\text{ext}}$$

$$\text{and} \quad \mu' = \chi_m + 1.$$

Repeating these steps for the magnetic scalar potential of the third region, results in the following expression for the reduced magnetic flux density in region 3 due to the magnetisation of the spherical shell

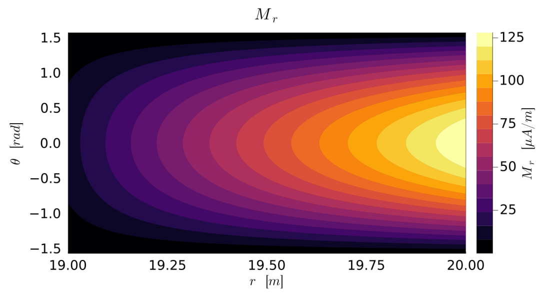
$$\mathbf{B}_{\text{red}}(x,y,z) = \mu_0\alpha_1 \begin{pmatrix} \frac{3xz}{(x^2+y^2+z^2)^{5/2}} \\ \frac{3yz}{(x^2+y^2+z^2)^{5/2}} \\ \frac{(2z^2-x^2-y^2)}{(x^2+y^2+z^2)^{5/2}} \end{pmatrix}, \tag{A.83}$$

$$\text{where} \quad \alpha_1 = \left(\frac{(2\mu'+1)(\mu'-1)}{(2\mu'+1)(\mu'+2) - 2\frac{a^3}{b^3}(\mu'-1)^2}\right)(b^3 - a^3)H_{\text{ext}}$$

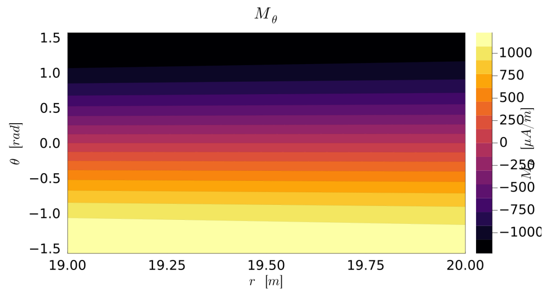
$$\text{and} \quad \mu' = \chi_m + 1.$$

## A.4. Visualisation of Magnetisation and Magnetic Vector Potential

To gain better insight into the analytical solution for the magnetisation inside a spherical shell, consider the following example. Suppose a spherical shell, with inner radius $19\,\mathrm{m}$ and outer radius $20\,\mathrm{m}$, is placed in an external uniform field $\mathbf{B}_{\text{ext}} = 50\mathbf{u}_z$ μ$\mathrm{T}$. The figure below shows both the $r$- and $\theta$-component of the magnetisation within the spherical shell.



**(a)** $M_r$.



**(b)** $M_\theta$

**Figure A.3:** The $r$- and $\theta$-component of the magnetisation within the spherical shell.

Moreover, an analytical expression for the magnetic vector potential is derived by [1]. It turns out that $\mathbf{A}$ only has a $\phi$-component. Visualising $A_\phi$ for within the spherical shell gives the following result.

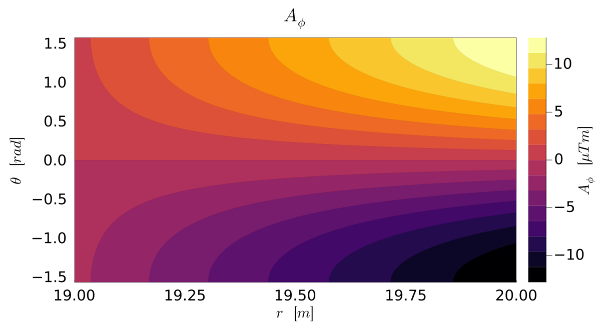


**Figure A.4:** The $\phi$-component of the magnetic vector potential within the spherical shell.

# B

# Details of the Calculations of Coefficient Matrix $\mathbf{C}$

The magnetic vector potential produced by a uniform magnetisation, denoted by $\tilde{\mathbf{M}}^k$, of element $\tau_k$ can be calculated as follows

$$\mathbf{A}(\mathbf{r}) = \frac{\mu_0}{4\pi} \iiint\limits_{\tau_k} \frac{\tilde{\mathbf{M}}^k \times (\mathbf{r}-\mathbf{r}')}{||\mathbf{r}-\mathbf{r}'||^3}\,\mathrm{d}\mathbf{r}', \tag{B.1}$$

$$\tilde{\mathbf{M}}^k \text{ is uniform} \Rightarrow = \frac{\mu_0}{4\pi}\tilde{\mathbf{M}}^k \times \iiint\limits_{\tau_k} \frac{(\mathbf{r}-\mathbf{r}')}{||\mathbf{r}-\mathbf{r}'||^3}\,\mathrm{d}\mathbf{r}'. \tag{B.2}$$

Introduce the vector identity [58]

$$\frac{\mathbf{r}-\mathbf{r}'}{||\mathbf{r}-\mathbf{r}'||^3} = \nabla' \frac{1}{||\mathbf{r}'-\mathbf{r}||} \tag{B.3}$$

and substitute in Equation B.2. This gives the following equation

$$\mathbf{A}(\mathbf{r}) = \frac{\mu_0}{4\pi}\tilde{\mathbf{M}}^k \times \iiint\limits_{\tau_k} \nabla' \frac{1}{||\mathbf{r}'-\mathbf{r}||}\,\mathrm{d}\mathbf{r}'. \tag{B.4}$$

By applying Gauss' theorem, the integral over the volume can be written as [11, p. 31]

$$\mathbf{A}(\mathbf{r}) = \frac{\mu_0}{4\pi}\tilde{\mathbf{M}}^k \times \iint\limits_{\partial\tau_k} \frac{1}{||\mathbf{r}-\mathbf{r}'||}\mathbf{n}(\mathbf{r}')\,\mathrm{d}\mathbf{r}', \tag{B.5}$$

$$= \frac{\mu_0}{4\pi}\tilde{\mathbf{M}}^k \times \sum_{S_f \in \partial\tau_k} \mathbf{n}_f \iint\limits_{S_f} \frac{1}{||\mathbf{r}-\mathbf{r}'||}\,\mathrm{d}\mathbf{r}', \tag{B.6}$$

$$= \frac{\mu_0}{4\pi}\tilde{\mathbf{M}}^k \times \sum_{S_f \in \partial\tau_k} \mathbf{n}_f W_f(\mathbf{r}), \tag{B.7}$$

where $S_f$ is any face of the boundary of the polyhedron, $\mathbf{n}_f$ is the outgoing normal unit vector of face $S_f$, and the function $W_f(\mathbf{r})$ is defined as

$$W_f(\mathbf{r}) = \iint\limits_{S_f} \frac{1}{||\mathbf{r}-\mathbf{r}'||}\,\mathrm{d}\mathbf{r}'. \tag{B.8}$$

As described in Chapter 2, the reduced flux density, denoted by $\tilde{\mathbf{B}}^{\mathbf{k}}_{\mathbf{red}}$ can be found by taking the curl of the magnetic vector potential. Thus, the reduced flux density equals

$$\tilde{\mathbf{B}}^{\mathbf{k}}_{\mathbf{red}}(\mathbf{r}) = \nabla \times \mathbf{A}(\mathbf{r}), \tag{B.9}$$

$$= \nabla \times \left( \frac{\mu_0}{4\pi} \sum_{S_f \in \partial\tau_k} \tilde{\mathbf{M}}^k \times \mathbf{n}_f W_f(\mathbf{r}) \right), \tag{B.10}$$

$$= \frac{\mu_0}{4\pi} \sum_{S_f \in \partial\tau_k} \nabla \times \left( \tilde{\mathbf{M}}^k \times \mathbf{n}_f W_f(\mathbf{r}) \right). \tag{B.11}$$

The following vector identity is introduced [59, p. 1]

$$\nabla \times (\mathbf{a} \times \mathbf{b}) = \mathbf{a}(\nabla \cdot \mathbf{b}) - \mathbf{b}(\nabla \cdot \mathbf{a}) + (\mathbf{b} \cdot \nabla)\mathbf{a} - (\mathbf{a} \cdot \nabla)\mathbf{b}. \tag{B.12}$$

With help of this identity, the following is obtained

$$\nabla \times \left( \tilde{\mathbf{M}}^k \times \mathbf{n}_f W_f(\mathbf{r}) \right) = \tilde{\mathbf{M}}^k (\nabla \cdot \mathbf{n}_f W_f(\mathbf{r})) - \mathbf{n}_f W_f(\mathbf{r})(\nabla \cdot \tilde{\mathbf{M}}^k) + (\mathbf{n}_f W_f(\mathbf{r}) \cdot \nabla)\tilde{\mathbf{M}}^k - (\tilde{\mathbf{M}}^k \cdot \nabla)\mathbf{n}_f W_f(\mathbf{r}), \tag{B.13}$$

$$\tilde{\mathbf{M}}^k \text{ is uniform} \Rightarrow = \tilde{\mathbf{M}}^k (\nabla \cdot \mathbf{n}_f W_f(\mathbf{r})) - (\tilde{\mathbf{M}}^k \cdot \nabla)\mathbf{n}_f W_f(\mathbf{r}), \tag{B.14}$$

$$= \tilde{\mathbf{M}}^k (W_f(\mathbf{r})(\nabla \cdot \mathbf{n}_f) + (\nabla W_f(\mathbf{r})) \cdot \mathbf{n}_f) - \tilde{\mathbf{M}}^{\mathbf{k}} \cdot (\nabla \mathbf{n}_f W_f(\mathbf{r})), \tag{B.15}$$

$$\nabla \cdot \mathbf{n}_f = 0 \Rightarrow = \tilde{\mathbf{M}}^k (\nabla W_f(\mathbf{r}) \cdot \mathbf{n}_f) - \mathbf{n}_f (\nabla W_f(\mathbf{r}) \cdot \tilde{\mathbf{M}}^k), \quad = -(\tilde{\mathbf{M}}^k \times \tag{B.16}$$

Substituting this into Equation B.11 results in

$$\tilde{\mathbf{B}}^{\mathbf{k}}_{\mathbf{red}}(\mathbf{r}) = -\frac{\mu_0}{4\pi} \sum_{S_f \in \tau_k} \left( \tilde{\mathbf{M}}^k \times \mathbf{n}_f \right) \times \nabla W_f(\mathbf{r}). \tag{B.17}$$

## B.1. Evaluate $\nabla W_f(\mathbf{r})$

To rewrite $\nabla W_f(\mathbf{r})$, the following vector identity is used [58]

$$\nabla \frac{1}{||\mathbf{r} - \mathbf{r}'||} = \mathbf{n}_f \times \left( \mathbf{n}_f \times \nabla' \frac{1}{||\mathbf{r}' - \mathbf{r}||} \right) + \mathbf{n}_f \frac{(\mathbf{r}' - \mathbf{r}) \cdot \mathbf{n}_f}{||\mathbf{r}' - \mathbf{r}||^3}. \tag{B.18}$$

First, this vector identity will be proven. For the vector triple product, the following relationship holds

$$\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = (\mathbf{a} \cdot \mathbf{c})\mathbf{b} - (\mathbf{a} \cdot \mathbf{b})\mathbf{c}. \tag{B.19}$$

Using this relation, the first term on the right-hand side can be written as

$$\mathbf{n}_f \times \left( \mathbf{n}_f \times \nabla' \frac{1}{||\mathbf{r}' - \mathbf{r}||} \right) = \mathbf{n}_f (\mathbf{n}_f \cdot \nabla' \frac{1}{||\mathbf{r}' - \mathbf{r}||}) - \nabla' \frac{1}{||\mathbf{r}' - \mathbf{r}||} (\mathbf{n}_f \cdot \mathbf{n}_f), \tag{B.20}$$

$$\mathbf{n}_f \cdot \mathbf{n}_f = 1 \Rightarrow = \mathbf{n}_f (\mathbf{n}_f \cdot \nabla' \frac{1}{||\mathbf{r}' - \mathbf{r}||}) - \nabla' \frac{1}{||\mathbf{r}' - \mathbf{r}||}. \tag{B.21}$$

Remember that

$$\frac{\mathbf{r}' - \mathbf{r}}{||\mathbf{r}' - \mathbf{r}||^3} = -\nabla' \frac{1}{||\mathbf{r}' - \mathbf{r}||}. \tag{B.22}$$

Using this vector identity and substituting Equation B.21 into Equation B.18, the following is obtained

$$\mathbf{n}_f \times \left( \mathbf{n}_f \times \nabla' \frac{1}{||\mathbf{r}' - \mathbf{r}||} \right) + \mathbf{n}_f \frac{(\mathbf{r}' - \mathbf{r}) \cdot \mathbf{n}_f}{||\mathbf{r}' - \mathbf{r}||^3} = \mathbf{n}_f (\mathbf{n}_f \cdot \nabla' \frac{1}{||\mathbf{r}' - \mathbf{r}||}) - \nabla' \frac{1}{||\mathbf{r}' - \mathbf{r}||} - \mathbf{n}_f (\nabla' \frac{1}{||\mathbf{r}' - \mathbf{r}||} \cdot \mathbf{n}_f), \tag{B.23}$$

$$= -\nabla' \frac{1}{||\mathbf{r}' - \mathbf{r}||}, \tag{B.24}$$

$$= \nabla \frac{1}{||\mathbf{r}' - \mathbf{r}||}. \tag{B.25}$$

This proves the identity in Equation B.18. Using this vector identity, $\nabla W_f(\mathbf{r})$ can be simplified as follows

$$\nabla W_f(\mathbf{r}) = \nabla \iint\limits_{S_f} \frac{1}{||\mathbf{r}-\mathbf{r}'||}\,\mathrm{d}\mathbf{r}', \tag{B.26}$$

$$= \iint\limits_{S_f} \nabla \frac{1}{||\mathbf{r}-\mathbf{r}'||}\,\mathrm{d}\mathbf{r}', \tag{B.27}$$

$$= \iint\limits_{S_f} \mathbf{n}_f \times \left(\mathbf{n}_f \times \nabla' \frac{1}{||\mathbf{r}'-\mathbf{r}||}\right) + \mathbf{n}_f \frac{(\mathbf{r}'-\mathbf{r})\cdot \mathbf{n}_f}{||\mathbf{r}'-\mathbf{r}||^3}, \tag{B.28}$$

$$= \mathbf{n}_f \times \iint\limits_{S_f} \mathbf{n}_f \times \nabla' \frac{1}{||\mathbf{r}'-\mathbf{r}||}\,\mathrm{d}\mathbf{r}' + \mathbf{n}_f \iint\limits_{S_f} \frac{(\mathbf{r}'-\mathbf{r})\cdot \mathbf{n}_f}{||\mathbf{r}'-\mathbf{r}||^3}\,\mathrm{d}\mathbf{r}'. \tag{B.29}$$

Suppose the face $S_f$ is triangular with vertices $\mathbf{r_1}$, $\mathbf{r_2}$, $\mathbf{r_3}$. The solid angle subtended to face $S_f$ as seen from point $\mathbf{r}$, denoted as $\Omega(\mathbf{r}, \mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3})$, is recognised in the second expression [60]. Equation B.29 results in

$$\nabla W_f(\mathbf{r}) = \mathbf{n}_f \times \iint\limits_{S_f} \mathbf{n}_f \times \nabla' \frac{1}{||\mathbf{r}'-\mathbf{r}||}\,\mathrm{d}\mathbf{r}' + \mathbf{n}_f \Omega(\mathbf{r}, \mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}), \tag{B.30}$$

with

$$\Omega(\mathbf{r}, \mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}) = 2\,\mathrm{arctan2}\, \frac{(\mathbf{r_1}-\mathbf{r})\cdot(\mathbf{r_2}-\mathbf{r})\times(\mathbf{r_3}-\mathbf{r})}{D(\mathbf{r}, \mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3})}, \tag{B.31}$$

$$\text{where} \quad D(\mathbf{r}, \mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}) = ||\mathbf{r_1}-\mathbf{r}||\,||\mathbf{r_2}-\mathbf{r}||\,||\mathbf{r_3}-\mathbf{r}|| + ||\mathbf{r_1}-\mathbf{r}||(\mathbf{r_2}-\mathbf{r})\cdot(\mathbf{r_3}-\mathbf{r}) \tag{B.32}$$

$$+ ||\mathbf{r_2}-\mathbf{r}||(\mathbf{r_1}-\mathbf{r})\cdot(\mathbf{r_3}-\mathbf{r}) + ||\mathbf{r_3}-\mathbf{r}||(\mathbf{r_1}-\mathbf{r})\cdot(\mathbf{r_2}-\mathbf{r}).$$

Using Stokes' theorem, the first expression can be simplified as [11, p. 34]

$$\iint\limits_{S_f} \mathbf{n}_f \times \nabla' \frac{1}{||\mathbf{r}'-\mathbf{r}||}\,\mathrm{d}\mathbf{r}' = \sum_{l_e \in \partial S_f} \mathbf{t}_e \int_{l_e} \frac{1}{||\mathbf{r}'-\mathbf{r}||}\,\mathrm{d}\mathbf{r}'. \tag{B.33}$$

Substituting this into the expression for $\nabla W_f(\mathbf{r})$ the following is obtained

$$\nabla W_f(\mathbf{r}) = \sum_{l_e \in \partial S_f} \mathbf{n}_f \times \mathbf{t}_e w_e(\mathbf{r}) + \mathbf{n}_f \Omega(\mathbf{r}, \mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}), \tag{B.34}$$

where $w_e(\mathbf{r}) = \int_{l_e} \frac{1}{||\mathbf{r}'-\mathbf{r}||}\,\mathrm{d}\mathbf{r}'$. This expression is evaluated in the following section.

## B.1.1. Evaluate $w_e(\mathbf{r})$

The calculations in this section are mostly based on [1]. The following integral is evaluated

$$w_e(\mathbf{r}) = \int_{l_e} \frac{1}{||\mathbf{r}'-\mathbf{r}||}\,\mathrm{d}\mathbf{r}' \tag{B.35}$$

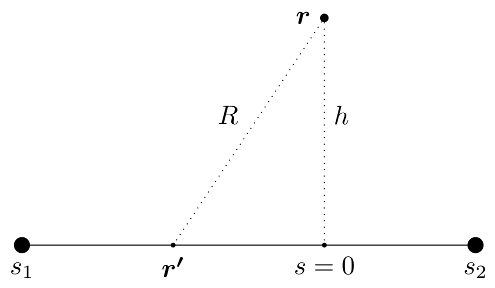
Assume that the three vertices of face $S_f$ lie in the $(x, y)$-plane. As a result,

$$R = |\mathbf{r}'-\mathbf{r}| = \sqrt{(x'-x)^2 + (y'-y)^2 + z^2}. \tag{B.36}$$

Thus, the integral equals

$$\int_{l_e} \frac{1}{R} dr'. \tag{B.37}$$

The situation can be visualized as follows



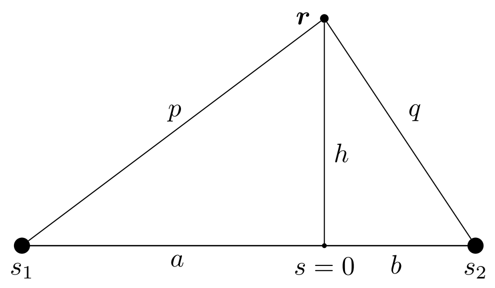
Now, $s_1$ and $s_2$ note the distance from the orthogonal projection of $\mathbf{r}$ on the edge to the endpoints of the edge. $s$ is the distance from the orthogonal projection of $\mathbf{r}$ to $\mathbf{r}'$. Note that

$$R = \sqrt{h^2 + s^2}. \tag{B.38}$$

The integral is calculated as follows

$$\begin{aligned}
\int_{l_e} \frac{dl'}{||\mathbf{r}' - \mathbf{r}||} &= \int_{l_e} \frac{dl'}{R} && \text{(B.39)}\\
&= \int_{s_1}^{s_2} \frac{ds}{R} && \text{(B.40)}\\
&= \int_{s_1}^{s_2} \frac{ds}{\sqrt{h^2 + s^2}} && \text{(B.41)}\\
&= \left[\ln(s + \sqrt{h^2 + s^2})\right]_{s=s_1}^{s=s_2} && \text{(B.42)}\\
&= \ln(\frac{s_2 + \sqrt{h^2 + s_2^2}}{s_1 + \sqrt{h^2 + s_1^2}}) && \text{(B.43)}\\
&= \ln(\frac{b + q}{-a + p}) && \text{(B.44)}
\end{aligned}$$

where $b, q, a, p$ are distances as depicted in the following figure.



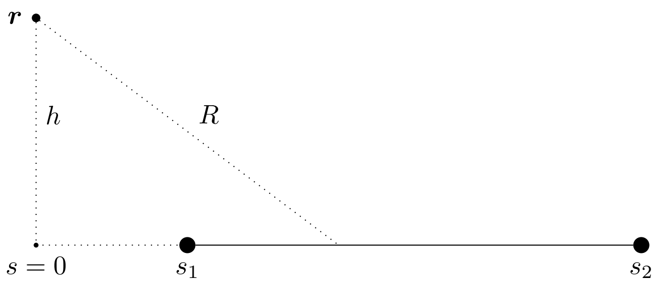
Notice that

$$\begin{aligned}
h^2 &= p^2 - a^2 = (p + a)(p - a) && \text{(B.45)}\\
&= q^2 - b^2 = (q + b)(q - b) && \text{(B.46)}
\end{aligned}$$

This implies

$$\begin{aligned}
(p + a)(p - a) = (q + b)(q - b) &\Rightarrow \frac{q + b}{p - a} = \frac{p + a}{q - b} && \text{(B.47)}\\
\frac{q + b}{p - a} &= \frac{q + b + p + a}{p - a + q - b} && \text{(B.48)}\\
&= \frac{q + p + (a + b)}{p + q - (a + b)} && \text{(B.49)}
\end{aligned}$$

Now, suppose that the orthogonal projection of $\mathbf{r}$ does not fall on the edge. The situations is visualized below.



Again, the integral can be calculated.

$$\int_{l_e} \frac{dl'}{||\mathbf{r}' - \mathbf{r}||} = \int_{l_e} \frac{dl'}{R} \tag{B.50}$$

$$= \int_{s_1}^{s_2} \frac{ds}{R} \tag{B.51}$$

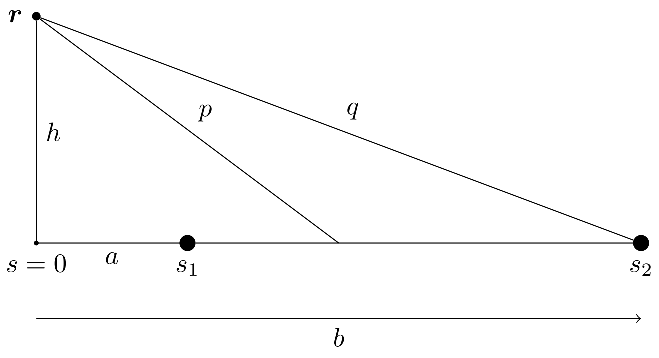
$$= \int_{s_1}^{s_2} \frac{ds}{\sqrt{h^2 + s^2}} \tag{B.52}$$

$$= \Big[\ln(s + \sqrt{h^2 + s^2})\Big]_{s=s_1}^{s=s_2} \tag{B.53}$$

$$= \ln(\frac{s_2 + \sqrt{h^2 + s_2^2}}{s_1 + \sqrt{h^2 + s_1^2}}) \tag{B.54}$$

$$= \ln(\frac{b + q}{a + p}) \tag{B.55}$$

where $b, q, a, p$ are distances as depicted in the following figure.



Again, it can be derived that

$$(p + a)(p - a) = (q + b)(q - b) \Rightarrow \frac{q + b}{p + a} = \frac{p - a}{q - b} \tag{B.56}$$

$$\frac{q + b}{p + a} = \frac{q + b + p - a}{p + a + q - b} \tag{B.57}$$

$$= \frac{q + p + (b - a)}{p + q - (b - a)} \tag{B.58}$$

It can be concluded that for both situations

$$\int_{l_e} \frac{dl'}{||\mathbf{r}' - \mathbf{r}||} = \ln(\frac{|\mathbf{r_1} - \mathbf{r}| + |\mathbf{r_2} - \mathbf{r}| + |\mathbf{r_1} - \mathbf{r_2}|}{|\mathbf{r_1} - \mathbf{r}| + |\mathbf{r_2} - \mathbf{r}| - |\mathbf{r_1} - \mathbf{r_2}|}). \tag{B.59}$$

C

# Analytical Expressions for $I_j$ and $\nabla I_{ij}$

In this appendix, the analytical expressions for the following integrals are presented

$$I_j(\mathbf{r}) = \iint\limits_{\mathcal{S}_j} \frac{1}{||\mathbf{r}-\mathbf{r}'||}\,\mathrm{d}\mathbf{r}' \quad \text{and} \quad \nabla I_{ij}(\mathbf{r}) = \nabla \iint\limits_{\mathcal{S}_j} \frac{\phi_i(\mathbf{r}')}{||\mathbf{r}-\mathbf{r}'||}\,\mathrm{d}\mathbf{r}'. \tag{C.1}$$

The same notation as described in Chapter 4 is used. Remember that for both integrals, a distinguish is made if surface $\mathcal{S}_j$ is triangular or rectangular. The analytical expressions are presented and special cases are considered for both integrals.
Recall that the solid angle subtended to a triangular face with vertices $\mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}$ is denoted by $\Omega(\mathbf{r}, \mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3})$ and equals

$$\Omega(\mathbf{r}, \mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}) = 2\,\mathrm{arctan2}\,\frac{(\mathbf{r_1}-\mathbf{r})\cdot(\mathbf{r_2}-\mathbf{r})\times(\mathbf{r_3}-\mathbf{r})}{D(\mathbf{r}, \mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3})}, \tag{C.2}$$

$$\text{where} \quad D(\mathbf{r}, \mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}) = ||\mathbf{r_1}-\mathbf{r}||||\mathbf{r_2}-\mathbf{r}||||\mathbf{r_3}-\mathbf{r}|| + ||\mathbf{r_1}-\mathbf{r}||(\mathbf{r_2}-\mathbf{r})\cdot(\mathbf{r_3}-\mathbf{r}) + ||\mathbf{r_2}-\mathbf{r}||(\mathbf{r_1}-\mathbf{r})\cdot(\mathbf{r_3}-\mathbf{r}) + ||\mathbf{r_3}-\mathbf{r}||(\mathbf{r_1}-\mathbf{r})\cdot(\mathbf{r_2}-\mathbf{r}). \tag{C.3}$$

## C.1. Special Cases for Evaluating $I_j$

This section focuses on the analytical expression for the integral $I_j$. Remember the following notation

$$L_T(\mathbf{r}, \mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}) = \iint\limits_{\mathcal{T}} \frac{1}{||\mathbf{r}-\mathbf{r}'||}\,\mathrm{d}\mathbf{r}', \quad L_R(\mathbf{r}, \mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}, \mathbf{r_4}) = \iint\limits_{\mathcal{R}} \frac{1}{||\mathbf{r}-\mathbf{r}'||}\,\mathrm{d}\mathbf{r}', \tag{C.4}$$

(C.5)

where $\mathcal{T}$ is a triangle with vertices $\mathbf{r_1}$, $\mathbf{r_2}$ and $\mathbf{r_3}$ and $\mathcal{R}$ is a rectangle with vertices $\mathbf{r_1}$, $\mathbf{r_2}$, $\mathbf{r_3}$ and $\mathbf{r_4}$. The analytical expressions for the integrals in this section are derived by [1]. This section shows the analytical expression and addresses the special cases.

### C.1.1. Special Cases for Evaluating $I_j$ - Triangular Surface

When the surface $\mathcal{S}_j$ is an arbitrary triangular surface with vertices $\mathbf{\bar{r}_1}, \mathbf{\bar{r}_2}, \mathbf{\bar{r}_3}$, the integral $I_j$ can be calculated as follows

$$
\begin{aligned}
L_T(\mathbf{r}, \mathbf{\bar{r}_1}, \mathbf{\bar{r}_2}, \mathbf{\bar{r}_3}) = (\mathbf{n_1} \cdot \mathbf{R_1}) \ln\left(\frac{R_2 + s_{12}}{R_1 + s_{11}}\right) &+ (\mathbf{n_2} \cdot \mathbf{R_2}) \ln\left(\frac{R_3 + s_{23}}{R_2 + s_{22}}\right) + (\mathbf{n_3} \cdot \mathbf{R_3}) \ln\left(\frac{R_1 + s_{31}}{R_3 + s_{33}}\right) \\
&- |\mathbf{n_0} \cdot \mathbf{R_1}| \Omega(\mathbf{r}, \mathbf{\bar{r}_1}, \mathbf{\bar{r}_2}, \mathbf{\bar{r}_3}), \qquad \text{(C.6)}
\end{aligned}
$$

$$
\begin{aligned}
\text{where} \quad & \mathbf{R_i} = \mathbf{\bar{r}_i} - \mathbf{r}, \quad R_i = ||\mathbf{R_i}||, \quad s_{ij} = \tau_\mathbf{i} \cdot \mathbf{R_j}, \quad \mathbf{n_i} = \tau_\mathbf{i} \times \mathbf{n_0}, \quad i, j = 1, 2, 3 \\
& \tau_\mathbf{1} = \frac{\mathbf{\bar{r}_2} - \mathbf{\bar{r}_1}}{||\mathbf{\bar{r}_2} - \mathbf{\bar{r}_1}||}, \quad \tau_\mathbf{2} = \frac{\mathbf{\bar{r}_3} - \mathbf{\bar{r}_2}}{||\mathbf{\bar{r}_3} - \mathbf{\bar{r}_2}||}, \quad \tau_\mathbf{3} = \frac{\mathbf{\bar{r}_1} - \mathbf{\bar{r}_3}}{||\mathbf{\bar{r}_1} - \mathbf{\bar{r}_3}||}, \\
& \mathbf{n_0} = \frac{(\mathbf{\bar{r}_2} - \mathbf{\bar{r}_1}) \times (\mathbf{\bar{r}_3} - \mathbf{\bar{r}_2})}{||(\mathbf{\bar{r}_2} - \mathbf{\bar{r}_1}) \times (\mathbf{\bar{r}_3} - \mathbf{\bar{r}_2})||}. \qquad \text{(C.7)}
\end{aligned}
$$

Suppose $\mathbf{r}$, $\mathbf{\bar{r}_1}$ and $\mathbf{\bar{r}_2}$ are colinear. $\mathbf{r}$ can be written in terms of $\mathbf{\bar{r}_1}$ and $\mathbf{\bar{r}_2}$ as $\mathbf{r} = \mathbf{\bar{r}_1} + a\tau_\mathbf{1}$ for some $a \in \mathbb{R}$. The denominator of the first logarithmic term in Equation C.6 equals

$$
\begin{aligned}
R_1 + s_{11} &= ||\mathbf{R_1}|| + \tau_\mathbf{1} \cdot \mathbf{R_1}, && \text{(C.8)} \\
&= \sqrt{\mathbf{R_1} \cdot \mathbf{R_1}} + \tau_\mathbf{1} \cdot \mathbf{R_1}, && \text{(C.9)} \\
&= \sqrt{(\mathbf{\bar{r}_1} - \mathbf{r}) \cdot (\mathbf{\bar{r}_1} - \mathbf{r})} + \tau_\mathbf{1} \cdot (\mathbf{\bar{r}_1} - \mathbf{r}), && \text{(C.10)} \\
&= \sqrt{(\mathbf{\bar{r}_1} - (\mathbf{\bar{r}_1} + a\tau_\mathbf{1})) \cdot (\mathbf{\bar{r}_1} - (\mathbf{\bar{r}_1} + a\tau_\mathbf{1}))} + \tau_\mathbf{1} \cdot (\mathbf{\bar{r}_1} - (\mathbf{\bar{r}_1} + a\tau_\mathbf{1})), && \text{(C.11)} \\
&= \sqrt{-a\tau_\mathbf{1} \cdot -a\tau_\mathbf{1}} + \tau_\mathbf{1} \cdot -a\tau_\mathbf{1}, && \text{(C.12)} \\
&= a||\tau_\mathbf{1}|| - a||\tau_\mathbf{1}||^2, && \text{(C.13)} \\
&= a - a, && \text{(C.14)} \\
&= 0. && \text{(C.15)}
\end{aligned}
$$

Thus, this term cannot be computed. However, when $\mathbf{r}$, $\mathbf{\bar{r}_1}$ and $\mathbf{\bar{r}_2}$ are colinear, $\mathbf{n_1}$ and $\mathbf{\bar{r}_1} - \mathbf{r}$ are perpendicular to each other. As a result, $\mathbf{n_1} \cdot \mathbf{R_1} = 0$, and $L_T$ reduces to

$$
L_T(\mathbf{r}, \mathbf{\bar{r}_1}, \mathbf{\bar{r}_2}, \mathbf{\bar{r}_3}) = (\mathbf{n_2} \cdot \mathbf{R_2}) \ln\left(\frac{R_3 + s_{23}}{R_2 + s_{22}}\right) + (\mathbf{n_3} \cdot \mathbf{R_3}) \ln\left(\frac{R_1 + s_{31}}{R_3 + s_{33}}\right). \qquad \text{(C.16)}
$$

This situation occurs when the evaluation point is on one of the edges of surface $S_j$. When $\mathbf{r}$, $\mathbf{\bar{r}_1}, \mathbf{\bar{r}_2}$ and $\mathbf{\bar{r}_3}$ are colinear, both $\mathbf{n_1} \cdot \mathbf{R_1}$ and $\mathbf{n_2} \cdot \mathbf{R_2}$ equals zero. Now, $L_T$ reduces to

$$
L_T(\mathbf{r}, \mathbf{\bar{r}_1}, \mathbf{\bar{r}_2}, \mathbf{\bar{r}_3}) = (\mathbf{n_3} \cdot \mathbf{R_3}) \ln\left(\frac{R_1 + s_{31}}{R_3 + s_{33}}\right). \qquad \text{(C.17)}
$$

This is the case when the evaluation point equals one of the vertices of the surface $S_j$.

### C.1.2. Special Cases for Evaluating $I_j$ - Rectangular Surface

When the surface $\mathcal{S}_j$ is an arbitrary rectangular surface with vertices $\mathbf{\bar{r}_1}, \mathbf{\bar{r}_2}$, $\mathbf{\bar{r}_3}$ and $\mathbf{\bar{r}_4}$, the integral $I_j$ can be calculated as follows

$$
\begin{aligned}
L_R(\mathbf{r}, \mathbf{\bar{r}_1}, \mathbf{\bar{r}_2}, \mathbf{\bar{r}_3}, \mathbf{\bar{r}_4}) = (\mathbf{n_1} \cdot \mathbf{R_1}) \ln\left(\frac{R_2 + s_{12}}{R_1 + s_{11}}\right) &+ (\mathbf{n_2} \cdot \mathbf{R_2}) \ln\left(\frac{R_3 + s_{23}}{R_2 + s_{22}}\right) + (\mathbf{n_3} \cdot \mathbf{R_3}) \ln\left(\frac{R_4 + s_{34}}{R_3 + s_{33}}\right) \\
&+ (\mathbf{n_4} \cdot \mathbf{R_4}) \ln\left(\frac{R_1 + s_{41}}{R_4 + s_{44}}\right) - |\mathbf{n_0} \cdot \mathbf{R_1}| \left(\Omega(\mathbf{r}, \mathbf{\bar{r}_1}, \mathbf{\bar{r}_2}, \mathbf{\bar{r}_3}) + \Omega(\mathbf{r}, \mathbf{\bar{r}_1}, \mathbf{\bar{r}_3}, \mathbf{\bar{r}_4})\right), \quad \text{(C.18)}
\end{aligned}
$$

$$
\begin{aligned}
\text{where} \quad & \mathbf{R_i} = \mathbf{\bar{r}_i} - \mathbf{r}, \quad R_i = ||\mathbf{R_i}||, \quad s_{ij} = \tau_\mathbf{i} \cdot \mathbf{R_j}, \quad \mathbf{n_i} = \tau_\mathbf{i} \times \mathbf{n_0}, \quad i, j = 1, 2, 3, 4 \\
& \tau_\mathbf{1} = \frac{\mathbf{\bar{r}_2} - \mathbf{\bar{r}_1}}{||\mathbf{\bar{r}_2} - \mathbf{\bar{r}_1}||}, \quad \tau_\mathbf{2} = \frac{\mathbf{\bar{r}_3} - \mathbf{\bar{r}_2}}{||\mathbf{\bar{r}_3} - \mathbf{\bar{r}_2}||}, \quad \tau_\mathbf{3} = \frac{\mathbf{\bar{r}_4} - \mathbf{\bar{r}_3}}{||\mathbf{\bar{r}_4} - \mathbf{\bar{r}_3}||}, \quad \tau_\mathbf{4} = \frac{\mathbf{\bar{r}_1} - \mathbf{\bar{r}_4}}{||\mathbf{\bar{r}_1} - \mathbf{\bar{r}_4}||}, \\
& \mathbf{n_0} = \frac{(\mathbf{\bar{r}_2} - \mathbf{\bar{r}_1}) \times (\mathbf{\bar{r}_3} - \mathbf{\bar{r}_2})}{||(\mathbf{\bar{r}_2} - \mathbf{\bar{r}_1}) \times (\mathbf{\bar{r}_3} - \mathbf{\bar{r}_2})||}. \qquad \text{(C.19)}
\end{aligned}
$$

The special cases are comparable to the cases mentioned in Section C.1.1. For example, when $\mathbf{r}$, $\bar{\mathbf{r}}_1$ and $\bar{\mathbf{r}}_2$ are colinear, $(\mathbf{n_1} \cdot \mathbf{R_1}) = 0$ and $L_R$ reduces to

$$L_R(\mathbf{r}, \bar{\mathbf{r}}_1, \bar{\mathbf{r}}_2, \bar{\mathbf{r}}_3, \bar{\mathbf{r}}_4) = (\mathbf{n_2} \cdot \mathbf{R_2}) \ln\left(\frac{R_3 + s_{23}}{R_2 + s_{22}}\right) + (\mathbf{n_3} \cdot \mathbf{R_3}) \ln\left(\frac{R_4 + s_{34}}{R_3 + s_{33}}\right) + (\mathbf{n_4} \cdot \mathbf{R_4}) \ln\left(\frac{R_1 + s_{41}}{R_4 + s_{44}}\right) \tag{C.20}$$

## C.2. Special Cases for Evaluating $\nabla I_{ij}$

This section focuses on the analytical expression for the integral $\nabla I_{ij}$. Remember the following notation

$$K_T(\mathbf{r}, \mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}) = \iint_{\mathcal{T}} \frac{\phi_1(\mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||}\, \mathrm{d}\mathbf{r}', \quad K_R(\mathbf{r}, \mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}, \mathbf{r_4}) = \iint_{\mathcal{R}} \frac{\phi_{12}(\mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||}\, \mathrm{d}\mathbf{r}', \tag{C.21}$$

where $\mathcal{T}$ is a triangle with vertices $\mathbf{r_1}$, $\mathbf{r_2}$ and $\mathbf{r_3}$ and $\mathcal{R}$ is a rectangle with vertices $\mathbf{r_1}$, $\mathbf{r_2}$, $\mathbf{r_3}$ and $\mathbf{r_4}$. In this notation, the second argument of $K_T$ denotes where the linear function $\phi$ assumes the value 1. For $K_R$ the second and third argument indicate the vertices where the linear function $\phi$ assumes the value 1. A detailed derivation of the integrals is done by [1]. Numerical experiments demonstrate that when combining these expressions with those from [7], the function `hcubature()` returns finite values and avoids `NaN` values at smaller relative tolerances compared to using only the expressions from [1]. This section shows how the expressions are combined and addresses the special cases.

### C.2.1. Special Cases for Evaluating $\nabla I_{ij}$ - Triangular Surface

When the surface $\mathcal{S}_j$ is an arbitrary triangular surface with vertices $\bar{\mathbf{r}}_1, \bar{\mathbf{r}}_2, \bar{\mathbf{r}}_3$, the integral $I_{ij}$ can be calculated as follows

$$\begin{aligned}
\nabla K_T(\mathbf{r}, \bar{\mathbf{r}}_1, \bar{\mathbf{r}}_2, \bar{\mathbf{r}}_3) = {} & \frac{\mathbf{n_2} L_T(\mathbf{r}, \bar{\mathbf{r}}_1, \bar{\mathbf{r}}_2, \bar{\mathbf{r}}_3)}{(\bar{\mathbf{r}}_1 - \bar{\mathbf{r}}_2) \cdot \mathbf{n_2}} + \mathbf{n_0}\, \mathrm{sign}(\mathbf{n_0} \cdot \mathbf{R_1}) \phi_1(\mathbf{r}) \Omega(\mathbf{r}, \bar{\mathbf{r}}_1, \bar{\mathbf{r}}_2, \bar{\mathbf{r}}_3) \\
& - \frac{\mathbf{n_0}(\mathbf{n_0} \cdot \mathbf{R_1})}{\mathbf{n_2} \cdot (\bar{\mathbf{r}}_1 - \bar{\mathbf{r}}_2)} \left[ (\mathbf{n_2} \cdot \mathbf{n_1}) \ln\left(\frac{R_2 + s_{12}}{R_1 + s_{11}}\right) + \ln\left(\frac{R_3 + s_{23}}{R_2 + s_{22}}\right) \right. \\
& \left. + (\mathbf{n_2} \cdot \mathbf{n_3}) \ln\left(\frac{R_1 + s_{31}}{R_3 + s_{33}}\right) \right] - \frac{\mathbf{n_1}}{s_{11} - s_{12}} \left[ R_2 - R_1 - s_{12} \ln\left(\frac{R_2 + R_1 + ||\bar{\mathbf{r}}_2 - \bar{\mathbf{r}}_1||}{R_2 + R_1 - ||\bar{\mathbf{r}}_2 - \bar{\mathbf{r}}_1||}\right) \right] \\
& - \frac{\mathbf{n_3}}{s_{31} - s_{33}} \left[ R_1 - R_3 - s_{33} \ln\left(\frac{R_1 + R_3 + ||\bar{\mathbf{r}}_1 - \bar{\mathbf{r}}_3||}{R_1 + R_3 - ||\bar{\mathbf{r}}_1 - \bar{\mathbf{r}}_3||}\right) \right],
\end{aligned} \tag{C.22}$$

$$\begin{aligned}
\text{where} \quad & \mathbf{R_i} = \bar{\mathbf{r}}_\mathbf{i} - \mathbf{r}, \quad R_i = ||\mathbf{R_i}||, \quad s_{ij} = \tau_\mathbf{i} \cdot \mathbf{R_j}, \quad \mathbf{n_i} = \tau_\mathbf{i} \times \mathbf{n_0}, \quad i, j = 1, 2, 3 \\
& \tau_\mathbf{1} = \frac{\bar{\mathbf{r}}_2 - \bar{\mathbf{r}}_1}{||\bar{\mathbf{r}}_2 - \bar{\mathbf{r}}_1||}, \quad \tau_\mathbf{2} = \frac{\bar{\mathbf{r}}_3 - \bar{\mathbf{r}}_2}{||\bar{\mathbf{r}}_3 - \bar{\mathbf{r}}_2||}, \quad \tau_\mathbf{3} = \frac{\bar{\mathbf{r}}_1 - \bar{\mathbf{r}}_3}{||\bar{\mathbf{r}}_1 - \bar{\mathbf{r}}_3||}, \\
& \mathbf{n_0} = \frac{(\bar{\mathbf{r}}_2 - \bar{\mathbf{r}}_1) \times (\bar{\mathbf{r}}_3 - \bar{\mathbf{r}}_2)}{||(\bar{\mathbf{r}}_2 - \bar{\mathbf{r}}_1) \times (\bar{\mathbf{r}}_3 - \bar{\mathbf{r}}_2)||}.
\end{aligned} \tag{C.23}$$

Suppose $\mathbf{r}$, $\bar{\mathbf{r}}_1$ and $\bar{\mathbf{r}}_2$ are colinear, which implies that $R_1 + s_{11} = R_2 + R_1 - ||\bar{\mathbf{r}}_2 - \bar{\mathbf{r}}_1|| = 0$. The challenges encountered in the evaluation of $L_T$ are discussed in Section C.1.1. The first logarithmic term that contains the expression vanishes, due to the fact that $\mathbf{n_0} \cdot \mathbf{R_1}$ equals 0. However, the second logarithmic term that contains the expression cannot be evaluated. The integral $\nabla I_{ij}$ over a triangular surface cannot be computed when the evaluation point lies on one of the edges of the surface or when the evaluation point equals a vertex.

## C.2.2. Special Cases for Evaluating $\nabla I_{ij}$ - Rectangular Surface

When the surface $\mathcal{S}_j$ is an arbitrary rectangular surface with vertices $\bar{\mathbf{r}}_1, \bar{\mathbf{r}}_2$, $\bar{\mathbf{r}}_3$ and $\bar{\mathbf{r}}_4$, the integral $I_{ij}$ can be calculated as follows

$$\nabla K_R(\mathbf{r},\bar{\mathbf{r}}_1,\bar{\mathbf{r}}_2,\bar{\mathbf{r}}_3,\bar{\mathbf{r}}_4) = \nabla K_T(\mathbf{r},\bar{\mathbf{r}}_2,\bar{\mathbf{r}}_3,\bar{\mathbf{r}}_4) + \nabla L_T(\mathbf{r},\bar{\mathbf{r}}_1,\bar{\mathbf{r}}_2,\bar{\mathbf{r}}_4) - \nabla K_T(\mathbf{r},\bar{\mathbf{r}}_4,\bar{\mathbf{r}}_1,\bar{\mathbf{r}}_2), \tag{C.24}$$

$$\begin{aligned}\text{where}\quad \nabla K_T(\mathbf{r},\bar{\mathbf{r}}_1,\bar{\mathbf{r}}_2,\bar{\mathbf{r}}_3) &= \frac{\mathbf{n_2} L_T(\mathbf{r},\bar{\mathbf{r}}_1,\bar{\mathbf{r}}_2,\bar{\mathbf{r}}_3)}{(\bar{\mathbf{r}}_1-\bar{\mathbf{r}}_2)\cdot\mathbf{n_2}} + \mathbf{n_0}\,\text{sign}(\mathbf{n_0}\cdot\mathbf{R_1})\phi_1(\mathbf{r})\Omega(\mathbf{r},\bar{\mathbf{r}}_1,\bar{\mathbf{r}}_2,\bar{\mathbf{r}}_3)\\ &- \frac{\mathbf{n_0}(\mathbf{n_0}\cdot\mathbf{R_1})}{\mathbf{n_2}\cdot(\bar{\mathbf{r}}_1-\bar{\mathbf{r}}_2)}\left[(\mathbf{n_2}\cdot\mathbf{n_1})\ln\left(\frac{R_2+s_{12}}{R_1+s_{11}}\right) + \ln\left(\frac{R_3+s_{23}}{R_2+s_{22}}\right)\right.\\ &\left.+(\mathbf{n_2}\cdot\mathbf{n_3})\ln\left(\frac{R_1+s_{31}}{R_3+s_{33}}\right)\right] - \frac{\mathbf{n_1}}{s_{11}-s_{12}}\left[R_2-R_1-s_{12}\ln\left(\frac{R_2+R_1+||\bar{\mathbf{r}}_2-\bar{\mathbf{r}}_1||}{R_2+R_1-||\bar{\mathbf{r}}_2-\bar{\mathbf{r}}_1||}\right)\right]\\ &- \frac{\mathbf{n_3}}{s_{31}-s_{33}}\left[R_1-R_3-s_{33}\ln\left(\frac{R_1+R_3+||\bar{\mathbf{r}}_1-\bar{\mathbf{r}}_3||}{R_1+R_3-||\bar{\mathbf{r}}_1-\bar{\mathbf{r}}_3||}\right)\right],\end{aligned} \tag{C.25}$$

$$\begin{aligned}\text{and}\quad \nabla L_T(\mathbf{r},\bar{\mathbf{r}}_1,\bar{\mathbf{r}}_2,\bar{\mathbf{r}}_3) &= -\mathbf{n_1}\ln\left(\frac{R_2+R_1+||\bar{\mathbf{r}}_2-\bar{\mathbf{r}}_1||}{R_2+R_1-||\bar{\mathbf{r}}_2-\bar{\mathbf{r}}_1||}\right) - \mathbf{n_2}\ln\left(\frac{R_3+R_2+||\bar{\mathbf{r}}_3-\bar{\mathbf{r}}_2||}{R_3+R_2-||\bar{\mathbf{r}}_3-\bar{\mathbf{r}}_2||}\right)\\ &- \mathbf{n_3}\ln\left(\frac{R_1+R_3+||\bar{\mathbf{r}}_1-\bar{\mathbf{r}}_3||}{R_1+R_3-||\bar{\mathbf{r}}_1-\bar{\mathbf{r}}_3||}\right) + \mathbf{n_0}\text{sign}(\mathbf{n_0}\cdot\mathbf{R_1})\Omega(\mathbf{r},\bar{\mathbf{r}}_1,\bar{\mathbf{r}}_2,\bar{\mathbf{r}}_3),\end{aligned} \tag{C.26}$$

$$\begin{aligned}\text{where}\quad \mathbf{R_i} &= \bar{\mathbf{r}}_\mathbf{i} - \mathbf{r},\quad R_i = ||\mathbf{R_i}||,\quad s_{ij} = \tau_\mathbf{i}\cdot\mathbf{R_j},\quad \mathbf{n_i} = \tau_\mathbf{i}\times\mathbf{n_0},\quad i,j=1,2,3\\ \tau_1 &= \frac{\bar{\mathbf{r}}_2-\bar{\mathbf{r}}_1}{||\bar{\mathbf{r}}_2-\bar{\mathbf{r}}_1||},\quad \tau_2 = \frac{\bar{\mathbf{r}}_3-\bar{\mathbf{r}}_2}{||\bar{\mathbf{r}}_3-\bar{\mathbf{r}}_2||},\quad \tau_3 = \frac{\bar{\mathbf{r}}_1-\bar{\mathbf{r}}_3}{||\bar{\mathbf{r}}_1-\bar{\mathbf{r}}_3||},\\ \mathbf{n_0} &= \frac{(\bar{\mathbf{r}}_2-\bar{\mathbf{r}}_1)\times(\bar{\mathbf{r}}_3-\bar{\mathbf{r}}_2)}{||(\bar{\mathbf{r}}_2-\bar{\mathbf{r}}_1)\times(\bar{\mathbf{r}}_3-\bar{\mathbf{r}}_2)||}.\end{aligned} \tag{C.27}$$

Suppose that the vertices of $\mathcal{S}_1$ are $\mathbf{r}'_1, \mathbf{r}'_2, \mathbf{r}'_3$ and $\mathbf{r}'_4$. The evaluation points that are on this surface are $\mathbf{r_1}$ and $\mathbf{r_2}$ for $E_1$, and $\frac{\mathbf{r_1}+\mathbf{r_2}}{2}$ for $E_2$. Note that for the calculation of $\nabla K_R$, the surface is split into two triangles. The first triangle has vertices $\mathbf{r}'_2, \mathbf{r}'_3, \mathbf{r}'_4$ and the second triangle has vertices $\mathbf{r}'_1, \mathbf{r}'_2, \mathbf{r}'_4$. The figure below shows a sketch of the situation.

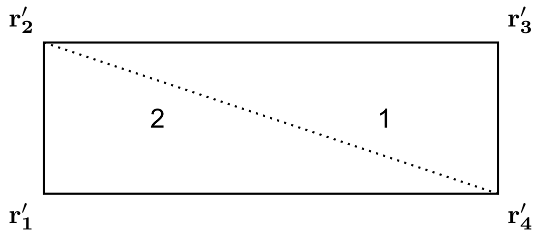


**Figure C.1:** Sketch surface $\mathcal{S}_1$

The vector $\mathbf{n_0}$ is the same for both triangles. However, the other variables in the two expressions $K_T$ depend on the triangle. The superscripts 1 or 2 denote the specific surface to which each variable belongs. From now on, $\Omega(\mathbf{r},\mathbf{r}'_1,\mathbf{r}'_2,\mathbf{r}'_3)$ and $\Omega(\mathbf{r},\mathbf{r}'_4,\mathbf{r}'_1,\mathbf{r}'_2)$ will be denoted by $\Omega^1$ and $\Omega^2$, respectively.

Using this notation, the expressions for $\nabla K_T$ for the two surfaces can be written as

$$\begin{aligned}
\nabla K_T(\mathbf{r},\mathbf{r_2'},\mathbf{r_3'},\mathbf{r_4'}) = {} & \frac{\mathbf{n_2^1} L_T(\mathbf{r},\mathbf{r_2'},\mathbf{r_3'},\mathbf{r_4'})}{(\mathbf{r_2'}-\mathbf{r_3'})\cdot\mathbf{n_2^1}} + \mathbf{n_0}\,\text{sign}(\mathbf{n_0}\cdot\mathbf{R_1^1})\phi_1^1(\mathbf{r})\Omega^1 \\
& - \frac{\mathbf{n_0}(\mathbf{n_0}\cdot\mathbf{R_1^1})}{\mathbf{n_2^1}\cdot(\mathbf{r_2'}-\mathbf{r_3'})}\left[(\mathbf{n_2^1}\cdot\mathbf{n_1^1})\ln\left(\frac{R_2^1+s_{12}^1}{R_1^1+s_{11}^1}\right)+\ln\left(\frac{R_3^1+s_{23}^1}{R_2^1+s_{22}^1}\right)\right. \\
& \left.+(\mathbf{n_2^1}\cdot\mathbf{n_3^1})\ln\left(\frac{R_1^1+s_{31}^1}{R_3^1+s_{33}^1}\right)\right] - \frac{\mathbf{n_1^1}}{s_{11}^1-s_{12}^1}\left[R_2^1-R_1^1-s_{12}^1\ln\left(\frac{R_2^1+R_1^1+||\mathbf{r_3'}-\mathbf{r_2'}||}{R_2^1+R_1^1-||\mathbf{r_3'}-\mathbf{r_2'}||}\right)\right] \\
& - \frac{\mathbf{n_3^1}}{s_{31}^1-s_{33}^1}\left[R_1^1-R_3^1-s_{33}^1\ln\left(\frac{R_1^1++R_3^1+||\mathbf{r_4'}-\mathbf{r_2'}||}{R_1^1+R_3^1-||\mathbf{r_4'}-\mathbf{r_2'}||}\right)\right], \qquad \text{(C.28)} \\
\nabla K_T(\mathbf{r},\mathbf{r_4'},\mathbf{r_1'},\mathbf{r_2'}) = {} & \frac{\mathbf{n_2^2} L_T(\mathbf{r},\mathbf{r_4'},\mathbf{r_1'},\mathbf{r_2'})}{(\mathbf{r_4'}-\mathbf{r_1'})\cdot\mathbf{n_2^2}} + \mathbf{n_0}\,\text{sign}(\mathbf{n_0}\cdot\mathbf{R_1^2})\phi_1^2(\mathbf{r})\Omega^2 \\
& - \frac{\mathbf{n_0}(\mathbf{n_0}\cdot\mathbf{R_1^2})}{\mathbf{n_2^2}\cdot(\mathbf{r_4'}-\mathbf{r_1'})}\left[(\mathbf{n_2^2}\cdot\mathbf{n_1^2})\ln\left(\frac{R_2^2+s_{12}^2}{R_1^2+s_{11}^2}\right)+\ln\left(\frac{R_3^2+s_{23}^2}{R_2^2+s_{22}^2}\right)\right. \\
& \left.+(\mathbf{n_2^2}\cdot\mathbf{n_3^2})\ln\left(\frac{R_1^2+s_{31}^2}{R_3^2+s_{33}^2}\right)\right] - \frac{\mathbf{n_1^2}}{s_{11}^2-s_{12}^2}\left[R_2^2-R_1^2-s_{12}^2\ln\left(\frac{R_2^2+R_1^2+||\mathbf{r_1'}-\mathbf{r_4'}||}{R_2^2+R_1^2-||\mathbf{r_1'}-\mathbf{r_4'}||}\right)\right] \\
& - \frac{\mathbf{n_3^2}}{s_{31}^2-s_{33}^2}\left[R_1^2-R_3^2-s_{33}^2\ln\left(\frac{R_1^2+R_3^2+||\mathbf{r_2'}-\mathbf{r_4'}||}{R_1^2+R_3^2+||\mathbf{r_2'}-\mathbf{r_4'}||}\right)\right], \qquad \text{(C.29)}
\end{aligned}$$

where the variables in the expressions can be calculated using the equations in C.27. Using this notation, $\nabla L_T(\mathbf{r},\mathbf{r_1'},\mathbf{r_2'},\mathbf{r_4'})$ can be written as

$$\begin{aligned}
\nabla L_T(\mathbf{r},\mathbf{r_1'},\mathbf{r_2'},\mathbf{r_4'}) = {} & -\mathbf{n_2^2}\ln\left(\frac{R_3^2+R_2^2+||\mathbf{r_2'}-\mathbf{r_1'}||}{R_3^2+R_2^2-||\mathbf{r_2'}-\mathbf{r_1'}||}\right) - \mathbf{n_3^2}\ln\left(\frac{R_1^2+R_3^2+||\mathbf{r_4'}-\mathbf{r_2'}||}{R_1^2+R_3^2-||\mathbf{r_4'}-\mathbf{r_2'}||}\right) \\
& - \mathbf{n_1^2}\ln\left(\frac{R_2^2+R_1^2+||\mathbf{r_4'}-\mathbf{r_1'}||}{R_2^2+R_1^2+||\mathbf{r_4'}-\mathbf{r_1'}||}\right) + \mathbf{n_0}\text{sign}(\mathbf{n_0}\cdot\mathbf{R_2^2})\Omega^2. \qquad \text{(C.30)}
\end{aligned}$$

When the evaluation point lies on one of the edges of the surface or equals a vertex, $\nabla K_T$ cannot be evaluated for one of the two triangles. As mentioned is Section C.2.1, one of the logarithmic term cannot be evaluated. Thus, it is impossible to evaluate this expression for the evaluation points in $E_1$. Now, consider the case when $\mathbf{r}$ is colinear with $\mathbf{r_2'}$ and $\mathbf{r_4'}$. Note that this implies $R_1^1+s_{33}^1 = R_3^2+s_{33}^2 = R_1^1+R_3^1-||\mathbf{r_4'}-\mathbf{r_2'}|| = R_1^2+R_3^2-||\mathbf{r_4'}-\mathbf{r_2'}|| = 0$. The logarithmic terms containing $R_1^1+s_{33}^1$ and $R_3^2+s_{33}^2$

disappear, since $\mathbf{n_0} \cdot \mathbf{R_1^1} = \mathbf{n_0} \cdot \mathbf{R_1^2} = 0$. Observe the following

$$\begin{aligned}
\mathbf{n_3^1} &= \frac{\mathbf{r'_2} - \mathbf{r'_4}}{||\mathbf{r'_2} - \mathbf{r'_4}||} \times \mathbf{n_0}, && \text{(C.31)}\\
&= -\frac{\mathbf{r'_4} - \mathbf{r'_2}}{||\mathbf{r'_2} - \mathbf{r'_4}||} \times \mathbf{n_0} && \text{(C.32)}\\
&= -\mathbf{n_3^2}, && \text{(C.33)}\\
s_{31}^1 &= \frac{\mathbf{r'_2} - \mathbf{r'_4}}{||\mathbf{r'_2} - \mathbf{r'_4}||} \cdot (\mathbf{r'_2} - \mathbf{r}), && \text{(C.34)}\\
&= \frac{\mathbf{r'_2} - \mathbf{r'_4}}{||\mathbf{r'_2} - \mathbf{r'_4}||} \cdot (\frac{1}{2}\mathbf{r'_2} - \frac{1}{2}\mathbf{r'_4}), && \text{(C.35)}\\
&= -\frac{\mathbf{r'_2} - \mathbf{r'_4}}{||\mathbf{r'_2} - \mathbf{r'_4}||} \cdot (\frac{1}{2}\mathbf{r'_4} - \frac{1}{2}\mathbf{r'_2}), && \text{(C.36)}\\
&= -\frac{\mathbf{r'_2} - \mathbf{r'_4}}{||\mathbf{r'_2} - \mathbf{r'_4}||} \cdot (\mathbf{r'_4} - \mathbf{r}), && \text{(C.37)}\\
&= -s_{33}^1, && \text{(C.38)}\\
s_{31}^2 &= \frac{\mathbf{r'_4} - \mathbf{r'_2}}{||\mathbf{r'_4} - \mathbf{r'_2}||} \cdot (\mathbf{r'_4} - \mathbf{r}), && \text{(C.39)}\\
&= \frac{\mathbf{r'_2} - \mathbf{r'_4}}{||\mathbf{r'_2} - \mathbf{r'_4}||} \cdot (\mathbf{r'_2} - \mathbf{r}), && \text{(C.40)}\\
&= -s_{33}^2, && \text{(C.41)}\\
&\Rightarrow s_{31}^1 = s_{31}^2 = -s_{33}^1 = -s_{33}^2, && \text{(C.42)}\\
R_1^1 &= (\mathbf{r'_2} - \mathbf{r}), && \text{(C.43)}\\
&= R_3^2, && \text{(C.44)}\\
R_3^1 &= (\mathbf{r'_4} - \mathbf{r}), && \text{(C.45)}\\
&= R_1^2. && \text{(C.46)}
\end{aligned}$$

Taking this into account, $K_R$ reduces as follows

$$
\begin{aligned}
\nabla K_R(\mathbf{r},\mathbf{r'_1},\mathbf{r'_2},\mathbf{r'_3},\mathbf{r'_4}) &= \nabla K_T(\mathbf{r},\mathbf{r'_2},\mathbf{r'_3},\mathbf{r'_4}) + \nabla L_T(\mathbf{r},\mathbf{r_1},\mathbf{r'_2},\mathbf{r'_4}) - \nabla K_T(\mathbf{r},\mathbf{r'_4},\mathbf{r'_1},\mathbf{r'_2}), \qquad \text{(C.47)}\\
&= \frac{\mathbf{n_2^1} L_T(\mathbf{r},\mathbf{r'_2},\mathbf{r'_3},\mathbf{r'_4})}{(\mathbf{r'_2}-\mathbf{r'_3})\cdot\mathbf{n_2^1}} + \mathbf{n_0}\,\mathsf{sign}(\mathbf{n_0}\cdot\mathbf{R_1^1})\phi_1^1(\mathbf{r})\Omega^1\\
&\quad - \frac{\mathbf{n_1^1}}{s_{11}^1 - s_{12}^1}\left[R_2^1 - R_1^1 - s_{12}^1 \ln\left(\frac{R_2^1+R_1^1+||\mathbf{r'_3}-\mathbf{r'_2}||}{R_2^1+R_1^1-||\mathbf{r'_3}-\mathbf{r'_2}||}\right)\right]\\
&\quad \underline{-\frac{\mathbf{n_3^1}}{s_{31}^1 - s_{33}^1}\left[R_1^1 - R_3^1 - s_{31}^1 \ln\left(\frac{R_1^1+R_3^1+||\mathbf{r'_4}-\mathbf{r'_2}||}{R_1^1+R_3^1-||\mathbf{r'_4}-\mathbf{r'_2}||}\right)\right]} - \mathbf{n_2^2}\ln\left(\frac{R_3^2+R_2^2+||\mathbf{r'_2}-\mathbf{r'_1}||}{R_3^2+R_2^2-||\mathbf{r'_2}-\mathbf{r'_1}||}\right)\\
&\quad \underline{- \mathbf{n_3^2}\ln\left(\frac{R_1^2+R_3^2+||\mathbf{r'_4}-\mathbf{r'_2}||}{R_1^2+R_3^2-||\mathbf{r'_4}-\mathbf{r'_2}||}\right)} - \mathbf{n_1^2}\ln\left(\frac{R_2^2+R_1^2+||\mathbf{r'_4}-\mathbf{r'_1}||}{R_2^2+R_1^2+||\mathbf{r'_4}-\mathbf{r'_1}||}\right) + \mathbf{n_0}\mathsf{sign}(\mathbf{n_0}\cdot\mathbf{R_2^2})\Omega^2\\
&\quad - \frac{\mathbf{n_2^2} L_T(\mathbf{r},\mathbf{r'_4},\mathbf{r'_1},\mathbf{r'_2})}{(\mathbf{r'_4}-\mathbf{r'_1})\cdot\mathbf{n_2^2}} - \mathbf{n_0}\,\mathsf{sign}(\mathbf{n_0}\cdot\mathbf{R_1^2})\phi_1^2(\mathbf{r})\Omega^2\\
&\quad + \frac{\mathbf{n_1^2}}{s_{11}^2 - s_{12}^2}\left[R_2^2 - R_1^2 - s_{12}^2 \ln\left(\frac{R_2^2+R_1^2+||\mathbf{r'_1}-\mathbf{r'_4}||}{R_2^2+R_1^2-||\mathbf{r'_1}-\mathbf{r'_4}||}\right)\right]\\
&\quad \underline{+ \frac{\mathbf{n_3^2}}{s_{31}^2 - s_{33}^2}\left[R_1^2 - R_3^2 - s_{33}^2 \ln\left(\frac{R_1^2+R_3^2+||\mathbf{r'_2}-\mathbf{r'_4}||}{R_1^2+R_3^2+||\mathbf{r'_2}-\mathbf{r'_4}||}\right)\right]}, \qquad \text{(C.48)}\\
&= \frac{\mathbf{n_2^1} L_T(\mathbf{r},\mathbf{r'_2},\mathbf{r'_3},\mathbf{r'_4})}{(\mathbf{r'_2}-\mathbf{r'_3})\cdot\mathbf{n_2^1}} + \mathbf{n_0}\,\mathsf{sign}(\mathbf{n_0}\cdot\mathbf{R_1^1})\phi_1^1(\mathbf{r})\Omega^1\\
&\quad - \frac{\mathbf{n_1^1}}{s_{11}^1 - s_{12}^1}\left[R_2^1 - R_1^1 - s_{12}^1 \ln\left(\frac{R_2^1+R_1^1+||\mathbf{r'_3}-\mathbf{r'_2}||}{R_2^1+R_1^1-||\mathbf{r'_3}-\mathbf{r'_2}||}\right)\right]\\
&\quad - \mathbf{n_2^2}\ln\left(\frac{R_3^2+R_2^2+||\mathbf{r'_2}-\mathbf{r'_1}||}{R_3^2+R_2^2-||\mathbf{r'_2}-\mathbf{r'_1}||}\right) - \mathbf{n_1^2}\ln\left(\frac{R_2^2+R_1^2+||\mathbf{r'_4}-\mathbf{r'_1}||}{R_2^2+R_1^2+||\mathbf{r'_4}-\mathbf{r'_1}||}\right) + \mathbf{n_0}\mathsf{sign}(\mathbf{n_0}\cdot\mathbf{R_2^2})\Omega^2\\
&\quad - \frac{\mathbf{n_2^2} L_T(\mathbf{r},\mathbf{r'_4},\mathbf{r'_1},\mathbf{r'_2})}{(\mathbf{r'_4}-\mathbf{r'_1})\cdot\mathbf{n_2^2}} - \mathbf{n_0}\,\mathsf{sign}(\mathbf{n_0}\cdot\mathbf{R_1^2})\phi_1^2(\mathbf{r})\Omega^2\\
&\quad + \frac{\mathbf{n_1^2}}{s_{11}^2 - s_{12}^2}\left[R_2^2 - R_1^2 - s_{12}^2 \ln\left(\frac{R_2^2+R_1^2+||\mathbf{r'_1}-\mathbf{r'_4}||}{R_2^2+R_1^2-||\mathbf{r'_1}-\mathbf{r'_4}||}\right)\right]\\
&\quad \underline{+\frac{\mathbf{n_3^2}}{s_{31}^2 - s_{33}^2}\left[R_3^2 - R_1^2 - s_{33}^2 \ln\left(\frac{R_1^2++R_3^2+||\mathbf{r'_4}-\mathbf{r'_2}||}{R_1^1+R_3^1-||\mathbf{r'_4}-\mathbf{r'_2}||}\right)\right]}\\
&\quad \underline{+ \frac{\mathbf{n_3^2}}{s_{31}^2 - s_{33}^2}\left[R_1^2 - R_3^2 - s_{33}^2 \ln\left(\frac{R_1^2+R_3^2+||\mathbf{r'_2}-\mathbf{r'_4}||}{R_1^2+R_3^2+||\mathbf{r'_2}-\mathbf{r'_4}||}\right)\right]} \underline{- \mathbf{n_3^2}\ln\left(\frac{R_1^2+R_3^2+||\mathbf{r'_4}-\mathbf{r'_2}||}{R_1^2+R_3^2-||\mathbf{r'_4}-\mathbf{r'_2}||}\right)},\\
&\qquad \text{(C.49)}\\
&= \frac{\mathbf{n_2^1} L_T(\mathbf{r},\mathbf{r'_2},\mathbf{r'_3},\mathbf{r'_4})}{(\mathbf{r'_2}-\mathbf{r'_3})\cdot\mathbf{n_2^1}} + \mathbf{n_0}\,\mathsf{sign}(\mathbf{n_0}\cdot\mathbf{R_1^1})\phi_1^1(\mathbf{r})\Omega^1\\
&\quad - \frac{\mathbf{n_1^1}}{s_{11}^1 - s_{12}^1}\left[R_2^1 - R_1^1 - s_{12}^1 \ln\left(\frac{R_2^1+R_1^1+||\mathbf{r'_3}-\mathbf{r'_2}||}{R_2^1+R_1^1-||\mathbf{r'_3}-\mathbf{r'_2}||}\right)\right]\\
&\quad - \mathbf{n_2^2}\ln\left(\frac{R_3^2+R_2^2+||\mathbf{r'_2}-\mathbf{r'_1}||}{R_3^2+R_2^2-||\mathbf{r'_2}-\mathbf{r'_1}||}\right) - \mathbf{n_1^2}\ln\left(\frac{R_2^2+R_1^2+||\mathbf{r'_4}-\mathbf{r'_1}||}{R_2^2+R_1^2+||\mathbf{r'_4}-\mathbf{r'_1}||}\right) + \mathbf{n_0}\mathsf{sign}(\mathbf{n_0}\cdot\mathbf{R_2^2})\Omega^2\\
&\quad - \frac{\mathbf{n_2^2} L_T(\mathbf{r},\mathbf{r'_4},\mathbf{r'_1},\mathbf{r'_2})}{(\mathbf{r'_4}-\mathbf{r'_1})\cdot\mathbf{n_2^2}} - \mathbf{n_0}\,\mathsf{sign}(\mathbf{n_0}\cdot\mathbf{R_1^2})\phi_1^2(\mathbf{r})\Omega^2\\
&\quad + \frac{\mathbf{n_1^2}}{s_{11}^2 - s_{12}^2}\left[R_2^2 - R_1^2 - s_{12}^2 \ln\left(\frac{R_2^2+R_1^2+||\mathbf{r'_1}-\mathbf{r'_4}||}{R_2^2+R_1^2-||\mathbf{r'_1}-\mathbf{r'_4}||}\right)\right]\\
&\quad - \frac{(2s_{33}^2 + s_{31}^2 - s_{33}^2)\mathbf{n_3^2}}{s_{31}^2 - s_{33}^2}\ln\left(\frac{R_1^2+R_3^2+||\mathbf{r'_4}-\mathbf{r'_2}||}{R_1^2+R_3^2-||\mathbf{r'_4}-\mathbf{r'_2}||}\right), \qquad \text{(C.50)}\\
2s_{33}^2 + s_{31}^2 - s_{33}^2 = 0 \Rightarrow &= \frac{\mathbf{n_2^1} L_T(\mathbf{r},\mathbf{r'_2},\mathbf{r'_3},\mathbf{r'_4})}{(\mathbf{r'_2}-\mathbf{r'_3})\cdot\mathbf{n_2^1}} + \mathbf{n_0}\,\mathsf{sign}(\mathbf{n_0}\cdot\mathbf{R_1^1})\phi_1^1(\mathbf{r})\Omega^1\\
&\quad - \frac{\mathbf{n_1^1}}{s_{11}^1 - s_{12}^1}\left[R_2^1 - R_1^1 - s_{12}^1 \ln\left(\frac{R_2^1+R_1^1+||\mathbf{r'_3}-\mathbf{r'_2}||}{R_2^1+R_1^1-||\mathbf{r'_3}-\mathbf{r'_2}||}\right)\right]\\
&\quad - \mathbf{n_2^2}\ln\left(\frac{R_3^2+R_2^2+||\mathbf{r'_2}-\mathbf{r'_1}||}{R_3^2+R_2^2-||\mathbf{r'_2}-\mathbf{r'_1}||}\right) - \mathbf{n_1^2}\ln\left(\frac{R_2^2+R_1^2+||\mathbf{r'_4}-\mathbf{r'_1}||}{R_2^2+R_1^2+||\mathbf{r'_4}-\mathbf{r'_1}||}\right) + \mathbf{n_0}\mathsf{sign}(\mathbf{n_0}\cdot\mathbf{R_2^2})\Omega^2\\
&\quad - \frac{\mathbf{n_2^2} L_T(\mathbf{r},\mathbf{r'_4},\mathbf{r'_1},\mathbf{r'_2})}{(\mathbf{r'_4}-\mathbf{r'_1})\cdot\mathbf{n_2^2}} - \mathbf{n_0}\,\mathsf{sign}(\mathbf{n_0}\cdot\mathbf{R_1^2})\phi_1^2(\mathbf{r})\Omega^2\\
&\quad + \frac{\mathbf{n_1^2}}{s_{11}^2 - s_{12}^2}\left[R_2^2 - R_1^2 - s_{12}^2 \ln\left(\frac{R_2^2+R_1^2+||\mathbf{r'_1}-\mathbf{r'_4}||}{R_2^2+R_1^2-||\mathbf{r'_1}-\mathbf{r'_4}||}\right)\right].
\end{aligned}
$$

There are no problematic terms in the above expression. Without loss of generality, it can be concluded that when $\mathbf{r}, \bar{\mathbf{r}}_\mathbf{2}$ and $\bar{\mathbf{r}}_\mathbf{4}$ are colinear, $K_R$ can still be computed for all rectangular surfaces.