Data-driven methods for magnetic signatures

S. L. de Gijsel

Delft University of Technology

TU Delft

Data-driven methods for magnetic signatures

by

S. L. de Gijsel

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Tuesday July 20, 2021 at 14:00.

Student number:	4497678	
Project duration:	November 1, 2020 – July 20, 2021	
Thesis committee:	Prof. dr. ir. A.W. Heemink,	TU Delft
	Dr. N.V. Budko,	TU Delft
	Dr. ir. E.S.A.M. Lepelaars,	TU Delft / TNO
	R.G. Tan, MSc,	TNO
	Ir. A.R.P.J. Vijn,	TU Delft / TNO

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

The story of this thesis started already in the first year of my master Applied Mathematics. In the course Advanced Modeling in Science, I was immediately intrigued by the project *Magnetic Anomaly Detection using Autonomous Drones*. In that project our team developed models of drones, magnetic sources, and noise, and using these we did research on the optimal ways to find magnetic anomalies. I found the topics involved so interesting and exciting, that I decided to write my thesis on a similar subject. During the past nine months, I started to learn a lot more about magnetism, data-driven techniques, but also about defence. Understanding more about these worlds got me even more enthusiastic.

I am grateful to everyone who has helped me over the past months, gave me advice, or supported me in any other way. Of course, many thanks go to my supervisors Aad Vijn, Reinier Tan, and Eugene Lepelaars, for all their technical and personal guidance and support. Even though the circumstances did not allow regularly meeting in person, we made the best out of it and it was always fun and inspiring to meet online. I want to thank Arnold Heemink as well, for his valuable advice throughout the project. And last but not least, I would like to thank Max Suurland in particular for all the fruitful discussions, and also all my family and other friends who supported me during these months.

S. L. de Gijsel
Delft, July 2021

Abstract

A severe incident with the cargo ship MSC Zoe, which lost hundreds of containers near the Dutch coast, shows how important it can be to detect and localise hidden objects at sea quickly. Also in defence applications localisation of hidden objects in water is important. If the objects are made of magnetic materials, they disturb the Earth magnetic field. This disturbance is called the object's magnetic signature. An aerial drone that carries magnetic field sensors could in principle be able to locate magnetic objects autonomously. Another defence-related application of such a drone would be estimating magnetic signatures of ships under water, using measurements above water - signature translation. For both applications, algorithms need to be developed that process magnetic field data, and computes estimates. In this thesis, data-driven techniques are applied to the localisation problem and the magnetic signature translation problem.

For the localisation problem, an algorithm is proposed to localise a magnetic dipole using a limited number of noisy measurements from a sensor array forming a horizontal grid. The algorithm is based on the theory of compressed sensing, and exploits the sparseness of the magnetic dipole in the location domain. Beforehand, a basis consisting of magnetic dipole fields belonging to individual magnetic dipoles in an evenly spaced 3D grid is constructed. In the algorithm, a number of sensors is chosen which perform each a measurement of the three magnetic field components. The sensors can be chosen randomly from the sensor array, but also optimal placement using QR pivoting is considered. Using the pre-constructed basis and the obtained field measurements, a sparse representation in the location domain is computed using ℓ_1 optimisation. The optimisation is performed using a primal-dual path-following algorithm. Based on the resulting sparse representation, a classification is produced, which consists of estimates on its location and on its magnetic moment magnitude and orientation. The possibility of performing iterations is explored, where the basis and chosen sensors improve after every location estimate. Using results from simulations as well as experiments, the algorithm is shown to be effective in localising magnetic dipoles.

To solve the signature translation problem, two approaches are investigated. One algorithm was developed using the Gappy POD technique, and one using neural networks. Gappy POD is an adaptation of the proper orthogonal decomposition (POD), where signals are decomposed in orthogonal components. Using only a few measurements (a gappy measurement), a full signal can be reconstructed. This method is adapted to also be able to translate from a field above a ship to a field below a ship. Tikhonov regularisation is applied in the process to obtain better results. Second, different neural networks are created, trained using data above and below a ship. Linear and non-linear networks are compared, and standard loss functions are compared to physics-guided losses. Both the Gappy POD based algorithm and linear neural networks are shown to give good magnetic signature estimates. The Gappy POD method is greatly improved by applying Tikhonov regularisation. Results show to improve when more basis modes are considered, and when more sensors are used for measurements. Linear neural networks show the same results as Gappy POD using Tikhonov regularisation, indicating that some regularisation is done while training the neural network. Non-linear neural networks show no improvement over linear ones, and a physics-guided loss function is not needed for a resulting field that obeys known laws of magnetism.

Contents

Acronyms

DCT Discrete Cosine Transform

MAD Magnetic Anomaly Detection

MSE Mean Squared Error

NRMSE Normalised Root Mean Squared Error

POD Proper Orthogonal Decomposition

QCLP Quadratically Constrained Linear Programming

ReLU Rectified Linear Unit

RIP Restricted Isometry Property

RMSE Root Mean Squared Error

SDP Semidefinite Programming

SGD Stochastic Gradient Descent

SOCP Second Order Conic Programming

SVD Singular Value Decomposition

1 Introduction

1.1. Introduction and motivation

In 2019, 342 containers fell of the container ship MSC Zoe during rough weather conditions, close to the Dutch and German Frisian Islands. In total, the MSC Zoe lost 3 million kilograms of freight, in 6 different locations [36]. Some of the containers contained toxic chemicals, which could have a severe negative environmental impact if not found quickly. One of these was filled with the toxic substance benzoyl peroxide, and was only found after almost four months [26]. Another container with highly toxic lithium-ion batteries was never found. In total, 25 percent of lost freight was never recovered [36], and it took more than a year to clean all parts of the Dutch Wadden Sea [27]. This incident shows how important it can be to quickly localise hidden objects at sea.

Now envision an aerial drone, equipped with a magnetic sensor, that can autonomously locate these containers using its magnetic field measurements. With these drones, the container detection problem, and localisation problem, could efficiently be solved. Besides containers, there are more relevant detection problems that could be solved with such drones, for example related to defence. Hostile ships or objects need to be detected when getting into protected waters, or near a naval fleet. In these cases, detection and localisation is crucial for safety. Even though the applications are very different, the detection problem is similar.

There are several techniques used for detection of objects at sea, whether floating or under water. Some of these are acoustics (for instance sonar), radar, and of course using 'normal' vision or cameras. The technique that is central in this thesis is detection using magnetism.

Virtually all vessels, and also steel containers, are at least partly made of magnetic materials. The Earth creates an Earth magnetic field, and when a magnetic material is placed in this background field, it becomes a magnetic source itself. In this way, the Earth magnetic field is disturbed, and this disturbance (anomaly) can be detected. The disturbance in the magnetic field that a vessel (or other object) creates is called its magnetic signature. The overarching name for detection of magnetic sources from the air is Magnetic Anomaly Detection (MAD).

When the object of interest is magnetic, it can be useful to use MAD as detection technique. Unlike other signals, the magnetic field propagates freely through non-magnetic materials: this is called a strong penetrability. For instance, there will not be much difference between the magnetic field of a floating container and one that is slightly under water [19]. Other techniques like radar or cameras can easily fail in these situations. Another advantage of MAD in defence-related applications is that it is a passive technique [29]. The sensor or sensing process can in principle not be revealed by the target, in contrast to for instance sonar, where signals are sent actively in the detection process.

Apart from the detection problem described above, a drone equipped with magnetic field sensors can also help in another problem: magnetic signature estimation. Staying in the domain of defence, it can be important to know the magnetic signature of a ship. For instance, to make sure it is not detected. This is important in areas where naval mines form a threat. These mines can have different sensors on board, among which magnetic field sensors. The mines are set to explode if a ship is detected, or sometimes if a certain type of ship is detected.

On the one hand, for vessels that must not be detected, it is crucial that their magnetic field disturbance is small enough. Only then - disregarding many other requirements - can they safely enter an area that is potentially filled with naval mines. On the other hand, magnetic signatures are important in minesweeping operations - detonating mines on purpose to clear an area for an upcoming fleet. Usually, a so called minesweeper tows equipment that emulates the signatures of a naval vessel. The goal is to detonate mines that would be triggered by the real version of this ship. If this succeeds, the route is clear for the real vessel. In this case, it is important that the signatures of the real vessel and the emulation are similar [17, Chapter 3].

For these reasons, it can be beneficial to run a 'signature check' before a dangerous area is entered, or before a mission is started. This is where the aerial drone comes into play. At sea, an efficient way of performing such a check would be with one of these drones. When carrying one or more magnetic field sensors, it would be able to scan the area above a ship. Then, this signature above the ship needs to be translated into a signature below the ship. If this final signature is as expected, the mission can proceed as planned. And if the signature contains unexpected features, adjustments need to be done beforehand.

We covered two problems where magnetic field measurements conducted from an aerial drone can be used: (1) to detect magnetic objects at sea, and (2) to obtain a full magnetic signature around the ship - in particular below it. In the process of solving these problems, the field measurements need to be processed, since they will not be perfect. Measurements will typically be noisy, and field measurements can only be done along the drone's flight path. Then, in the detection problem, an estimated target location needs to be produced from the field measurements. And in the signature estimation problem, the measurements above a ship must be translated to a field below the ship. The goal of this thesis is to develop data-driven techniques to solve these problems.

1.2. Background of data-driven techniques

A standard approach to a physical problem like this is to analyse which physics laws are relevant. From these laws and rules, a physical model can be constructed. Then a simulation can be built using numerical approximations, or in the case of processing measurements, an inverse problem can be defined and solved. This whole approach is based on physics, in particular on behaviour that is already well-understood. But what if the laws of physics covering the problem are unknown, or the situation is too complex to model without oversimplification? In that case, another approach can be used: data-driven techniques.

Data-driven science focuses on the discovery of information just from observations, or data. It is not a new field in science: already in the 17th century, Johannes Kepler formulated his laws of planetary motion solely based on astronomical observations [38]. However, developments in the field are accelerating nowadays, with the rise of machine learning and statistical learning [7]. Using the large amounts of available computational power, data can be used to make new discoveries and to create new models or improve existing ones. In this thesis, the focus will be on three different data-driven techniques: compressed sensing, Gappy POD, and neural networks.

For magnetic source localisation, the method of compressed sensing is explored. This technique uses the concept of compression, but turns it around. The original idea of compression is that information can be stored using less storage space than originally, losing a negligible or at least very small amount of quality. Well-known examples are sound and image compression: with just a small loss of quality, huge amounts of storage space can be saved. In classical compression, only after performing the full measurement, the measurement data is encoded to be stored efficiently. The idea of compressed sensing is to never perform the full measurement in the first place, since it was apparently larger than needed to cover all needed information. Compressed sensing uses few - or sparse - measurements to estimate the encoded information as well as

possible. It turns out that this is possible since the final goal is the information in compressed form, and not the full data including all (unnecessary) details.

In the case of magnetic signature reconstruction, Gappy POD and neural networks are used. The proper orthogonal decomposition (POD) aims to find 'basis modes' in data. For example, audio data can be decomposed in sine waves, each with a separate frequency. In this case, the sine waves are the modes. In audio, this is quite straightforward, but in many situations, one has to search for the modes in a large set of data. A POD algorithm finds similarities between data entries. When the basis modes are found, the algorithm will try to fit these modes to measurements. In Gappy POD, the measurements are gappy, as the name suggests. Then it comes down to fitting the basis modes to the few measurements that are available. In the end, Gappy POD constructs a linear system to solve, hence non-linearities in the data cannot be found.

Neural networks are perhaps the most well-known recent innovation within data-driven science. The idea is simple: construct a network that is inspired by the functioning of the human brain. The network consists of so called neurons, that work approximately like neurons in a brain - each individual neuron is capable of receiving, modifying and then transmitting information to all connected neurons. Some input can be fed to the first neurons, and in the end, after possibly many layers of neurons, the last neurons produce an output.

This network is then trained to recognise input data and create proper output data. Some networks are trained using labelled data. For instance, input images are marked by humans as either dogs or cats, and based on this information, the network is trained to recognise dogs and cats. This is called supervised learning. A contrasting way of training neural networks is unsupervised learning. In this method, target information is not provided - the dog and cat images would not be labelled, for instance. It is the goal of the network to discover patterns on its own [7, Chapter 5].

In this thesis, the focus is on supervised learning. If an unseen magnetic field input is given to a well-trained network, it should be able to predict the desired output - a full magnetic field estimate below a ship. In contrast to Gappy POD, the total system can be non-linear due to the way neurons receive information from previous layers of neurons. In this way, the network could 'catch' non-linearities in the training data that would be disregarded by the linear Gappy POD.

1.3. Research goals and approach

As described in the previous sections, the goal of this research is to explore data-driven techniques for localisation using magnetic field measurements, and magnetic signature reconstruction. The main research question of this thesis summarises this, and consists of two parts:

How can data-driven techniques be used for

- *localisation of magnetic sources?*
- *reconstruction of the magnetic signatures of ships?*

Both questions are focused on achieving the goal in challenging situations, in particular when few and/or noisy measurements are available. To answer the two parts of the main question, the research is divided in two parts as well. Each contains a number of relevant steps towards the final goals:

Localisation:

1. Investigating the concept of compressed sensing;
2. Developing an algorithm to localise a magnetic dipole in a 3D space;
3. Exploring possibilities to choose optimal sensor locations when the number of measurements is restricted;
4. Analysing the algorithm performance using simulated data;
5. Validating the algorithm using real-world measurements.

Signature reconstruction:

1. Investigating the concepts of Gappy POD and neural networks;
2. Implementing a Gappy POD algorithm for signature reconstruction around a ship;
3. Designing a neural network for signature reconstruction around a ship;
4. Analysing and comparing the performance of the developed Gappy POD algorithm and neural network.

1.4. Outline

This report starts with two introductory chapters: Chapter 2 presents some background knowledge about magnetostatics, and Chapter 3 contains all relevant knowledge of linear algebra that is used throughout this thesis. The rest of the report is divided into two parts: Part I is about magnetic source localisation, and Part II about signature reconstruction. Each of them has a separate chapter with conclusions and recommendations.

Part I is divided into four chapters. An introduction to compressed sensing, including relevant mathematical background, is given in Chapter 4. The design and implementation of the compressed sensing algorithm for localisation is discussed in Chapter 5. Chapter 6 contains results and analysis of the implemented algorithm. This includes both results using simulated data and using real-world data. Finally, Chapter 7 describes the conclusions drawn from the results, and recommendations for future research.

Part II is divided into four chapters. Chapter 8 presents the problem of signature translation and the physical model used in the implementations in the following chapters. Chapter 9 discusses all relevant background information about Gappy POD, presents the implementation of a Gappy POD algorithm, and gives the results of an analysis. Chapter 10 has the same structure, but now for neural networks. At the end of this chapter, a comparison between the Gappy POD and neural network methods and implementations is made. Finally, Chapter 11 presents the conclusions of this thesis, and provides recommendations for future research.

2

Magnetostatics

All techniques in this research focus on magnetic field measurements. In particular, measurements of magnetic fields that do not change in time, or *static* fields. As an introduction, this chapter covers the basis of the theory of magnetostatics, mostly based on [16, Chapter 5].

2.1. Maxwell's equations and the vector potential

In the 19th century, James Clerk Maxwell combined, and improved, the set of four equations which are now together called Maxwell's equations. The set consists of Gauss's, Faraday's, and Ampère's law, and Gauss's law for magnetism. Together, these provide the overarching framework for electrodynamics. Each of the four equations describes a property of either the magnetic field $\mathbf{B}$ [T] or electric field $\mathbf{E}$ [NC^{-1}] or equivalently [Vm^{-1}], or a relation between both. Two of these describe the divergence of the fields, and the other two describe the curl. Maxwell's equations are given by [16, p. 326]:

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0}, \qquad \text{(Gauss's law)}$$

$$\nabla \cdot \mathbf{B} = 0, \qquad \text{(Gauss's law for magnetism)}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}, \qquad \text{(Faraday's law)}$$

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

Here, ρ [Cm^{-3}] denotes the total electric charge density, $\mathbf{J}$ [Am^{-2}] the total electric current density, ϵ_0 [Fm^{-1}] the permittivity of free space, and $\mu_0 \approx 1.257 \cdot 10^{-6}$ Hm^{-1} the permeability of free space [16, p. 216].

Considering only the magnetostatic field $\mathbf{B}$ now, the time derivatives vanish and the following remains:

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

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

2.1.1. Solving the magnetostatic equations

Eq. (2.1) gives rise to the use of a vector potential $\mathbf{A}$ in magnetostatics, allowed by Theorem 2.1:

$$\mathbf{B} = \nabla \times \mathbf{A}. \tag{2.3}$$

Theorem 2.1. *The following conditions for a vector field* $\mathbf{F}$ *are equivalent:*

(1) $\nabla \cdot \mathbf{F} = 0$ *everywhere.*
(2) $\mathbf{F}$ *is the curl of some vector function:* $\mathbf{F} = \nabla \times \mathbf{A}$.

Since the divergence of a curl is always zero, Eq. (2.1) is satisfied. Calculating the curl of $\mathbf{B}$, using Eq. (2.2), gives:

$$\nabla \times \mathbf{B} = \nabla \times (\nabla \times \mathbf{A}) = \nabla(\nabla \cdot \mathbf{A}) - \nabla^2 \mathbf{A} = \mu_0 \mathbf{J}. \tag{2.4}$$

To $\mathbf{A}$, it is possible to add any function whose curl is zero, and Eq. (2.3) will still hold. In particular, it is always possible to choose a vector potential $\mathbf{A}$ that is divergence-less:

$$\nabla \cdot \mathbf{A} = 0. \tag{2.5}$$

This equation is also called the gauge fixing condition which defines the *Coulomb gauge*. Combining Eq. (2.4) and Eq. (2.5) results in

$$\nabla^2 \mathbf{A} = -\mu_0 \mathbf{J}, \tag{2.6}$$

which is a system of Poisson equations with known solution [18, p. 180]

$$\mathbf{A}(\mathbf{r}) = \frac{\mu_0}{4\pi} \int \frac{\mathbf{J}(\mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||} d^3\mathbf{r}'. \tag{2.7}$$

Here, $\mathbf{r}$ is some point in space, and $\mathbf{r}'$ is the integration variable.

2.2. Biot-Savart

From Eq. (2.3) and Eq. (2.7), the Biot-Savart law for volume currents can be derived. This law gives the relation between the magnetic field $\mathbf{B}$ and volume current density $\mathbf{J}$. The magnetic field is equal to

$$\mathbf{B}(\mathbf{r}) = \nabla \times \mathbf{A}(\mathbf{r}) = \nabla \times \frac{\mu_0}{4\pi} \int \frac{\mathbf{J}(\mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||} d^3\mathbf{r}' = \frac{\mu_0}{4\pi} \int \nabla \times \left(\frac{\mathbf{J}(\mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||} \right) d^3\mathbf{r}', \tag{2.8}$$

where the last operation is allowed since the curl is calculated over the $\mathbf{r}$ coordinates, while the integral is taken over the $\mathbf{r}'$ coordinates. Using the product rule for some scalar function ϕ and vector field $\mathbf{F}$,

$$\nabla \times (\phi \mathbf{F}) = \nabla\phi \times \mathbf{F} + \phi \nabla \times \mathbf{F}, \tag{2.9}$$

the following is obtained:

$$\begin{aligned} \nabla \times \left(\frac{\mathbf{J}(\mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||} \right) &= \left(\frac{1}{||\mathbf{r} - \mathbf{r}'||} \right) \nabla \times \mathbf{J}(\mathbf{r}') + \nabla \left(\frac{1}{||\mathbf{r} - \mathbf{r}'||} \right) \times \mathbf{J}(\mathbf{r}') \\ &= 0 - \left(\frac{\mathbf{r} - \mathbf{r}'}{||\mathbf{r} - \mathbf{r}'||^3} \right) \times \mathbf{J}(\mathbf{r}') \\ &= \frac{\mathbf{J}(\mathbf{r}') \times (\mathbf{r} - \mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||^3}. \end{aligned} \tag{2.10}$$

Inserting this identity into Eq. (2.8) results in the Biot-Savart law for volume currents [25]:

$$\mathbf{B}(\mathbf{r}) = \frac{\mu_0}{4\pi} \int \frac{\mathbf{J}(\mathbf{r}') \times (\mathbf{r} - \mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||^3} d^3\mathbf{r}'. \tag{2.11}$$

2.3. Multipole expansion

A magnetic field originating from materials or objects can be described by a distribution of magnetic moments $\mathbf{m}$ [A m^2] with each its own orientation and magnitude. Far enough from the object, the field will tend to be that of a point magnetic dipole. A perfect magnetic dipole field can be thought of as the field of an infinitesimally small bar magnet or current loop.

The vector potential $\mathbf{A}$ can be approximated at a distance from the source by a multipole expansion, starting from the magnetic dipole (magnetic monopoles do not exist in nature). The expansion comes in the form of a power series in $\frac{1}{r}$, with r the distance to the source. The vector potential of a current loop can be written as [16, p. 243]

$$\mathbf{A}(\mathbf{r}) = \frac{\mu_0 I}{4\pi} \sum_{n=0}^{\infty} \frac{1}{r^{n+1}} \oint (r')^n P_n(\cos\alpha) d\mathbf{l}', \tag{2.12}$$

where I is the current flowing through the loop, r is the distance to the source, r' is the distance from the volume element to the source, and α is the angle between the vectors $\mathbf{r}$ and $\mathbf{r}'$. P_n are the Legendre polynomials, starting with $P_1(x) = x$, $P_2(x) = \frac{3x^2-1}{2}$, $P_3(x) = \frac{5x^3-3x}{2}$, and $d\mathbf{l}'$ is the current element considered.

Since no magnetic monopoles are present in nature, the monopole term ($n = 0$) vanishes [16, p. 243]. From the remaining terms in the expansion, the dipole is the dominant term at large distance. In these cases, the field originating from a source sufficiently far away can be approximated by the field of a magnetic dipole. Its vector potential is given by

$$\begin{aligned} \mathbf{A}_{\text{dip}}(\mathbf{r}) &= \frac{\mu_0 I}{4\pi r^2} \oint r' \cos\alpha \, d\mathbf{l}' \\ &= \frac{\mu_0 I}{4\pi r^3} \oint (\mathbf{r}\cdot\mathbf{r}') d\mathbf{l}' \\ &= \frac{\mu_0}{4\pi} \frac{\mathbf{m}\times\mathbf{r}}{r^3}, \end{aligned} \tag{2.13}$$

where $\mathbf{m}$ is the dipole moment of the current loop, defined by the vector area $\mathbf{a}$ of the loop, with orientation determined by the right hand rule:

$$\mathbf{m} := I\mathbf{a}.$$

2.4. Magnetic field of a point dipole

From this vector potential, it is possible to derive the magnetic field of a point dipole. To this end, Eq. (2.3) is used:

$$\begin{aligned} \mathbf{B}_{\text{dip}}(\mathbf{r}) &= \nabla \times \mathbf{A}_{\text{dip}}(\mathbf{r}) \\ &= \frac{\mu_0}{4\pi} \left(\nabla \times \left(\mathbf{m} \times \frac{\mathbf{r}}{r^3} \right) \right). \end{aligned} \tag{2.14}$$

Now the following identity is used [32, p. 123]:

$$\nabla \times (\mathbf{u}\times\mathbf{v}) = \mathbf{u}(\nabla\cdot\mathbf{v}) - \mathbf{v}(\nabla\cdot\mathbf{u}) + (\mathbf{v}\cdot\nabla)\mathbf{u} - (\mathbf{u}\cdot\nabla)\mathbf{v}. \tag{2.15}$$

Combining with $\nabla\cdot\mathbf{m} = 0$ and $\nabla\mathbf{m} = \mathbf{0}$ (the Jacobian of $\mathbf{m}$ only has zero entries) gives:

$$\nabla \times \left(\mathbf{m} \times \frac{\mathbf{r}}{r^3} \right) = \left(\nabla \cdot \frac{\mathbf{r}}{r^3} \right) \mathbf{m} - (\mathbf{m}\cdot\nabla) \frac{\mathbf{r}}{r^3}. \tag{2.16}$$

The first term in Eq. (2.16) vanishes:

$$\begin{aligned} \nabla \cdot \frac{\mathbf{r}}{r^3} &= \frac{1}{r^3} \nabla\cdot\mathbf{r} + \left(\nabla \frac{1}{r^3} \right) \cdot \mathbf{r} \\ &= \frac{3}{r^3} - \frac{3\mathbf{r}}{r^5} \cdot \mathbf{r} \\ &= 0. \end{aligned} \tag{2.17}$$

For the second term of Eq. (2.16), the Jacobian of $\frac{\mathbf{r}}{r^3}$ is needed (here notation is $\mathbf{r} = x\hat{\mathbf{i}} + y\hat{\mathbf{j}} + z\hat{\mathbf{k}}$ and $r^2 = x^2 + y^2 + z^2$):

$$\nabla \frac{\mathbf{r}}{r^3} = \frac{1}{r^3} \begin{bmatrix} 1 - \frac{3x^2}{r^2} & -\frac{3xy}{r^2} & -\frac{3xz}{r^2} \\ -\frac{3xy}{r^2} & 1 - \frac{3y^2}{r^2} & -\frac{3yz}{r^2} \\ -\frac{3xz}{r^2} & -\frac{3yz}{r^2} & 1 - \frac{3z^2}{r^2} \end{bmatrix}. \tag{2.18}$$

Then, the full second term of Eq. (2.16) becomes:

$$\begin{aligned} (\mathbf{m} \cdot \nabla) \frac{\mathbf{r}}{r^3} &= \frac{1}{r^3} \begin{bmatrix} m_x - \frac{3}{r^2}(m_x x^2 + m_y xy + m_z xz) \\ m_y - \frac{3}{r^2}(m_x xy + m_y y^2 + m_z yz) \\ m_z - \frac{3}{r^2}(m_x xz + m_y yz + m_z z^2) \end{bmatrix} = \frac{1}{r^3} \begin{bmatrix} m_x - \frac{3}{r^2} x(m_x x + m_y y + m_z z) \\ m_y - \frac{3}{r^2} y(m_x x + m_y y + m_z z) \\ m_z - \frac{3}{r^2} z(m_x x + m_y y + m_z z) \end{bmatrix} \\ &= \frac{\mathbf{m}}{r^3} - \frac{3\mathbf{r}(\mathbf{m} \cdot \mathbf{r})}{r^5}. \end{aligned} \tag{2.19}$$

Combining the previous equations, and letting $\hat{\mathbf{r}} = \frac{\mathbf{r}}{r}$ gives the magnetic field of a point dipole:

$$\mathbf{B}_{\text{dip}}(\mathbf{r}) = \frac{\mu_0}{4\pi} \left(\frac{3\hat{\mathbf{r}}(\mathbf{m} \cdot \hat{\mathbf{r}}) - \mathbf{m}}{r^3} \right). \tag{2.20}$$

2.5. Magnetisation

Magnetic fields are related to currents flowing in materials, see Eq. (2.2). For a wire or coil through which current is flowing, it is obvious that a current (**J**) and therefore a magnetic field (**B**) is present. For materials, for example permanent magnets, this is not so obvious. However, in every magnetic material, tiny bits of currents can be found in the form of spinning electrons in atoms. These spins can be thought of as small dipoles. Normally, the atoms in a material are not aligned and hence the dipoles point in random directions and cancel out. Under influence of an external magnetic field, the dipoles can be realigned, causing the material to be magnetised. Some materials magnetise in the same direction as the applied field (paramagnetism), and some in the opposite direction (diamagnetism). However, the strongest form of magnetism - and the one that you encounter in daily life - is ferromagnetism. In ferromagnetic material, the electron spins are aligned. When an external field is applied, the spins align with this field. Ferromagnetic material can retain some of its magnetism after time.

Magnetisation of a material is indicated by the symbol $\mathbf{M}$ [$\mathrm{A\,m^{-1}}$], and defined as the magnetic dipole moment per unit volume. Magnetisation is the result of the alignment of dipole moments $\mathbf{m}$ inside the material. Bound currents in the material can be related to the magnetisation of an object by

$$\mathbf{J}_b = \nabla \times \mathbf{M}. \tag{2.21}$$

Apart from bound currents in the material $\mathbf{J}_b$, there can be other currents flowing through the material. The latter are called free currents and are denoted by $\mathbf{J}_f$. Free currents are, in fact, the currents that can be controlled directly in an experiment. Bound currents, on the other hand, result from magnetisation of material, and cannot be controlled separately. The total current present in the material is then

$$\mathbf{J} = \mathbf{J}_b + \mathbf{J}_f. \tag{2.22}$$

It is possible to rewrite Ampère's law in terms of only free current, which is useful since these currents are controllable in experiments. Starting from Eq. (2.2), this results in

$$\frac{1}{\mu_0}(\nabla \times \mathbf{B}) = \mathbf{J} = \mathbf{J}_f + \mathbf{J}_b = \mathbf{J}_f + (\nabla \times \mathbf{M}), \tag{2.23}$$

and from there, defining the auxiliary field $\mathbf{H}$ [$\mathrm{A\,m^{-1}}$] as

$$\mathbf{H} := \frac{1}{\mu_0}\mathbf{B} - \mathbf{M}, \tag{2.24}$$

a new form of Ampère's law in terms of the auxiliary field and free current arises:

$$\nabla \times \mathbf{H} = \mathbf{J}_f. \tag{2.25}$$

Note that outside material, $\mathbf{M} = \mathbf{0}$, and therefore

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

In many magnetic materials, both paramagnetic and diamagnetic, the magnetisation is linearly dependent on the applied magnetic field. For these linear media, the magnetic susceptibility χ_m is defined as the (dimensionless) proportionality constant:

$$\mathbf{M} = \chi_m \mathbf{H}. \tag{2.26}$$

From Eq. (2.24) it then follows that there is a constant μ such that

$$\mathbf{B} = \mu \mathbf{H}. \tag{2.27}$$

μ [$\mathrm{H\,m^{-1}}$] is called the permeability of the material.

If there are no free currents present in a region, Eq. (2.25) indicates that the curl of the **H**-field is zero, hence it is possible to define a magnetostatic scalar potential Φ, such that

$$\mathbf{H} = -\nabla \Phi. \tag{2.28}$$

In linear media, Eq. (2.1) then produces the Laplace equation

$$\nabla^2 \Phi = 0. \tag{2.29}$$

Not all magnetic materials are linear media: ferromagnets have the ability to maintain a non-zero magnetisation even when no external field is applied. Examples of ferromagnetic materials include iron, cobalt, nickel and their alloys. Within ferromagnetic material, many small domains exist. A picture of such domains is shown in Fig. 2.1. Inside one domain, all dipoles formed by unpaired electron spins are aligned. In case the material is not magnetised, these individual domains cancel out, even though the dipoles within domains are aligned. When applying an external magnetic field to a ferromagnetic substance, most dipoles will not be affected by the torque created by the external magnetic field, with the help of surrounding dipoles in parallel orientation. At the borders between different domains, however, the torque originating from the applied magnetic field will cause some changes. The torque helps dipoles in the less similarly oriented domain to 'join' the domain that is oriented more parallel to the external field. In other words, borders between domains move. This effect causes a net magnetisation of the material in the same direction as the applied field.

A part of the magnetisation remains after the external field is switched off: not all dipoles change back to their original orientation. What is left can be called a permanent magnet. When applying an increasing external field which now oriented in exactly opposite direction of the previous field, at some point enough dipoles will have switched to reach zero magnetisation again. Increasing the field even more, the ferromagnetic material will eventually have a permanent magnetisation in opposite direction as before. This process, or path, is called a hysteresis loop. An example of such a loop or curve is shown in Fig. 2.2.

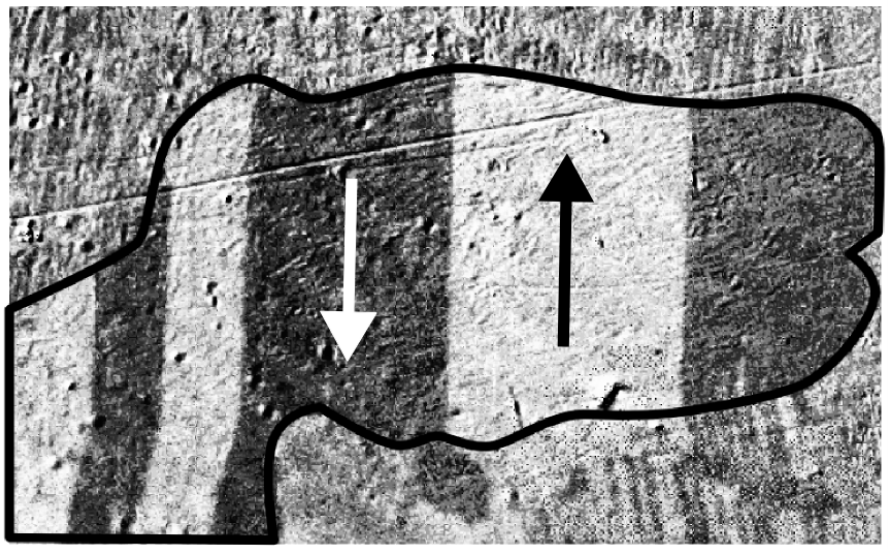

Figure 2.1: Magnetic domains in a grain of electrical steel (outlined). The orientation of the domains is indicated with arrows [39].

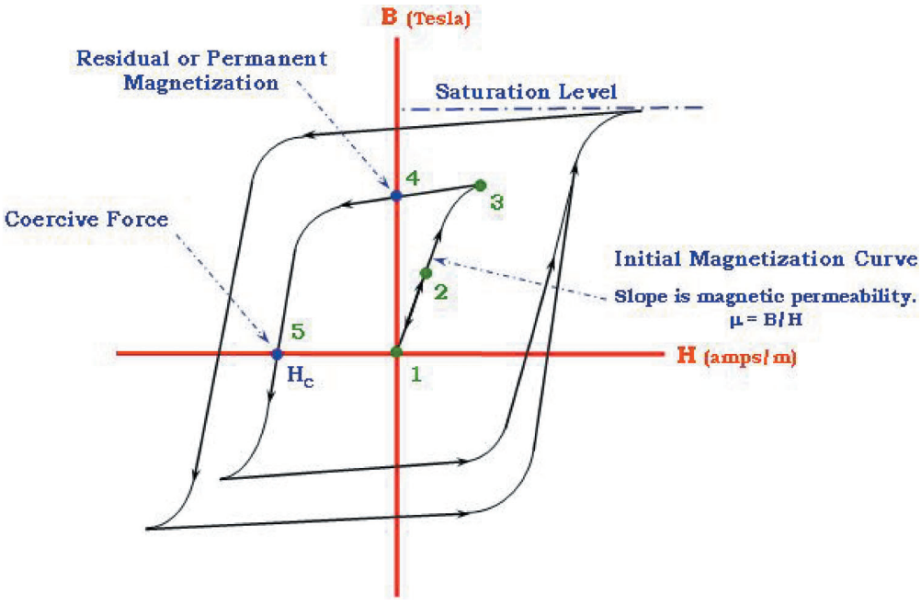

Figure 2.2: A hysteresis curve for ferromagnetic material [17, p. 9].

3

Prerequisites from linear algebra

Throughout this thesis, many techniques, definitions, and methods from linear algebra are used. This chapter presents an overview of the relevant topics. These are divided into the following: norms, general definitions, the singular value decomposition, the QR decomposition, and the condition number.

3.1. Relevant definitions in linear algebra

Definitions from the field of linear algebra that are used throughout this thesis are given below.

- Two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{C}^n$ are *orthogonal* if $\langle \mathbf{x}, \mathbf{y} \rangle = 0$, where $\langle \mathbf{x}, \mathbf{y} \rangle := \sum_{i=1}^{n} x_i \overline{y_i}$ is the standard inner product.
- Two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{C}^n$ are *orthonormal* if they are orthogonal and $\|\mathbf{x}\| = \|\mathbf{y}\| = 1$. Refer to Section 3.2.1 for a description of norms.
- The *projection* of a vector $\mathbf{y} \in \mathbb{C}^n$ onto a vector $\mathbf{x} \in \mathbb{C}^n$ is defined by $\text{proj}_{\mathbf{x}}(\mathbf{y}) = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\langle \mathbf{x}, \mathbf{x} \rangle} \mathbf{x}$, with $\langle \mathbf{x}, \mathbf{y} \rangle$ the inner product between vectors $\mathbf{x}$ and $\mathbf{y}$.
- A matrix $\mathbf{U}$ is *unitary* if $\mathbf{U}\mathbf{U}^* = \mathbf{U}^*\mathbf{U} = \mathbf{I}$.
- A matrix $\mathbf{A}$ is *upper triangular* if $a_{i,j} = 0$ for all $i > j$.
- The *linear span* of a set of vectors S is the set of all linear combinations of the vectors in S.
- The *rank* of a matrix is the dimension of the vector space spanned by its columns (or rows). A matrix is said to be of *full column rank* if all its columns are linearly independent.

3.2. Norms

To describe the size of the elements of vectors or matrices, one uses the concept of norms. An introduction to norms for both vectors and matrices is given in this section.

3.2.1. Vector norms

Consider an n-dimensional vector $\mathbf{x}$, which is an element of the *vector space* $\mathbb{R}^n$. On a vector space, a norm can be introduced, which will be able to determine the size of all vectors within the space. For the considered vector $\mathbf{x} \in \mathbb{R}^n$, the following norms are defined:

$$\|\mathbf{x}\|_1 := \sum_{i=1}^{n} |x_i| \qquad (\ell_1 \text{ norm})$$

$$\|\mathbf{x}\|_2 := \sqrt{\sum_{i=1}^{n} x_i^2} \qquad (\ell_2 \text{ norm})$$

The ℓ_0 "norm", which is sometimes called a pseudo-norm, is defined to be the number of non-zero elements in a vector, written more mathematically as

$$\|\mathbf{x}\|_0 := \#\{x_i : x_i \neq 0\}. \qquad (\ell_0 \text{ "norm"})$$

Note that this is not a norm, since it does not satisfy the scaling property: $\|\alpha\mathbf{x}\|_0$ is in general not equal to $|\alpha|\,\|\mathbf{x}\|_0$ as required.

3.2.2. Matrix norms

For some $m \times n$ matrix $\mathbf{A}$, the Frobenius norm is defined as:

$$\|\mathbf{A}\|_F := \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{i,j}|^2}. \qquad \text{(Frobenius norm)}$$

3.3. Singular value decomposition (SVD)

A very useful concept in linear algebra, especially in the field of compressed sensing, is the singular value decomposition (SVD). The SVD allows for creating lower-dimensional approximations to high-dimensional systems. This section is based on [7, Chapter 1].

Suppose that some measurements have been done, and each measurement is represented as a column vector $\mathbf{x}_i \in \mathbb{C}^n$. For instance, this vector could contain pixels from an image, or magnetic field measurements. After doing m such measurements, all data is gathered in a matrix $\mathbf{X} \in \mathbb{C}^{n\times m}$:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_m \end{bmatrix}. \tag{3.1}$$

Now, the SVD for any matrix $\mathbf{X} \in \mathbb{C}^{n\times m}$ is the following unique decomposition:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*, \tag{3.2}$$

with $\mathbf{U} \in \mathbb{C}^{n\times n}$ and $\mathbf{V} \in \mathbb{C}^{m\times m}$ unitary matrices with orthonormal columns, and $\mathbf{\Sigma} \in \mathbb{R}^{n\times m}$ a matrix with non-negative entries on the diagonal, in descending order, and zeros everywhere else. Note that the descending order of the elements in $\mathbf{\Sigma}$ causes the uniqueness of this decomposition. These elements are called *singular values*. The number of non-zero singular values is equal to the rank of $\mathbf{X}$. The conjugate transpose * can be replaced by a regular transpose ^{T} if the transposed matrix is real-valued.

A special and very useful property of the SVD is that a matrix $\tilde{\mathbf{X}}$ with reduced rank r can be created from it, providing an optimal approximation to $\mathbf{X}$. This is done by keeping the leading r singular values and columns of $\mathbf{U}$ and $\mathbf{V}$. The following theorem, proved by Eckart and Young in [13], as formulated in [7, p. 7-8], states this for the Frobenius norm (see Section 3.2.2):

Theorem 3.1. *The optimal rank-r approximation to $\mathbf{X}$, in a least-squares sense, is given by the rank-r SVD truncation $\tilde{\mathbf{X}}$:*

$$\underset{\tilde{\mathbf{X}},\, s.t.\, \mathrm{rank}(\tilde{\mathbf{X}})=r}{\arg\min} \left\|\mathbf{X}-\tilde{\mathbf{X}}\right\|_F = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^*. \tag{3.3}$$

Here, $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$ denote the first r leading columns of $\mathbf{U}$ and $\mathbf{V}$, and $\tilde{\mathbf{\Sigma}}$ contains the leading $r \times r$ sub-block of $\mathbf{\Sigma}$.

Using this theorem, it now becomes clear why the SVD helps creating lower-dimensional approximations to high-dimensional systems, as said at the beginning of this section. In fact, once the SVD has been computed, this is only a matter of deleting part of the matrices.

3.4. QR decomposition

Any matrix $\mathbf{A} \in \mathbb{C}^{m\times n}$, with rank($\mathbf{A}$) = n, can be factorised as

$$\mathbf{A} = \mathbf{QR}, \tag{3.4}$$

where $\mathbf{Q} \in \mathbb{C}^{m\times m}$ is unitary and $\mathbf{R} \in \mathbb{C}^{m\times n}$ is upper triangular with positive diagonal elements [6, p.254]. This is called a QR decomposition, or QR factorisation. In this thesis, we will only look at matrices $\mathbf{A}$ of full column rank. For these matrices, it holds that col($\mathbf{Q}$) = col($\mathbf{A}$) and since $\mathbf{Q}$ is unitary the columns of $\mathbf{Q}$ form an orthonormal basis of the column space of $\mathbf{A}$. A well-known way to compute the QR decomposition is the Gram-Schmidt process.

3.4.1. Gram-Schmidt process

The Gram-Schmidt process applied to a matrix $\mathbf{A}$ with n columns returns a QR decomposition. The idea of the process is to find an orthonormal basis for the columns of $\mathbf{A}$. The first vector of an orthogonal basis, $\mathbf{u}_1$, can be the first column $\mathbf{a}_1$:

$$\mathbf{u}_1 = \mathbf{a}_1. \tag{3.5}$$

For the second vector, the column $\mathbf{a}_2$ is used, but it needs to be modified to be orthogonal to the first. This is done by subtracting from $\mathbf{a}_2$ the projection of $\mathbf{a}_2$ onto $\mathbf{u}_1$. This projection can be seen as the part that makes $\mathbf{u}_1$ and $\mathbf{a}_2$ not orthogonal. Hence,

$$\mathbf{u}_2 = \mathbf{a}_2 - \text{proj}_{\mathbf{u}_1}(\mathbf{a}_2) \tag{3.6}$$

is orthogonal to $\mathbf{u}_1$. For the next column $\mathbf{a}_3$, its projection onto both $\mathbf{u}_1$ and $\mathbf{u}_2$ needs to be subtracted, and so on:

$$\mathbf{u}_k = \mathbf{a}_k - \sum_{j=1}^{k-1} \text{proj}_{\mathbf{u}_j}(\mathbf{a}_k). \tag{3.7}$$

Finally, to make the basis orthonormal, every column $\mathbf{u}_k$ must be normalised:

$$\mathbf{e}_k = \frac{\mathbf{u}_k}{\|\mathbf{u}_k\|}, \qquad k = 1, \ldots, n. \tag{3.8}$$

The vectors $\mathbf{e}_k$ form the matrix $\mathbf{Q}$ in the QR decomposition, and the matrix $\mathbf{R}$ is formed by the inner products of the columns $\mathbf{a}_j$ and the vectors $\mathbf{e}_k$:

$$\mathbf{Q} = [\mathbf{e}_1 \quad \mathbf{e}_2 \quad \cdots \quad \mathbf{e}_n]; \tag{3.9}$$

$$\mathbf{R} = \begin{bmatrix} \langle \mathbf{e_1}, \mathbf{a_1} \rangle & \langle \mathbf{e_1}, \mathbf{a_2} \rangle & \langle \mathbf{e_1}, \mathbf{a_3} \rangle & \cdots \\ 0 & \langle \mathbf{e_2}, \mathbf{a_2} \rangle & \langle \mathbf{e_2}, \mathbf{a_3} \rangle & \cdots \\ 0 & 0 & \langle \mathbf{e_3}, \mathbf{a_3} \rangle & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{3.10}$$

3.4.2. Column pivoting

A variation to this technique is the so-called QR decomposition with column pivoting. The focus here will be on the inclusion of column pivoting in the Gram-Schmidt process. This method is used if $\mathbf{A}$ is not of full column rank, and to improve numerical stability of the Gram-Schmidt process. It decomposes the $m \times n$ matrix $\mathbf{A}$ such that

$$\mathbf{AP} = \mathbf{QR}, \tag{3.11}$$

where $\mathbf{P} \in \mathbb{R}^{n\times n}$ is a permutation matrix. Such a matrix consists of rows that contain one element equal to 1 and have their other elements equal to zero. There can be no duplicate rows. Multiplying by such a matrix results in the same matrix, but with its columns switched up [7, p. 111-112].

If column pivoting is incorporated into the Gram-Schmidt process, the columns are not just iterated over from left to right, but in each iteration the largest column is chosen. Such a method where in every iteration a best action is chosen is called a *greedy* approach. Later in this thesis, the concept of compressed sensing is

introduced, where a limited number of measurements are used to reconstruct signals. There, column pivoting within the QR decomposition algorithm will be exploited to obtain optimal measurements. This will be demonstrated in Section 5.3.2.

In the Gram-Schmidt process, $\mathbf{P}$ is constructed throughout the iterations. The matrix $\mathbf{P}$ is initialised as the identity matrix: $\mathbf{P}_{(0)} = \mathbf{I}_n$. Instead of iterating over the columns $\mathbf{a}_j$ in order of appearance in $\mathbf{A}$, the next column in step k is chosen as

$$\mathbf{a}_p = \max_{j \geq k} \left\| \mathbf{a}_j \right\|_2 . \tag{3.12}$$

In other words: the largest remaining column of $\mathbf{A}$ is chosen next. This process is called column pivoting. If $p \neq k$ then columns p and k of $\mathbf{A}$ are swapped, and this is described by a change in the permutation matrix $\mathbf{P}$. Columns p and k are swapped, for example in the first iteration ($k = 1$):

$$\mathbf{P}_{(1)} = \begin{bmatrix} \mathbf{e}_p & \mathbf{e}_2 & \dots & \mathbf{e}_{p-1} & \mathbf{e}_1 & \mathbf{e}_{p+1} & \dots & \mathbf{e}_n \end{bmatrix}, \tag{3.13}$$

where $\mathbf{e}_k$ here denotes the unity vector with a 1 at position k [6, p. 292]. The consequence of this scheme is that the diagonal elements of $\mathbf{R}$ will be non-increasing:

$$r_{kk} \geq r_{jj}, \quad k > j. \tag{3.14}$$

In fact, the elements in $\mathbf{R}$ satisfy the following inequalities [5]:

$$r_{kk}^2 \geq \sum_{i=k}^{j} r_{ij}^2, \quad j = k+1, \dots, n. \tag{3.15}$$

An interesting property of QR algorithms with column pivoting is that they are *rank-revealing*. If $r_{kk} = 0$, then $r_{ij} = 0$ for $i, j \geq k$. The rank of $\mathbf{A}$ is $k-1$ if r_{kk} is the first diagonal element of $\mathbf{R}$ equal to zero. However, it must be noted that the algorithm can fail in identifying nearly rank-deficient matrices [5, p. 105].

Column pivoting is not restricted to the Gram-Schmidt process alone. Another way to compute a QR decomposition is using Householder transformations. In this algorithm - and others - the idea is the same: choose a next column such that their norm is largest. An excellent description of the Householder method and the modification of column pivoting to it is given in [10].

3.5. Condition number

Suppose we have a linear system of equations

$$\mathbf{y} = \mathbf{A}\mathbf{x}, \tag{3.16}$$

and the goal is to approximate $\mathbf{x}$ from some noisy measurement $\mathbf{y}$. How sensitive this approximation is to errors in the input $\mathbf{y}$ is indicated by the condition number of $\mathbf{A}$, defined by

$$\kappa(\mathbf{A}) = \left\| \mathbf{A}^{-1} \right\| \left\| \mathbf{A} \right\| . \tag{3.17}$$

In the ℓ_2 norm, this condition number is given by

$$\kappa(\mathbf{A}) = \frac{\sigma_{\max}(\mathbf{A})}{\sigma_{\min}(\mathbf{A})}, \tag{3.18}$$

where $\sigma_{\max}(\mathbf{A})$ is the maximum singular value of $\mathbf{A}$, and $\sigma_{\min}(\mathbf{A})$ the minimum singular value. To ensure a small error in the approximation of $\mathbf{x}$, the condition number $\kappa(\mathbf{A})$ must be small. Refer to Section 3.3 for an explanation of singular values.

I

Magnetic dipole localisation

4

Compressed sensing

Data compression techniques have been around for decades, and can be found in a wide range of applications, from audio to video and from web traffic to genetic sequences. The key concept is similar in all of these areas: reducing the size of data (mostly measured in number of bits for digital data) by encoding it. The trick is to find a balance between the quality of the encoded data and its size.

Typically, compression is applied after measuring data. For example, after a digital image is captured, the image data is encoded in order to store it efficiently. The compressed image can be many times smaller than the original. The original image is made up of a large amount of measurements, for example the pixel values in an image. The smaller the size of the compressed image, the less detail from the original image is preserved.

Compressed sensing turns the idea of compression around completely: if the goal is to have a small, compressed image with only marginal loss of quality, why not perform less measurements in the first place? After all, the small details acquired by the many measurements would get lost in the compression process anyway.

Compressed sensing, also referred to as compressive sensing, is the technique of recovering a signal using a very limited number of measurements. Note that such a signal can also be an image, for example in 2D. With this method, fewer measurements are needed than prescribed by classical information theory, such as the famous Shannon-Nyquist theorem [31]. For this technique to work, the measured signal must be sparse in some basis. The sparseness property determines the level at which a signal can be compressed, without the loss of information. This concept will be further explained in the upcoming sections - it is the same property that makes signal compressible.

The mathematical background of compressed sensing was developed in the 1980s, with new optimisation techniques to recover sparse signals. In recent years, the field has been growing quickly, thanks to the development of efficient algorithms, growing computing power, and the establishment of solid mathematical foundations. Today, compressed sensing techniques are applied in many important applications, ranging from imaging techniques in cameras, MRI and seismology to RADAR and communication networks [30].

This chapter contains a short background of data compression, including an example based on a sound signal. This is followed by a discussion of compressed sensing, consisting of an introduction to the topic, and an explanation of the optimisation principles that form the cornerstones of compressed sensing.

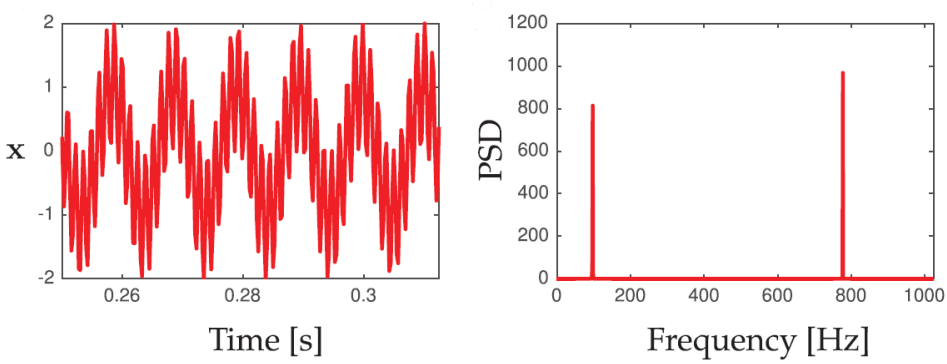

Figure 4.1: Signal (left) and power spectral density (right) of a two-tone sounds signal $x(t) = c_1 \cos(2\pi \cdot 97t) + c_2 \cos(2\pi \cdot 777t)$ [7, p. 94].

4.1. Compression

Compression in general makes use of the sparsity of a signal. If a signal can be described well by a limited number of modes in some domain, only these modes have to be stored and one can save storage space. Such a signal is called sparse. Think of sound, that can be described by a collection of frequencies, as illustrated in Fig. 4.1. The left plot shows the signal in time, and the right plot shows the discrete cosine transform (DCT) of this signal.

For example, the JPEG image compression method and mp3 audio compression technique make use of the DCT. After transformation, the signal can be heavily truncated - and hence less information needs to be stored. There is no or small quality loss detectable after transforming back using the inverse DCT. There are many variants of the DCT, each slightly altered, in which x_0 to x_{N-1} are transformed to X_0 to X_{N-1}. The basic idea of all of them is to express a signal as the sum of cosine waves of various frequencies. The most commonly used of these is DCT-II, given by [1]

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \qquad k = 0,\ldots,N-1. \tag{4.1}$$

and inverse

$$x_n = \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} X_k \frac{1}{\sqrt{1+\delta_{k1}}} \cos\left[\frac{\pi}{2N}(2n-1)(k-1)\right] \qquad n = 0,\ldots,N-1. \tag{4.2}$$

Coefficients x_n represent the original signal, and X_k are the basis coefficients. δ_{ij} denotes the Kronecker delta. Eq. (4.2) is in fact a summation with basis coefficients s_k and basis functions $\psi(k,n)$:

$$x_n = \sum_{k=0}^{N-1} s_n \psi(k,n) \qquad n = 0,\ldots,N-1, \tag{4.3}$$

$$s_k = X_k, \tag{4.4}$$

$$\psi(k,n) = \sqrt{\frac{2}{N}} \frac{1}{\sqrt{1+\delta_{k1}}} \cos\left[\frac{\pi}{2N}(2n-1)(k-1)\right]. \tag{4.5}$$

And finally, this is nothing more than a matrix-vector product

$$\mathbf{x} = \mathbf{\Psi}\mathbf{s}, \tag{4.6}$$

where $\mathbf{x}$ and $\mathbf{s}$ contain elements x_n and s_k, respectively, and $\mathbf{\Psi}$ has elements $\Psi_{kn} = \psi(k,n)$. In other words, the columns of $\mathbf{\Psi}$ contain the basis modes - in this case cosine waves - and the elements of $\mathbf{s}$ describe the magnitude of these modes in the original signal $\mathbf{x}$.

Many signals are sparse in Fourier or wavelet bases. These are called *universal bases*, since they can be applied to a lot of different problems. For some signals, one can find a specific basis that will only work for that type

of signal. Then the basis is called a *tailored basis*. A main advantage of considering a signal in one of these domains is that it provides useful information. For instance, for sound signals, a collection of frequencies and their magnitudes is obtained, just like in Fig. 4.1. Based on this information, one could start filtering noise or unwanted frequencies, or even just perform a meaningful analysis of the signal. Besides, if a time-varying signal is periodic, one needs a lot less storage space to save exactly the same information. Instead of the full signal in time, just a starting condition and the magnitudes of all frequency (or different basis) components is needed to exactly reconstruct the full signal in time. On the other hand, it requires some processing power to convert between original signals and their basis representations.

4.2. Sparse representation

Just as derived in the previous section for the DCT, in general it is possible for a sparse signal to write

$$\mathbf{x} = \mathbf{\Psi s}, \tag{4.7}$$

where $\mathbf{x} \in \mathbb{R}^m$ is the signal, $\mathbf{\Psi} \in \mathbb{R}^{m \times n}$ is some basis used for transformation and $\mathbf{s} \in \mathbb{R}^n$ is a sparse vector, containing only a small number of non-zero elements. $\mathbf{\Psi}$ is called a transform basis: it transforms a signal from the domain in which the signal is sparse to the original domain. In the example of sound, $\mathbf{x}$ would be the sound signal, $\mathbf{\Psi}$ could contain cosine basis functions with different frequencies, and $\mathbf{s}$ would be the vector indicating which cosine basis functions are present and with what amplitude.

Many of the elements in $\mathbf{s}$ for a real measurement are nearly zero - the non-zero part is often just noise. Setting these components to exactly zero saves a lot of space: only the few non-zero components that are left need to be stored in order to reconstruct the signal. In the case of the sound signal in Fig. 4.1, just 2 components need to be stored: one at 97 Hz and one at 777 Hz. The loss of quality after transforming back is small or zero.

4.3. Compressed sensing

While compression needs highly detailed measurements $\mathbf{x}$, which can then be truncated, compressed sensing uses the sparsity of a signal (see Section 4.2) *before* measuring to require relatively few measurements. Specifically, only a part of $\mathbf{x}$ needs to be measured. These measurements are transformed to the sparse domain, where the full sparse representation $\mathbf{s}$ is estimated. Transforming the latter back to the original domain, an estimate of the original signal $\mathbf{x}$ can be produced.

First, what is meant by a measurement? Consider a vector $\mathbf{x}$ as in Eq. (4.7). This vector could represent the pixels of an image, a sound signal, or any other physical *state* of a system. A measurement $\mathbf{y}$ of this state reveals part of the components of $\mathbf{x}$, for example a few pixels of the image. The goal is to reconstruct the original image in $\mathbf{x}$ using the measurement. Performing p measurements can be described by a multiplication of $\mathbf{x}$ by a *measurement matrix* $\mathbf{C} \in \mathbb{R}^{p \times m}$ containing rows with zeroes and one value of 1, resulting in a measurement vector $\mathbf{y} \in \mathbb{R}^p$:

$$\mathbf{y} = \mathbf{Cx}. \tag{4.8}$$

Essentially, this matrix $\mathbf{C}$ selects some entries from $\mathbf{x}$ that will end up in the measurement vector $\mathbf{y}$.

The system from Eq. (4.7) can then be written as

$$\mathbf{y} = \mathbf{C\Psi s} = \mathbf{\Theta s}, \tag{4.9}$$

with $\mathbf{\Theta} \in \mathbb{R}^{p \times n}$. Now, $\mathbf{C}$ selects certain rows from $\mathbf{\Psi}$ that end up in $\mathbf{\Theta}$. Hence, $\mathbf{\Theta}$ transforms a signal from the sparse domain to a measurement in the original domain. An illustration of these matrices and vectors is shown in Fig. 4.2 [7, p. 89].

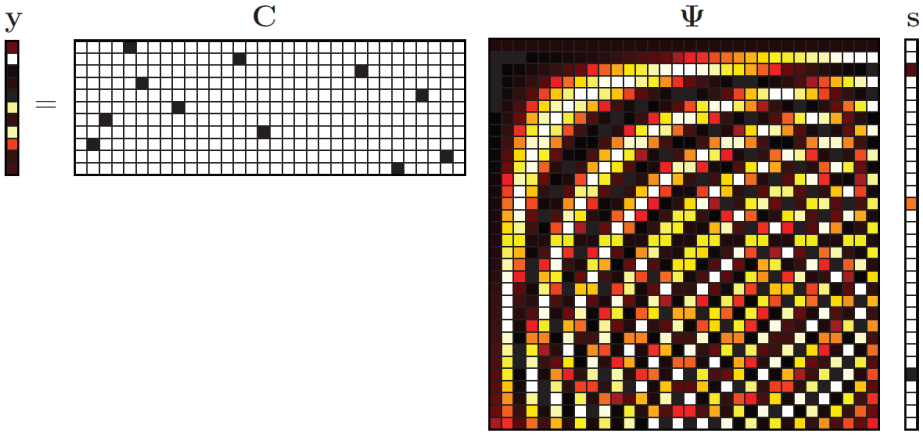

Figure 4.2: Visualisation of matrices and vectors in a compressed sensing system $\mathbf{y} = \mathbf{C\Psi s}$, where $\mathbf{y}$ is a measurement, $\mathbf{C}$ a measurement matrix, $\mathbf{\Psi}$ a transform basis, and $\mathbf{s}$ the sparse representation of the original signal in this basis [7, p. 90].

4.4. Sparse solution recovery

The system in Eq. (4.9) is underdetermined: there are infinitely many vectors $\mathbf{s}$ that solve the system for a certain $\mathbf{y}$ and $\mathbf{\Theta}$. The solution $\mathbf{s}$ that needs to be found is the sparsest solution possible that leads to the obtained measurement: $\mathbf{y} = \mathbf{\Theta s}$. Following the theory of Occam's razor [3], it is assumed that of two possible models, the simplest one is most likely correct. This sparsest solution can be found by optimisation in the ℓ_0 pseudo-norm. In case of noise-free measurements:

$$\text{Minimise } \|\mathbf{s}\|_0 \quad \text{s.t.} \quad \mathbf{y} = \mathbf{\Theta s}. \tag{P_0}$$

Unfortunately, this optimisation is *non-convex*, which in general means that either the objective function (here $\|\mathbf{s}\|_0$) or the feasible set (here $\mathbf{y} = \mathbf{\Theta s}$) is not convex. A convex set is, simply put, a set that contains no holes or indents. More strictly: for any two points in the set, the line segment connecting these points is fully contained in the set as well. A convex function has a graph with a hollow shape: for any two function values, the line segment connecting these points is above the rest of the graph. For many convex optimisation problems - in contrast to non-convex problems - efficient algorithms are available.

In this case, the problem (P_0) is non-convex since the function $\|\cdot\|_0$ is non-convex. The problem requires a brute-force search to find the solution, making it impractical at the very least, and in most cases simply unusable [7, p. 89].

There are multiple possibilities to go from here. For this kind of problem, *greedy algorithms* have been proposed, solving an iterative matching pursuit problem [35]. Others have explored approximating the function $\|\cdot\|_0$ by smoothing functions, for example in sound localisation [21]. Due to these smoothing functions, the optimisation problem becomes convex and therefore solvable. Details of such methods will not be discussed here, but these methods could be used in further research to provide alternatives or additions to the compressed sensing approach that is central in this thesis. In this work, the focus is on replacement of the ℓ_0 optimisation by an ℓ_1 optimisation:

$$\text{Minimise } \|\mathbf{s}\|_1 \quad \text{s.t.} \quad \mathbf{y} = \mathbf{\Theta s}. \tag{P_1}$$

This replacement is allowed if certain conditions hold. These conditions will be described in the upcoming sections. Why these problems lead to similar solutions can intuitively be made clear by a geometric representation of the different norms, shown in Fig. 4.3. The visualisation can be seen in this context as the optimisation of a sparse vector of size 2. The ℓ_0 optimisation ensures the sparsest solution with as many elements of the resulting vector equal to zero. The ℓ_1 optimisation will give the same result in many cases, with an intersection in one of the axes. In contrast, ℓ_2 optimisation will almost never result in a sparse solution.

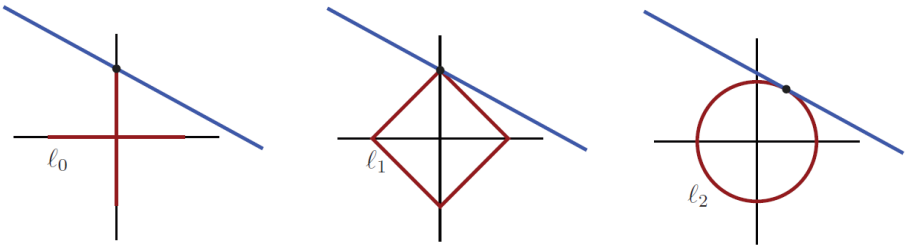

Figure 4.3: Geometric interpretation of optimisation in the ℓ_0, ℓ_1, and ℓ_2 norms [7, p. 96].

For convex ℓ_1 optimisation, many good solvers are available. Hence, the relaxation from ℓ_0 to convex ℓ_1 optimisation, enables sparse optimisation and has made compressed sensing techniques viable. Under what conditions this is possible is described for noise-free and noisy measurements in Section 4.4.1 and Section 4.4.2, respectively.

When using the ℓ_1 norm optimisation, the compressed sensing process can be illustrated by Fig. 4.4. It gives an example in image reconstruction: a few pixels are measured, and the goal is to reconstruct the full image from that. It is done by solving a minimisation problem such as (P_1), followed by a transformation back to the original domain with $\mathbf{x} = \mathbf{\Psi s}$. In this specific case, $\mathbf{\Psi}$ (and the $\mathbf{\Theta}$ that follows from it) is a Fourier transformation base.

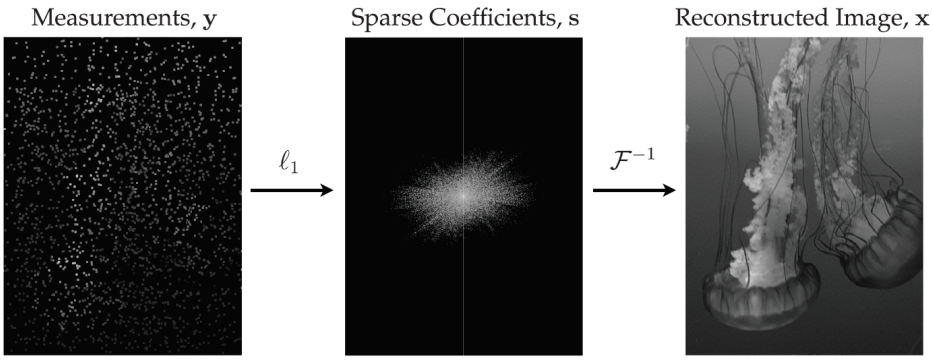

Figure 4.4: Illustration of the compressed sensing process for image reconstruction. Note that this illustration is dramatised according to the author: in reality, a lot more measurements would be needed for proper reconstruction [7, p. 91].

4.4.1. Relaxation from ℓ_0 to ℓ_1 for noise-free measurements

To make compressed sensing viable, the relaxation from ℓ_0 to ℓ_1 optimisation needs to be possible. For randomly chosen measurement matrices $\mathbf{C}$ - think of randomly chosen pixels such as in Fig. 4.4 - it turns out that the results of (P_0) and (P_1) are equal if enough measurements are done. In other words: the solution of the sparse vector $\mathbf{s}$ in (P_1) is exact.

To understand exactly how many measurements would be needed, first the concept of *incoherence* must be explained. If $\mathbf{C}$ is incoherent with respect to $\mathbf{\Psi}$, it means that rows of $\mathbf{C}$ are not correlated with columns of $\mathbf{\Psi}$. Suppose that rows in $\mathbf{C}$ are almost equal to columns - modes - in $\mathbf{\Psi}$. In other words, $\mathbf{C}$ is very coherent with respect to $\mathbf{\Psi}$. The multiplication $\mathbf{C\Psi} = \mathbf{\Theta}$ then results in rows with entries that are very small, except for one. This situation is illustrated in Fig. 4.5, where rows of $\mathbf{C}$ are taken exactly equal to the last columns of $\mathbf{\Psi}$. A measurement here provides no information on the sparse vector, unless one of the last modes is active. In contrast, measurements in the case of an incoherent $\mathbf{C}$ provide information on many different modes, making

it better possible to infer which modes are active and hence estimate $\mathbf{s}$. A formal calculation for the coherence μ of $\mathbf{C}$ and $\mathbf{\Psi}$ is given by [9]:

$$\mu(\mathbf{C},\mathbf{\Psi}) = \sqrt{n}\max_{j,k}|\langle \mathbf{c}_k, \boldsymbol{\psi}_j\rangle|, \tag{4.10}$$

with $\mathbf{c}_k$ rows of $\mathbf{C}$, and $\boldsymbol{\psi}_j$ columns of $\mathbf{\Psi}$.

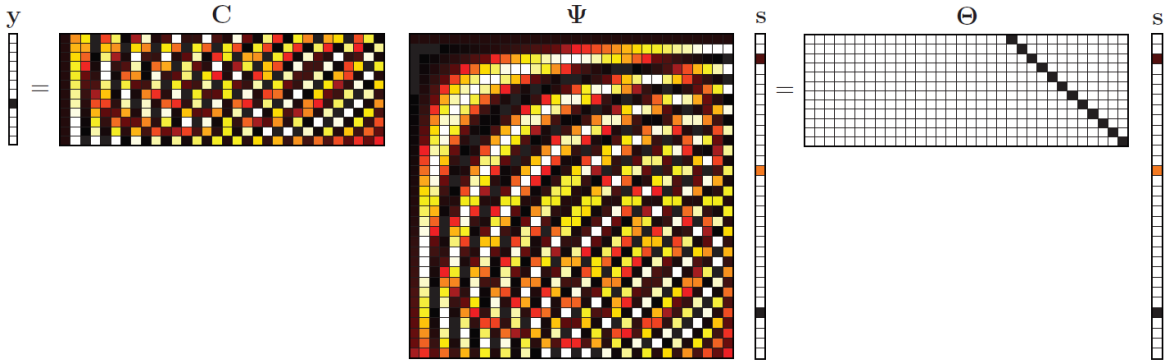

Figure 4.5: Visualisation of matrices in the undesirable case where $\mathbf{C}$ is coherent with respect to $\mathbf{\Psi}$ [7, p. 99].

Now, Theorem 4.2, adapted from [9], gives an exact lower bound on the amount of measurements. It confirms the intuition that the less coherent measurements are with respect to the sparsifying basis, the less measurements are needed. Furthermore, the sparser a signal is, the less measurements are needed. This makes sense: if just a few modes can be active at the same time, these modes can likely be caught by performing few measurements. For Theorem 4.2, first a definition is needed on sparseness of vectors.

Definition 4.1. *A vector $\mathbf{s} \in \mathbb{R}^n$ is K-sparse if at most K entries of $\mathbf{s}$ are non-zero.*

Theorem 4.2. *Let $\mathbf{s} \in \mathbb{R}^n$ be a signal that is K-sparse in the basis $\mathbf{\Psi}$. Let $\mathbf{\Phi}$ be an orthonormal basis. Suppose m measurements are taken randomly in the $\mathbf{\Phi}$ domain. If for some positive constant k_1,*

$$m \geq k_1 \cdot \mu^2(\mathbf{\Phi},\mathbf{\Psi}) \cdot K \cdot \log n, \tag{4.11}$$

then the solution to (P_1) is exact with overwhelming probability.

Remark. *This probability exceeds $1-\delta$ if $m \geq k_1 \cdot \mu^2(\mathbf{\Phi},\mathbf{\Psi}) \cdot K \cdot \log(n/\delta)$ [9].*

4.4.2. Relaxation from ℓ_0 to ℓ_1 for noisy measurements

Until now, the focus has been on noise-free measurements and exact reconstruction of a sparse signal. In reality, this ideal situation is almost never encountered. For compressed sensing techniques to be powerful in real-life applications, they need to be able to handle noisy data. Suppose that instead of Eq. (4.8), the measurement $\mathbf{y}$ is noisy:

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{z} = \mathbf{\Theta}\mathbf{s} + \mathbf{z}, \tag{4.12}$$

where $\mathbf{z} \in \mathbb{R}^n$ is a bound error term: $\|\mathbf{z}\|_2 \leq \epsilon$.

Looking at the optimisation problem (P_1), the condition $\mathbf{y} = \mathbf{\Theta}\mathbf{s}$ cannot hold exactly if noise is present. Therefore, a relaxed condition is considered:

$$\text{Minimise } \|\mathbf{s}\|_1 \quad \text{s.t.} \quad \|\mathbf{y} - \mathbf{\Theta}\mathbf{s}\|_2 \leq \epsilon. \tag{$P_{1,\epsilon}$}$$

Here, ϵ is some bound related to the noise level of $\mathbf{z}$. This optimisation problem is *convex* - refer to Section 4.4.3 for details and the proof.

For a good recovery, the aim is to have little difference between the solution $\mathbf{s}$ of $(P_{1,\epsilon})$ and the sparsest solution $\mathbf{s}_0$ of

$$\text{Minimise } \|\mathbf{s}\|_0 \quad \text{s.t.} \quad \|\mathbf{y} - \mathbf{\Theta}\mathbf{s}\|_2 \leq \epsilon. \tag{$P_{0,\epsilon}$}$$

Results on how well the solution of $(P_{1,\epsilon})$ approximates the solution of $(P_{0,\epsilon})$ depend on a certain property of the matrix $\mathbf{\Theta}$: the *restricted isometry property (RIP)*. Mathematically, the *isometry constant* δ_K is defined as the smallest number such that

$$(1-\delta_K)\|\mathbf{s}\|_2^2 \leq \|\mathbf{\Theta s}\|_2^2 \leq (1+\delta_K)\|\mathbf{s}\|_2^2 \tag{4.13}$$

for all K-sparse vectors $\mathbf{s}$. If δ_K is not too close to one, then the RIP of order K is said to hold for $\mathbf{\Theta}$. The idea of this property is that if a matrix satisfies the RIP, then multiplying a sparse vector $\mathbf{s}$ by this matrix will not influence its ℓ_2 norm much - its Euclidean length is approximately preserved. Another useful perspective on this property is that it indicates that any subset of K columns of $\mathbf{\Theta}$ is approximately orthogonal [7, 9].

Why is this property useful in compressed sensing? Recall that the goal is to have a solution to $(P_{1,\epsilon})$ that is close to the sparsest solution, of $(P_{0,\epsilon})$. If the RIP holds, then $\mathbf{\Theta}$ does not alter the Euclidean length of $\mathbf{s}$ much. Suppose we have two K-sparse vectors, $\mathbf{s}$ and $\mathbf{s}^*$, with a reasonable distance between them. Multiplying by $\mathbf{\Theta}$ creates measurements $\mathbf{y}$ and $\mathbf{y}^*$. The RIP ensures that these measurements cannot be very close together, nor can they be very far apart. Plugging the $2K$-sparse vector $\mathbf{s}-\mathbf{s}^*$ in Eq. (4.13) gives:

$$(1-\delta_{2K})\left\|\mathbf{s}-\mathbf{s}^*\right\|_2^2 \leq \left\|\mathbf{y}-\mathbf{y}^*\right\|_2^2 \leq (1+\delta_{2K})\left\|\mathbf{s}-\mathbf{s}^*\right\|_2^2. \tag{4.14}$$

This gives confidence that it is possible to properly reconstruct sparse signals based on compressive measurements. [9] The RIP is used in many results obtained in the field of compressed sensing. One of them is stated in Theorem 4.3, and is due to [8]. It establishes a bound on the ℓ_2 error of $\mathbf{s}_{1,\epsilon}$, the solution to $(P_{1,\epsilon})$. A proof can also be found in [8].

Theorem 4.3. *Consider the vector* $\mathbf{s}$ *in the system of Eq.* (4.12)*, with* $\|\mathbf{z}\|_2 \leq \epsilon$*. Let* $\mathbf{s}_K$ *be the vector* $\mathbf{s}$*, but with all but the largest K components set to 0. Consider the optimisation problem* $(P_{1,\epsilon})$*, and assume that for the isometry constant of* $\mathbf{\Theta}$*,* $\delta_{2K} < \sqrt{2}-1$ *holds. Then the solution* $\mathbf{s}_{1,\epsilon}$ *to* $(P_{1,\epsilon})$ *obeys*

$$\left\|\mathbf{s}_{1,\epsilon}-\mathbf{s}\right\|_2 \leq C_0 \frac{\left\|\mathbf{s}-\mathbf{s}_K\right\|_1}{\sqrt{K}} + C_1\epsilon, \tag{4.15}$$

for constants C_0 *and* C_1 *depending on* δ_{2K} *as follows:*

$$\begin{aligned} C_0 &= 2\frac{1-(1-\sqrt{2})\delta_{2K}}{1-(1+\sqrt{2})\delta_{2K}}; \\ C_1 &= \frac{4\sqrt{1+\delta_{2K}}}{1-(1+\sqrt{2})\delta_{2K}}. \end{aligned} \tag{4.16}$$

Remark. *If* $\mathbf{s}$ *is K-sparse, then* $\mathbf{s}=\mathbf{s}_K$*, hence the reconstruction error is given by*

$$\left\|\mathbf{s}_{1,\epsilon}-\mathbf{s}\right\|_2 \leq C_1\epsilon. \tag{4.17}$$

Remark. *The constants* C_0 *and* C_1 *are in general small. For example, when* $\delta_{2K}=0.2$*, then* $\left\|\mathbf{s}_{1,\epsilon}-\mathbf{s}\right\|_2 \leq 4.2\frac{\|\mathbf{s}-\mathbf{s}_K\|_1}{\sqrt{K}} + 8.5\epsilon$ [8]*. A graph of* C_0 *and* C_1 *versus* δ_{2K} *is shown in Fig. 4.6.*

Considering this result, the aim is to find a measurement matrix $\mathbf{C}$ such that $\mathbf{\Theta}$ satisfies the RIP. If that is the case, the relaxed optimisation problem $(P_{1,\epsilon})$ may be used. The first result on matrices $\mathbf{\Theta}=\mathbf{C\Psi}$ satisfying the RIP is for a random measurement matrix $\mathbf{C}$ and an orthonormal basis $\mathbf{\Psi}$. Random measurement matrices can be Bernoulli or Gaussian distributed, for example. If the number of measurements m is large enough, then $\mathbf{\Theta}$ satisfies the RIP. To be precise,

$$m \geq CK\log(n/K), \tag{4.18}$$

where K is the sparsity of the signal to be reconstructed, n is the size of the signal vector, and C is some constant depending on both. For a more in-depth discussion, see [4].

Throughout this project, some matrices $\mathbf{C}$ in $\mathbf{\Theta}=\mathbf{C\Psi}$ are chosen randomly. For those, it is assumed that they satisfy the RIP. For other matrices $\mathbf{\Theta}$, it will become clear in Section 5.3.2 that rows of $\mathbf{\Psi}$ are chosen to be in $\mathbf{\Theta}$ such that the condition number of $\mathbf{\Theta}$ is minimised. This corresponds to $\mathbf{\Theta}$ being as orthogonal as possible at the chosen elements. As discussed before, satisfying the RIP corresponds to have approximately orthogonal columns. Hence, it will be assumed that also the matrices $\mathbf{\Theta}$ that are not created by a randomly chosen measurement matrix $\mathbf{C}$ satisfy the RIP.

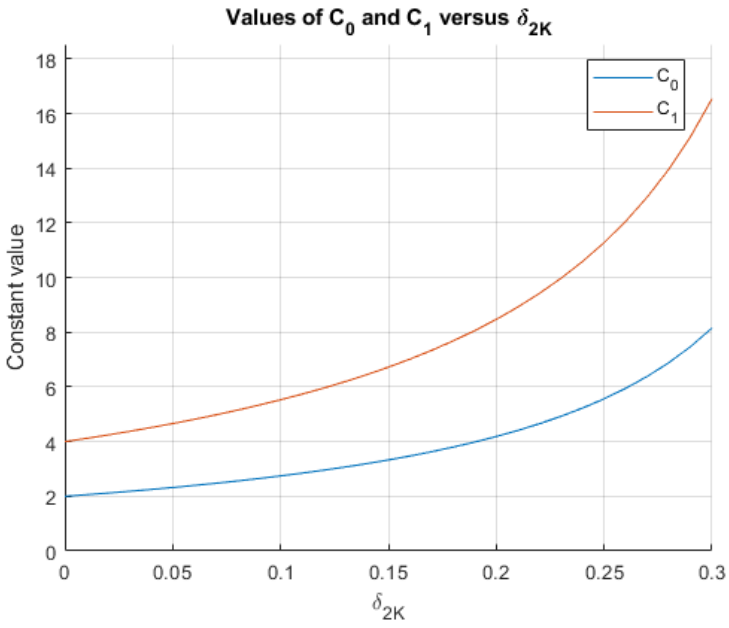

Figure 4.6: Values of C_0 and C_1 plotted against δ_{2K}, as in Eq. (4.16).

4.4.3. Convexity of the minimisation problems

In this section, convexity of optimisation problems (P_1) and $(P_{1,\epsilon})$ is proved. In fact, we present the proof for the general optimisation problem Eq. (4.19). The vector $\mathbf{c}$ appearing in the objective function can be seen as additional weights to the sparse vector $\mathbf{s}$. A motivation to use such weights will be presented in Section 5.3.4. Problems (P_1) and $(P_{1,\epsilon})$ can be obtained from Eq. (4.19) by making $\mathbf{c}$ a vector of all ones. To obtain (P_1), one must additionally set $\epsilon = 0$.

Lemma 4.4. *The following optimisation problem, with $\mathbf{s} \in \mathbb{R}^n$, for given $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^p$, $\boldsymbol{\Theta} \in \mathbb{R}^{p\times n}$, and ϵ real and non-negative, is convex:*

$$\underset{\mathbf{s}}{\text{Minimise}} \ \|\mathbf{c}\odot\mathbf{s}\|_1 \quad s.t. \quad \left\|\mathbf{y}-\boldsymbol{\Theta}\mathbf{s}\right\|_2 \le \epsilon. \tag{4.19}$$

Here, $\odot$ denotes element wise multiplication (the Hadamard product).

Proof. To show that the problem is convex, we must show that both the objective function

$$f_0(\mathbf{s}) = \|\mathbf{c}\odot\mathbf{s}\|_1 \tag{4.20}$$

and the inequality constraint function

$$f_1(\mathbf{s}) = \left\|\mathbf{y}-\boldsymbol{\Theta}\mathbf{s}\right\|_2 - \epsilon = \left\|\boldsymbol{\Theta}\mathbf{s}-\mathbf{y}\right\|_2 - \epsilon \tag{4.21}$$

are convex functions.

Let $\lambda \in [0,1]$, and let $\mathbf{s}_1, \mathbf{s}_2 \in \mathbb{R}^n$. For both of the following proofs, the triangle inequality is used, which holds by definition for norms. For f_0, we have, using the Hadamard product's distributive property:

$$\begin{aligned} f_0(\lambda\mathbf{s}_1 + (1-\lambda)\mathbf{s}_2) &= \left\|\mathbf{c}\odot(\lambda\mathbf{s}_1 + (1-\lambda)\mathbf{s}_2)\right\|_1 \\ &= \left\|\lambda\mathbf{c}\odot\mathbf{s}_1 + (1-\lambda)\mathbf{c}\odot\mathbf{s}_2)\right\|_1 \\ &\le \lambda\left\|\mathbf{c}\odot\mathbf{s}_1\right\|_1 + (1-\lambda)\left\|\mathbf{c}\odot\mathbf{s}_2\right\|_1 \\ &= \lambda f_0(\mathbf{s}_1) + (1-\lambda) f_0(\mathbf{s}_2). \end{aligned} \tag{4.22}$$

Likewise, we obtain for f_1:

$$\begin{aligned} f_1(\lambda \mathbf{s}_1 + (1-\lambda)\mathbf{s}_2) &= \left\| \mathbf{\Theta}(\lambda \mathbf{s}_1 + (1-\lambda)\mathbf{s}_2) - \mathbf{y} \right\|_2 - \epsilon \\ &= \left\| \lambda(\mathbf{\Theta}\mathbf{s}_1 - \mathbf{y}) + (1-\lambda)(\mathbf{\Theta}\mathbf{s}_2 - \mathbf{y}) \right\|_2 - \epsilon \\ &\leq \lambda \left\| \mathbf{\Theta}\mathbf{s}_1 - \mathbf{y} \right\|_2 + (1-\lambda) \left\| \mathbf{\Theta}\mathbf{s}_2 - \mathbf{y} \right\|_2 - \epsilon \\ &= \lambda \left(\left\| \mathbf{\Theta}\mathbf{s}_1 - \mathbf{y} \right\|_2 - \epsilon \right) + (1-\lambda) \left(\left\| \mathbf{\Theta}\mathbf{s}_2 - \mathbf{y} \right\|_2 - \epsilon \right) \\ &= \lambda f_1(\mathbf{s}_1) + (1-\lambda) f_1(\mathbf{s}_2). \end{aligned} \tag{4.23}$$

□

4.5. Characterising and solving the minimisation problems

Consider again the minimisation problem of Eq. (4.19). This problem needs to be solved numerically in the application of compressed sensing in this work. To determine what kind of solver is needed for this, it is first important to identify what class the problem belongs to.

This task becomes easier when the problem is rewritten slightly. The objective function $\|\mathbf{c} \odot \mathbf{s}\|_1$ is equal to $\sum_{i=1}^{n} |c_i s_i|$. It turns out to be helpful to introduce a second variable to get rid of these absolute values in the objective function. Let $|c_i s_i| \leq t_i$ for all $i = 1, \ldots, n$. Then minimising all elements of $\mathbf{t}$ is at the same time minimising the original objective function. Since $\mathbf{t}$ is positive (its elements are larger than some absolute values), the new objective function can be set to $\mathbf{1}^T\mathbf{t}$. The new constraint $|c_i s_i| \leq t_i$ is equivalent to $c_i s_i \leq t_i$ and $c_i s_i \geq -t_i$. Hence, the problem can be rewritten as

$$\min_{\mathbf{t},\mathbf{s}} \mathbf{1}^T\mathbf{t} \tag{4.24}$$

$$\text{s.t. } \left\| \mathbf{y} - \mathbf{\Theta}\mathbf{s} \right\|_2 \leq \epsilon; \tag{4.25}$$

$$c_i s_i \leq t_i \quad \text{for all } i = 1, \ldots, n; \tag{4.26}$$

$$c_i s_i \geq -t_i \quad \text{for all } i = 1, \ldots, n. \tag{4.27}$$

Then, the ℓ_2 inequality is rewritten by squaring the norm:

$$\left\| \mathbf{y} - \mathbf{\Theta}\mathbf{s} \right\|_2^2 = (\mathbf{y} - \mathbf{\Theta}\mathbf{s})^T (\mathbf{y} - \mathbf{\Theta}\mathbf{s}) = \mathbf{y}^T\mathbf{y} + \mathbf{s}^T\mathbf{\Theta}^T\mathbf{\Theta}\mathbf{s} - 2\mathbf{y}^T\mathbf{\Theta}\mathbf{s}. \tag{4.28}$$

Since the norm is always positive, this term can be taken to be less or equal than ϵ^2. This leaves us with the following form of the problem:

$$\min_{\mathbf{t},\mathbf{s}} \mathbf{1}^T\mathbf{t} \tag{4.29}$$

$$\text{s.t.} \mathbf{y}^T\mathbf{y} + \mathbf{s}^T\mathbf{\Theta}^T\mathbf{\Theta}\mathbf{s} - 2\mathbf{y}^T\mathbf{\Theta}\mathbf{s} \leq \epsilon^2; \tag{4.30}$$

$$c_i s_i \leq t_i \quad \text{for all } i = 1, \ldots, n; \tag{4.31}$$

$$c_i s_i \geq -t_i \quad \text{for all } i = 1, \ldots, n. \tag{4.32}$$

This is a linear program with quadratic constraints (QCLP). That the objective function $\mathbf{1}^T\mathbf{t}$ is linear needs no further explanation, and the first constraint is indeed of the form $\frac{1}{2}\mathbf{s}^T P\mathbf{s} + \mathbf{q}^T\mathbf{s} + r \leq 0$, making it quadratic. The other two constraints are linear, and are therefore also part of the class of quadratic constraints.

If the problem at hand would not have involved noisy data, then the first constraint would have been linear: $\mathbf{y} = \mathbf{\Theta}\mathbf{s}$, and therefore the problem would be an instance of linear programming (LP). This type of problems if fairly easy to solve using the simplex method.

However, since we are dealing with QCLP, another optimisation algorithm must be used. QCLP is a subclass of SOCP (second-order conic programming), where the objective function must also be linear, but the constraints may also be of the form $\left\| \mathbf{A}\mathbf{x} - \mathbf{b} \right\|_2 \leq \mathbf{c}^T\mathbf{x} + d$, where the objective function is optimised over $\mathbf{x}$. SOCP problems can be solved efficiently using interior point methods, in which the solution is found after iterations in which a path through the feasible region is followed.

The solver that will be used in this study is SDPT3 [33], which implements interior point methods for SDP problems. The class of SOCP problems is contained within the larger class of SDP problems. In particular, the interior point methods that are used in SDPT3 are primal-dual path-following algorithms. These contain prediction and correction steps that were first introduced by Mehrotra [24].

5

Localisation algorithm

Using magnetic field sensors, magnetised objects can be detected, localised and classified. The objects that must be detected are mostly far away from the sensors that are used to detect them. In these situations, their field is approximately that of a magnetic dipole. In this algorithm, the focus is on estimation of the location and moment of a single magnetic dipole. The detection problem is not considered - it is assumed a magnetic dipole is present.

In the upcoming sections, first a description of the problem is given, and then it is explained how compressed sensing is applied to it. After that, the implemented algorithm is described, and the results obtained by applying this algorithm are shown.

5.1. Problem description

In the magnetic dipole localisation problem, a search space Ω is considered. This space contains a single magnetic dipole, with an unknown location and magnetic moment. The goal of the implemented algorithm is to estimate the location and moment of this magnetic dipole using simulated measurements of a limited number of optimally chosen sensors. The problem is illustrated in Fig. 5.1.

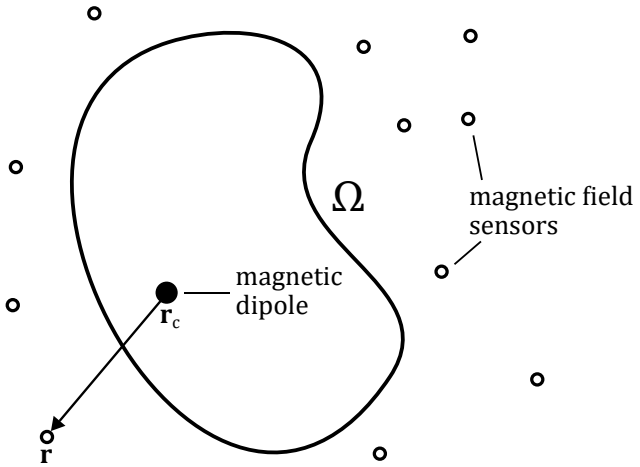

Figure 5.1: Illustration of the localization problem, showing the search space Ω containing one magnetic dipole with centre location $\mathbf{r}_c$, and magnetic field sensors outside Ω. The location of a field measurement is denoted by $\mathbf{r}$.

In the specific instance of the problem that is discussed in this thesis, a horizontally placed sensor array is considered. In a 3D space above, a single magnetic dipole (with random location and magnetic moment) is placed. One can think of this as the sensor array containing evenly spaced drone measurements, and the object being in some space below the drone flight path. The goal of the implemented algorithm is to estimate the location and moment of this magnetic dipole using simulated measurements of a limited number of sensors of the sensor array.

The exact considered setup is as follows: the search area spans from x = -0.3 m to 0.3 m, from y = -0.15 m to 0.15 m and from z = -0.35 to -0.15 m. The sensor array is placed at z = 0.145 m and sensors are evenly spaced between x = -0.38 m and 0.38 m and between y=-0.18 m and 0.18 m, with a distance of 4 cm between sensors in both directions. An illustration of this setup is given in Fig. 5.2.

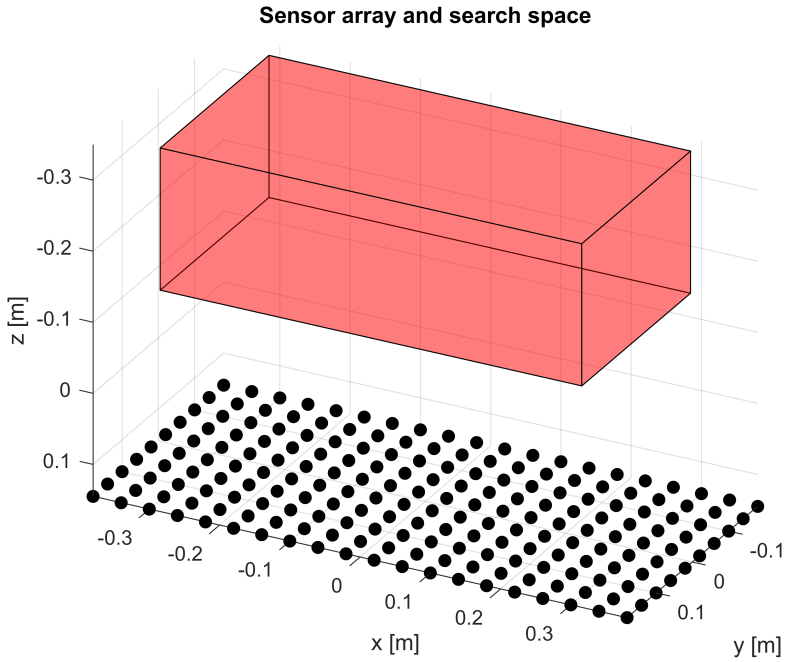

Figure 5.2: An illustration of the setup, where the black dots indicate the sensors of the array, and the red box indicates the area in which the magnetic dipole is located. Note that the convention of the z-axis pointing downwards is used.

5.2. Applying compressed sensing to magnetic dipole localisation

For the localisation problem as described in the previous section, the sparseness that is needed is present in the location domain - it is assumed that there is only one magnetic dipole. In other words, the sparse domain is here chosen to be the location domain. Recall that the goal in compressed sensing applications is to solve a system

$$\mathbf{y} = \mathbf{C\Psi s} = \mathbf{\Theta s}, \tag{5.1}$$

where $\mathbf{s}$ is sparse, as described in Section 4.3. It makes sense to think of the vector $\mathbf{s}$ as representing unique locations and moments of single magnetic dipoles. An arbitrary magnetic dipole could then be represented by an 'average' of a few magnetic dipoles. The entries corresponding to these dipoles would then be non-zero, and other entries would be zero, making $\mathbf{s}$ sparse. To cover the full search area, a grid of magnetic dipoles is chosen. At each grid point, six magnetic dipoles are considered: with their magnetic moment pointing in the positive and negative x, y, and z direction. This interpretation of the sparse vector is illustrated in Fig. 5.3. The assumption here is that the field of a dipole can be represented as an average of the fields of other dipoles close by.

The signal $\mathbf{x}$ is defined as the magnetic field $\mathbf{B}$ at the locations of all sensors in the sensor array, ordered as $[B_{x,1}, B_{y,1}, B_{z,1}, B_{x,2}, \ldots, B_{z,k}]$, with $1, 2, \ldots, k$ representing all the available sensors. A measurement $\mathbf{y}$ contains

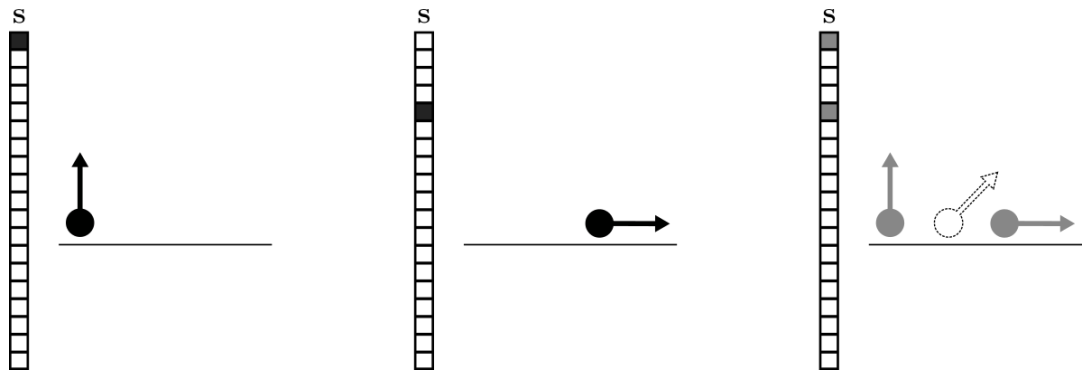

Figure 5.3: The interpretation of the sparse vector **s**. The left and middle vector represent two different dipoles, and from the right vector, an 'average' dipole (dotted line) can be reconstructed.

a selection of this field, with just the x-, y-, and z-component of selected sensors. Hence, $\mathbf{C}$ is a matrix containing selected rows from the identity matrix - sometimes this is called a *selection matrix*.

The entries of $\mathbf{s}$ correspond to unique magnetic dipoles, and $\mathbf{y}$ is a measurement of the field at the plane of the sensor array. Following the system Eq. (5.1), the basis $\mathbf{\Psi}$ must consist of columns representing the field of a single magnetic dipole at the sensor plane. This basis must be generated at the start of the algorithm. The details are be explained in Section 5.3.

5.3. Overview of the algorithm

A compressed sensing algorithm for estimating field, location and moment of a magnetic dipole has been implemented. The algorithm must first be initialised by determining the basis $\mathbf{\Psi}$, and then follows several steps to produce an estimate. This section describes both the initialisation and the different steps.

The main algorithm can be summarised by the following steps, given that just a selection of n sensors of the array can be used. The steps are also visualised in a flow diagram in Fig. 5.4.

1. Initialise basis $\mathbf{\Psi}$.
2. Choose n sensors to perform measurements, either randomly or using QR pivoting (see Section 5.3.2). This results in a measurement matrix $\mathbf{C}$. Multiplying by the initialised basis $\mathbf{\Psi}$ gives the required matrix $\mathbf{\Theta}$.
3. Obtain the measurement vector $\mathbf{y}$ of size $3n$ from these chosen sensors.
4. Perform the ℓ_1 optimisation $(P_{1,\epsilon})$.
5. Classification: compute an estimated location and moment from the resulting sparse vector $\mathbf{s}$. An estimated magnetic dipole field also follows.

The upcoming subsections describe the basis initialisation and all steps of the algorithm one by one, in more detail.

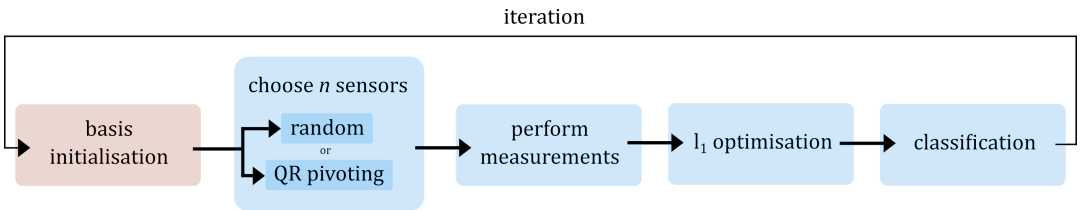

Figure 5.4: Flow diagram of the algorithm steps, where basis initialisation forms the starting point.

5.3.1. Basis initialisation

For a compressed sensing algorithm to work, a basis $\mathbf{\Psi}$ must be found that allows for transformation between the original and sparse domain of the signal. To do this, a set of initialisation locations has been chosen, representative of the locations the actual magnetic dipole might be in. For this, a predefined 3D grid of locations is defined, covering the full search area.

Every entry of the sparse vector $\mathbf{s}$ corresponds to a single magnetic dipole at one of the initialisation locations, with moments of $1\,\mathrm{Am}^2$ in x, y, z, and $-x$, $-y$, $-z$ direction. Every one of these dipoles creates a unique field at the sensors. Following Eq. (4.7), $\mathbf{\Psi}$ can be formed by putting the measurements (columns) of the initialisation dipoles together. In the resulting matrix, the n^{th} column is the field corresponding to the n^{th} initialisation dipole, as in Fig. 5.5. The measurements used for initialisation of $\mathbf{\Psi}$ are noise-free - in fact, they are simply calculations.

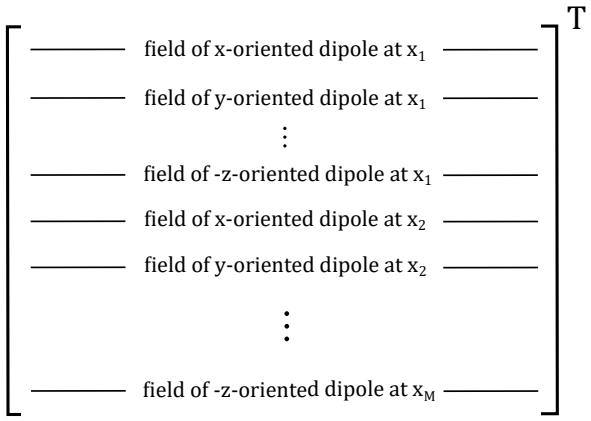

Figure 5.5: Structure of the transformation matrix $\mathbf{\Psi}$ for M grid points.

5.3.2. Sensor choice

In this compressed sensing application, we are interested in reconstructing the magnetic field (and thereby obtaining a location and moment estimate) with just a few measurements. Each of the measurements is the output of one of the sensors in the sensor array depicted in Fig. 5.2. Suppose that just n sensors are available to perform measurements. Every sensor measures three components of the field, giving a total of $p = 3n$ measurements. It is not immediately clear how to choose these sensors, or how to show if a particular choice is good or bad. This section discusses both topics, and introduces two methods of choosing sensors: random selection and optimal selection using QR decomposition.

In the system $\mathbf{x} = \mathbf{C\Psi s} = \mathbf{\Theta s}$, the condition number $\kappa(\mathbf{\Theta})$ determines how well $\mathbf{s}$ can be approximated from some noisy measurement $\mathbf{y}$ (see Section 3.5). The goal is to minimise this condition number by choosing an appropriate measurement matrix $\mathbf{C}$. If $\mathbf{\Theta}$ is not square, the aim is to reduce the condition number of $\mathbf{\Theta}^T\mathbf{\Theta}$. This is because this term is involved in the pseudo-inverse, a generalisation of the inverse to non-square matrices [7, p.110-111]. Now we consider two ways of choosing sensors: randomly and using QR pivoting.

Random choice

The first of the two considered ways to select n sensors is random selection. When this method is applied, n sensors are drawn randomly from the set of available sensors, without repetition. All chosen sensors will then measure the 3 components B_x, B_y and B_z, giving $p = 3n$ measurements in total. The resulting measurement matrix $\mathbf{C}$ consists of rows of the identity matrix corresponding to the measurements of the chosen sensors. It turns out that the condition number of $\mathbf{\Theta}$ is generally very large in this case, indicating that a reconstruction of $\mathbf{s}$ from $\mathbf{y}$ could contain a large error. The condition number can be reduced by *oversampling*. If $\mathbf{\Psi}$ is of rank r, performing $p = r+10$ or more measurements greatly reduces the condition number, allowing for better signal reconstruction [7, p.111]. However, depending on the initialisation grid choice, $\mathbf{\Psi}$ has many columns. This means that the rank r of $\mathbf{\Psi}$ is in general equal to the number of rows in $\mathbf{\Psi}$. This number of rows represents

the amount of possible measurements. It is therefore not possible to choose an amount of measurements p that is larger than the rank r of $\mathbf{\Psi}$.

Choice using QR pivoting

For a smarter choice of sensors, QR decomposition is used, in combination with a singular value decomposition (SVD). As explained in Section 3.4, a QR decomposition with column pivoting of an $m \times n$ matrix $\mathbf{A}$ takes the form

$$\mathbf{AP} = \mathbf{QR}, \tag{5.2}$$

where $\mathbf{Q}$ is a unitary $m \times m$ matrix, $\mathbf{R}$ an upper triangular $m \times n$ matrix, and $\mathbf{P}$ an $n \times n$ permutation matrix - in other words, a matrix that describes how the columns of $\mathbf{A}$ are changed. For a detailed explanation of the QR decomposition including column pivoting, refer to Section 3.4.

From applying a *greedy algorithm* for QR factorisation with column pivoting, an optimal choice for $\mathbf{P}$ follows, where the condition number of $\mathbf{A}$ is minimal. A greedy algorithm tries to optimise a problem by choosing the local optimum in each iteration. In this case: choosing a permutation for $\mathbf{P}$ such that the condition number of Θ is reduced most. That this method really works, becomes apparent when running a test with 16 sensors, with both random sensor choice and using QR pivoting. Where condition numbers of $\mathbf{\Theta}$ for random sensor choice vary between 100 and 1000, this is reduced to a (fixed) value around 50 for sensor choice with QR pivoting - a reduction by a factor of 10 on average.

The goal is to find the optimal sensors, corresponding to rows of $\mathbf{\Psi}$. This method is suited for finding optimal columns, so in this context, the transposes of the basis $\mathbf{\Psi}$ and the measurement matrix $\mathbf{C}$ are considered:

$$\mathbf{\Psi}^T\mathbf{C}^T = \mathbf{QR}. \tag{5.3}$$

The pivots from the permutation in this decomposition indicate the order of best representing the basis modes in $\mathbf{\Psi}$. The above decomposition is only applicable if $\mathbf{\Psi}$ is square. In the case where $\mathbf{\Psi}$ has more rows than columns (oversampling), the condition number of $\mathbf{\Theta}^T\mathbf{\Theta}$ needs to be minimised, as mentioned earlier in this section. Hence, the following decomposition can be used [7, p. 112]:

$$(\mathbf{\Psi}\mathbf{\Psi}^T)\mathbf{C}^T = \mathbf{QR}. \tag{5.4}$$

In general, depending on the chosen initialisation grid, $\mathbf{\Psi}$ has more rows than columns (undersampling). In that case, this method is not directly applicable. This issue can be solved by a matrix of lower rank: a reduced SVD-basis. This method is described later in this section.

It must be noted that the resulting pivots indicate specific *measurement* indices, not sensors. Not all three measurements B_x, B_y, B_z from one sensor have to appear consecutively as pivots - in other words, three components may have different importance in representing the basis modes. Therefore, sensors have been chosen in order of either of their components appearing in the pivot sequence.

Reducing the rank of basis $\mathbf{\Psi}$

For a successful application of the QR pivoting method, it is needed that the number of columns of $\mathbf{\Psi}$ is smaller or equal than the number of rows. This is dependent on the initialisation grid, and in general not the case. It is possible to describe only a limited number of field modes using the columns of a new basis $\mathbf{\Psi}_r$.

In this algorithm, a reduction of the $m \times n$ matrix $\mathbf{\Psi}$ to r modes is obtained by performing an SVD, of which a more detailed discussion can be found in Section 3.3:

$$\mathbf{\Psi} = \mathbf{U\Sigma V}^T, \tag{5.5}$$

where $\mathbf{U} \in \mathbb{R}^{m\times m}$ and $\mathbf{V} \in \mathbb{R}^{n\times n}$ are unitary matrices, and $\mathbf{\Sigma} \in \mathbb{R}^{m\times n}$ is diagonal. Reduced versions of the matrices are created, where $\tilde{\mathbf{\Sigma}}$ contains the upper left $r \times r$ block of $\mathbf{\Sigma}$, and $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$ the first r columns of $\mathbf{U}$ and $\mathbf{V}$, respectively. Then, the reduced basis $\mathbf{\Psi}_r$ is computed by

$$\mathbf{\Psi}_r = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^T. \tag{5.6}$$

It might seem that a lot of the information from the original basis is lost when reducing it to much less modes. However, it turns out that a very large part of the fields from the original $\mathbf{\Psi}$ can be represented as a combination of the first few modes available in $\mathbf{\Psi}_r$. This is quantified by looking at the singular values corresponding to each of the columns of $\mathbf{\Psi}_r$. The *energy* each of these modes contains is represented by its singular value. The fraction of the energy η of the first r modes of $\mathbf{\Psi}$ is given by the sum of its singular values, divided by the total sum of all n singular values:

$$\eta(r) = \frac{\sum_{i=1}^{r} \sigma_i}{\sum_{i=1}^{n} \sigma_i} \tag{5.7}$$

From Fig. 5.6, it becomes clear that the first few dozen modes already contain a very large part of the total energy.

If $\mathbf{\Psi}_r$ contains a large enough fraction of total energy from $\mathbf{\Psi}$, almost all fields from $\mathbf{\Psi}$ can be described by modes from $\mathbf{\Psi}_r$. Sensors chosen with the QR pivoting method, and with $\mathbf{\Psi}_r$ as a basis, are best for differentiating between modes from $\mathbf{\Psi}_r$ with a measurement $\mathbf{y}$. It is assumed that if modes in $\mathbf{\Psi}_r$ contain most energy from $\mathbf{\Psi}$, then sensors resulting from QR pivoting using $\mathbf{\Psi}_r$ should also be good for distinguishing between unique magnetic dipoles as represented in the basis $\mathbf{\Psi}$.

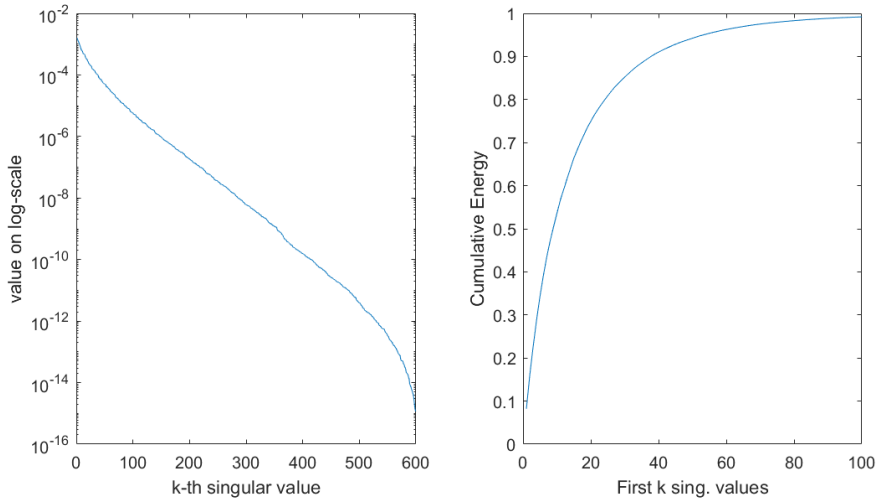

Figure 5.6: Left: sorted singular values. Right: the cumulative energy of the first 100 singular values.

5.3.3. Performing measurements

To simulate magnetic field measurements by the n chosen sensors, the following formula for the magnetic field is used, see Eq. (2.13):

$$\mathbf{B}_{\text{dip}}(\mathbf{r}) = \frac{\mu_0}{4\pi}\left(\frac{3\hat{\mathbf{r}}(\mathbf{m}\cdot\hat{\mathbf{r}}) - \mathbf{m}}{r^3}\right). \tag{5.8}$$

Here, μ_0 is again the permeability of free space, $\mathbf{m}$ is the magnetic moment of the dipole, $\hat{\mathbf{r}}$ is the unit vector pointing from the magnetic dipole to the sensor, and r is the absolute distance between the magnetic dipole and sensor.

To every component (x, y, z) of every sensor measurement independently, some Gaussian noise is added, i.i.d. with a standard deviation $\sigma = 100\,\text{nT}$.

5.3.4. ℓ_1 optimisation

As described in Section 4.4, the computationally hard problem of finding the sparsest near-solution can be replaced by an ℓ_1 optimisation. The minimisation problem stated in the previous chapter is ($P_{1,\epsilon}$):

$$\text{Minimise } ||\mathbf{s}||_1 \quad \text{s.t.} \quad ||\mathbf{y}-\mathbf{\Theta s}||_2 \leq \epsilon. \tag{$P_{1,\epsilon}$}$$

There is, however, a problem with this formulation. The goal is to find the sparsest vector **s**, but this optimisation problem will not give such result. The issue here is that dipoles close to the sensor array generate a much larger field in the sensors than dipoles far away. Since these fields are larger, smaller entries of **s** - or smaller dipole moments - are needed to approximate the measurement **y** by $\mathbf{\Theta s}$. The 1-norm of **s** is minimised, and therefore the algorithm will always prefer activating multiple smaller moments close to the sensor array instead of activating the expected dipoles near the actual location.

To overcome this problem, some compensation is needed such that dipoles near the sensor array are no longer preferred over those further away. This is done by adding a penalty to each element of **s**, representing 'how preferred' the dipole was. All penalties are stored in a vector **c**. Note that scaling the columns of $\mathbf{\Theta}$ could also have been an option. Since the magnetic dipole field drops by a factor r^3, where r is the distance to the dipole, a sensible choice is to make the penalty of a dipole location equal to $\frac{1}{d^3}$, where d is the vertical distance from the dipole to the sensor array. This way, for sensors close to the array, d is small, and the penalty is large. The resulting objective function becomes $||\mathbf{c}\odot\mathbf{s}||_1$. The resulting values in the compensation vector are shown in Fig. 5.7. Since the initialisation grid consists of dipole layers, the penalty increases in steps, and are constant per height level.

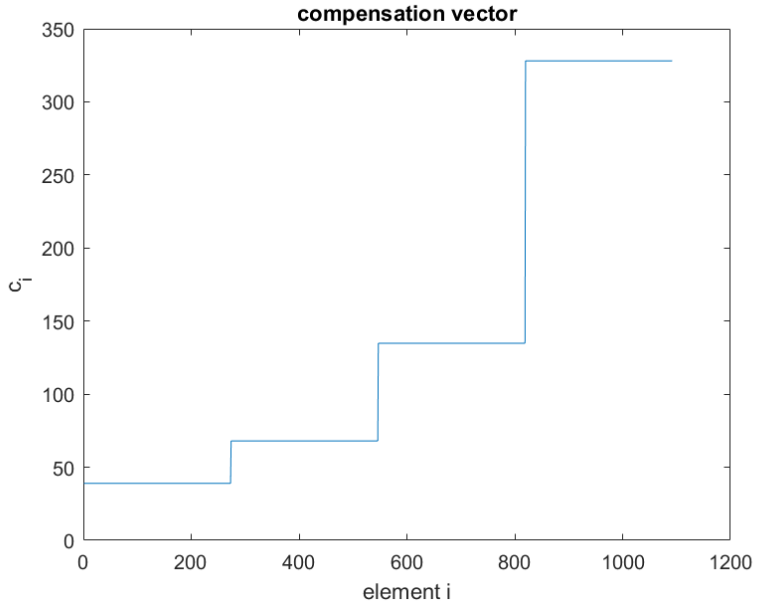

Figure 5.7: Example of the values of a compensation vector, where higher elements correspond to magnetic dipole locations closer to the sensor array.

One last adaptation to the minimisation problem is that only positive elements of **s** are allowed, since the basis already contains magnetic dipole moments in both positive and negative directions. An alternative would be to store only magnetic dipole fields of magnetic dipoles pointing in positive x, y, and z-directions in $\mathbf{\Psi}$. However, the optimisation algorithm provided better results with the restriction that each s_i must be larger than or equal to zero. The resulting minimisation problem is:

$$\begin{aligned} \text{Minimise} \quad & ||\mathbf{c}\odot\mathbf{s}||_1 \\ \text{s.t.} \quad & ||\mathbf{y}-\mathbf{\Theta s}||_2 \leq \epsilon; \\ & s_i \geq 0 \quad \forall i. \end{aligned} \tag{5.9}$$

This problem is convex, as proved in Section 4.4.3, so it can be solved using efficient, existing convex optimisation algorithms. This implementation uses the CVX package [14, 15] for MATLAB with solver SDPT3, version 4.0 [33, 34] as explained in Section 4.5.

Based on the noise with standard deviation σ as described in Section 5.3.3, an appropriate value for the error bound ϵ is chosen. The random vector $\mathbf{X} := \mathbf{y} - \boldsymbol{\Theta}\mathbf{s}$ has uncorrelated elements with zero mean and variance σ^2. For a choice of ϵ, it is important to know the characteristic of $||\tilde{\mathbf{X}}||_2$.

Scaling $\mathbf{X}$ results in $\mathbf{X} = \sigma\tilde{\mathbf{X}}$, where $\tilde{X}_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)$. Now, note that $||\tilde{\mathbf{X}}||_2 = \sqrt{\sum_{i=1}^{p} \tilde{X}_i^2}$ is distributed according to the chi distribution: $||\tilde{\mathbf{X}}||_2 \sim \chi_p$. This leads to the following mean and variance for $||\tilde{\mathbf{X}}||_2$:

$$\mathrm{E}||\mathbf{X}||_2 = \mathrm{E}\left[\sigma||\tilde{\mathbf{X}}||_2\right] = \sigma\mathrm{E}||\tilde{\mathbf{X}}||_2 = \sigma\sqrt{2}\frac{\Gamma((p+1)/2)}{\Gamma(p/2)}; \tag{5.10}$$

$$\mathrm{Var}||\mathbf{X}||_2 = \mathrm{Var}\left[\sigma||\tilde{\mathbf{X}}||_2\right] = \sigma^2\mathrm{Var}||\tilde{\mathbf{X}}||_2 = \sigma^2(p - \mathrm{E}||\tilde{\mathbf{X}}||_2^2). \tag{5.11}$$

The gamma function fraction in Eq. (5.10) can be approximated using Stirling's gamma function approximation [32, p.148]:

$$\sqrt{2}\frac{\Gamma((p+1)/2)}{\Gamma(p/2)} = \sqrt{p}\left(1 - \frac{1}{4(p+1)} + \mathcal{O}\left(\frac{1}{p^2}\right)\right). \tag{5.12}$$

From this, the following approximate estimated value and variance are obtained:

$$\mathrm{E}||\mathbf{X}||_2 \approx \sigma\sqrt{p}\left(1 - \frac{1}{4(p+1)}\right), \tag{5.13}$$

$$\begin{aligned}\mathrm{Var}||\mathbf{X}||_2 &\approx \sigma^2\left(p - p\left(1 - \frac{1}{2(p+1)} + \frac{1}{16(p+1)^2}\right)\right)\\ &= \sigma^2 p\left(\frac{1}{2(p+1)} - \frac{1}{16(p+1)^2}\right)\\ &= \sigma^2 p\frac{8p+7}{16(p+1)^2}.\end{aligned} \tag{5.14}$$

The chi distribution approaches the shape of a normal distribution for small values of p already. In this application, p is never chosen very small. Therefore, just as for the normal distribution, it is assumed that more than 95 percent of values lie within the two standard deviations interval:

$$P(\mathrm{E}||\mathbf{X}||_2 - 2\sqrt{\mathrm{Var}||\mathbf{X}||_2} \le ||\mathbf{X}||_2 \le \mathrm{E}||\mathbf{X}||_2 + 2\sqrt{\mathrm{Var}||\mathbf{X}||_2}) > 0.95. \tag{5.15}$$

The upper bound of this interval is

$$\begin{aligned}\mathrm{E}||\mathbf{X}||_2 + 2\sqrt{\mathrm{Var}||\mathbf{X}||_2} &\approx \sigma\sqrt{p}\left(1 - \frac{1}{4(p+1)} + 2\frac{\sqrt{8p+7}}{4(p+1)}\right)\\ &= \sigma\sqrt{p}\left(1 + \frac{2\sqrt{8p+7}-1}{4(p+1)}\right)\\ &\approx \sigma\sqrt{p}\left(1 + \frac{2\sqrt{2}}{2\sqrt{p}}\right)\\ &= \sigma\left(\sqrt{p} + \sqrt{2}\right),\end{aligned} \tag{5.16}$$

for reasonably large p. Hence, $\epsilon = \sigma\left(\sqrt{p} + \sqrt{2}\right)$ is chosen as error upper bound.

5.3.5. Classification: estimating magnetic dipole location and moment

To estimate the magnetic dipole's location and magnetic moment, weighted averages are taken. The sparse vector elements are used as weights to average over the initialisation locations and moments. For the magnetic moment and location, the n elements of $\mathbf{s}$ can be taken as weights:

$$\hat{\mathbf{x}} = \sum_{i=1}^{n} s_i \mathbf{x}_i; \tag{5.17}$$

$$\hat{\mathbf{m}} = \sum_{i=1}^{n} s_i \mathbf{m}_i, \tag{5.18}$$

where $\mathbf{x}_i$ and $\mathbf{m}_i$ are the location and moment corresponding to magnetic dipole i in the initialisation, respectively. Fig. 5.3 has illustrated this method of averaging as well.

A test was done to confirm in which situations the assumption can be made that the field of an 'averaged magnetic dipole' is equal to the sum of separate magnetic dipole fields. In the simulated test, four random magnetic dipoles were placed at the corners of a square with each time a different distance between them. The combined field of these dipoles was computed, as well as the field of the single dipole with location and moment calculated as in Eq. (5.17). The fields were calculated at different heights: the further away from the source(s), the better the separate dipole fields blend together to one field. The fields were calculated in a 40 cm by 40 cm grid. Each of the four magnetic dipoles was assigned three magnetic moment components by drawing from a uniform distribution between 0 and 1 ($\mathrm{A\,m^2}$). That gives each of the 'average moment' components a value between 0 and 4 $\mathrm{A\,m^2}$, which results in a moment magnitude in the order of a few $\mathrm{A\,m^2}$.

The results are shown in Table 5.1. Indeed, the error between the separate and combined field measurements drops when height increases or when grid distance decreases. For reference, the field components and a cross section of those at 0.1 m height, with a grid distance of 0.1 m, are shown in Fig. 5.8 and Fig. 5.9. At the same height, for 0.02 m grid distance, the same plots are shown in Fig. 5.10 and Fig. 5.11. The fields for 0.1 m grid distance are almost incomparable (RMSE of $2.8 \cdot 10^4$ nT). However, the fields for 0.02 m grid distance with an average RMSE of $3.1 \cdot 10^3$ nT look very similar.

height [m] / grid distance [m]	0.02	0.05	0.1
0.1	$3.1 \cdot 10^3$	$1.0 \cdot 10^4$	$2.8 \cdot 10^4$
0.2	$2.2 \cdot 10^2$	$1.1 \cdot 10^3$	$2.5 \cdot 10^3$
0.3	76	$2.2 \cdot 10^2$	$5.6 \cdot 10^2$

Table 5.1: Average RMSE of the magnetic field in nT in a 0.4 by 0.4 m grid. The error is calculated over four separate magnetic dipoles with random moments placed in a square and one averaged magnetic dipole. The RMSE values are averaged over 10 random magnetic dipole moment allocations.

The question is now whether this grid discretisation error is significant in the optimisation. To answer this, we must compare the RMSE values discussed here to the error bound that was derived in Section 5.3.4. Which term is dominant depends completely on the magnitude of the sensor noise and the magnitude of the magnetic moment of the dipole. If sensor noise is large compared to the signal strength, then this will be the dominant term. The tests described here were conducted using magnetic moments in the order of a few $\mathrm{A\,m^2}$. When measuring magnetic moments of different magnitude, one needs to scale the expected RMSE accordingly.

The feasible region in the optimisation problem Eq. (5.9) is equivalent to

$$\left\| \mathbf{y} - \boldsymbol{\Theta}\mathbf{s} \right\|_2^2 \leq \epsilon^2 = \sigma^2 (p + 2\sqrt{2p} + 2), \tag{5.19}$$

where p is the number of measurements, as argued in Section 5.3.4. We must add to that the expected average RMSE due to grid discretisation σ_{grid} times the number of measurements p:

$$\left\| \mathbf{y} - \boldsymbol{\Theta}\mathbf{s} \right\|_2^2 \leq \sigma^2 (p + 2\sqrt{2p} + 2) + \sigma_{\text{grid}}^2 p. \tag{5.20}$$

Hence, the adjusted value for ϵ is now:

$$\epsilon = \sqrt{\sigma^2(p + 2\sqrt{2p} + 2) + \sigma_{\text{grid}}^2 p}. \tag{5.21}$$

For a large number of measurements, it is clear that σ and σ_{grid} scale equally with p. Therefore, the RMSE values reported here should be compared directly to the sensor noise standard deviation σ to check whether discretisation errors have any influence.

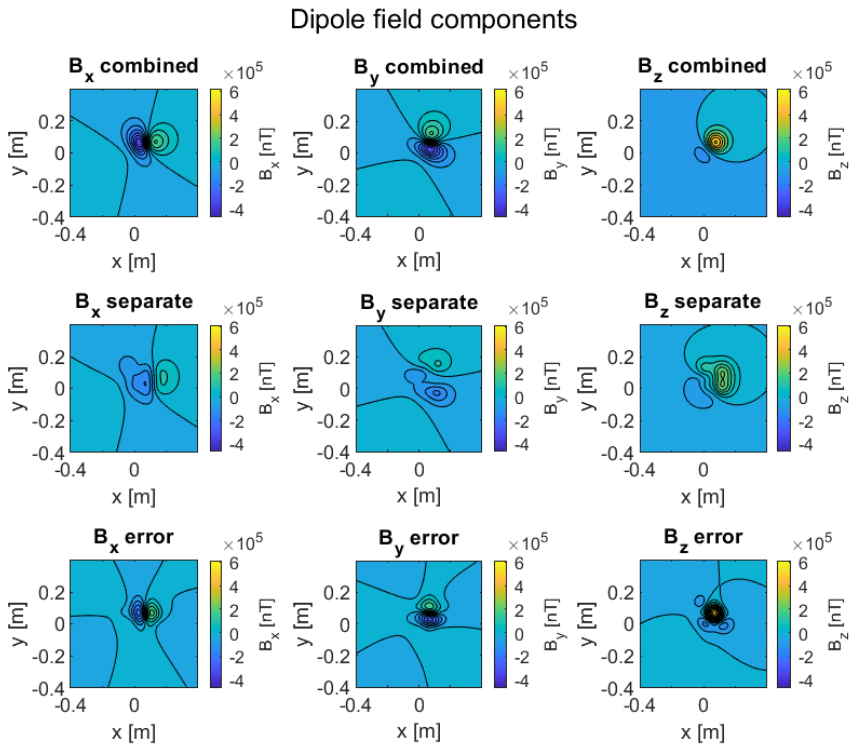

Figure 5.8: Magnetic field components of one averaged magnetic dipole (combined), of the four magnetic dipoles placed in a square, 0.1 m apart, that form the average magnetic dipole (separate), and the difference between them (error). Measurements are done at a height of 0.1 m.

5.4. Extended algorithm using iterations

To improve the estimates produced by the algorithm, it can be applied iteratively. This has been implemented by shrinking the search area. The algorithm described in the previous section is ran once to find an initial estimate for location (and moment). Based on this estimate, a new basis is constructed, as in Section 5.3.1. Instead of using the full search area to choose initialisation locations from, a smaller volume was used, in the form of a cube. The centre of this cube is the estimated location from the first iteration. Using this new basis, the same steps as in Section 5.3 are followed, using n new sensors, and a new estimate is produced. This process continues a predefined number of times. The search cube starts with length equal to ten times the distances between the initial magnetic dipoles, hence containing at most 10^3 dipole locations. Every iteration, the search radius shrinks by the distance between the initial magnetic dipoles (grid distance), eliminating on each side of the cube one row of grid points that were considered in the previous iteration.

Convergence properties of this algorithm are not proved. Instead, it is based on a heuristic approach, and developed by trial and error. For instance, a sphere-shaped shrinking search space was tried, but results were worse than for the box-shaped space. Also various staring sizes and shrinking parameters have been tested, resulting in the iteration strategy described above. With these search tactics, good results have been produced, as will be described in Section 6.1.4.

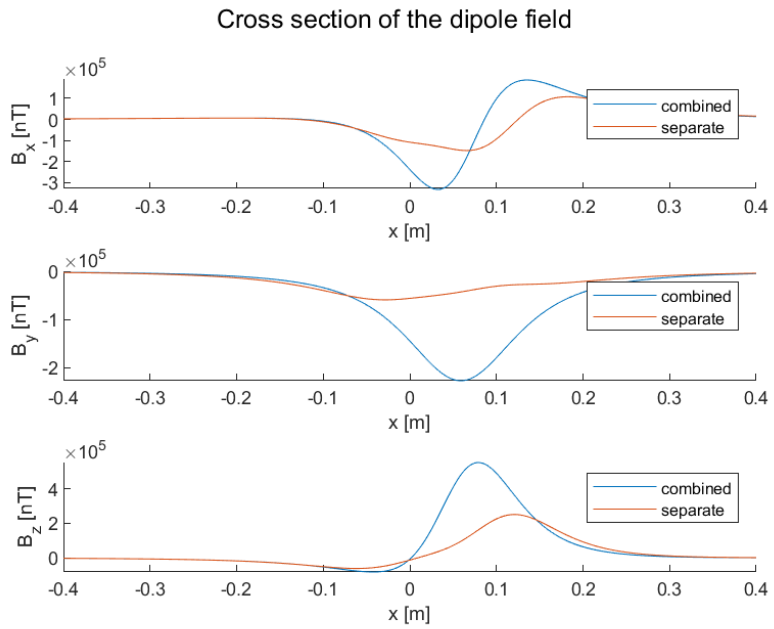

Figure 5.9: Magnetic field components at a cross section parallel to the x-axis of one averaged magnetic dipole (combined), and of the four magnetic dipoles placed in a square, 0.1 m apart, that form the average magnetic dipole (separate). Measurements are done at a height of 0.1 m.

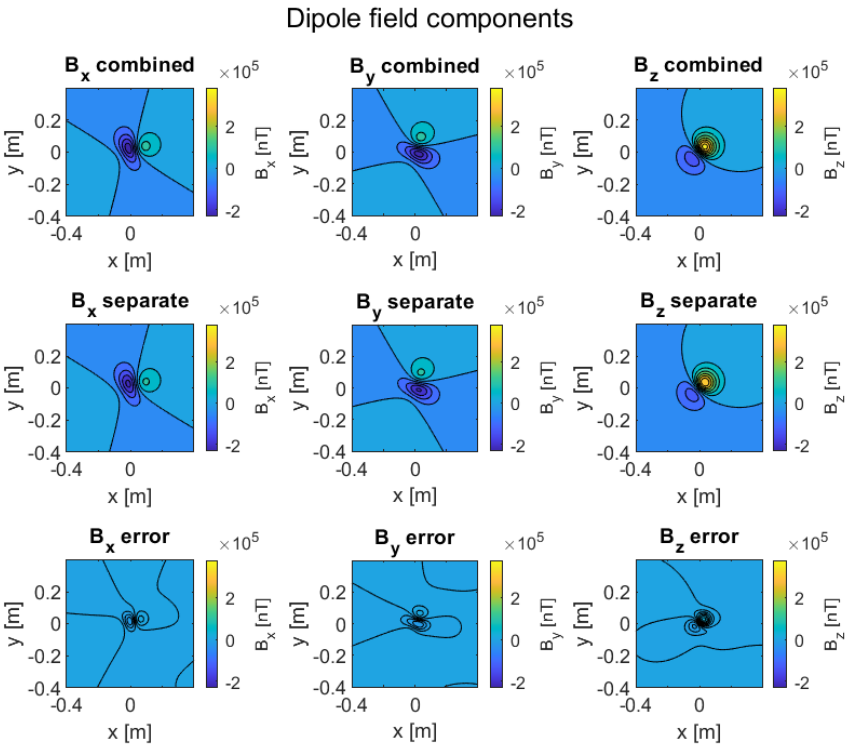

Figure 5.10: Magnetic field components of one averaged magnetic dipole (combined), of the four magnetic dipoles placed in a square, 0.02 m apart, that form the average magnetic dipole (separate), and the difference between them (error). Measurements are done at a height of 0.1 m.

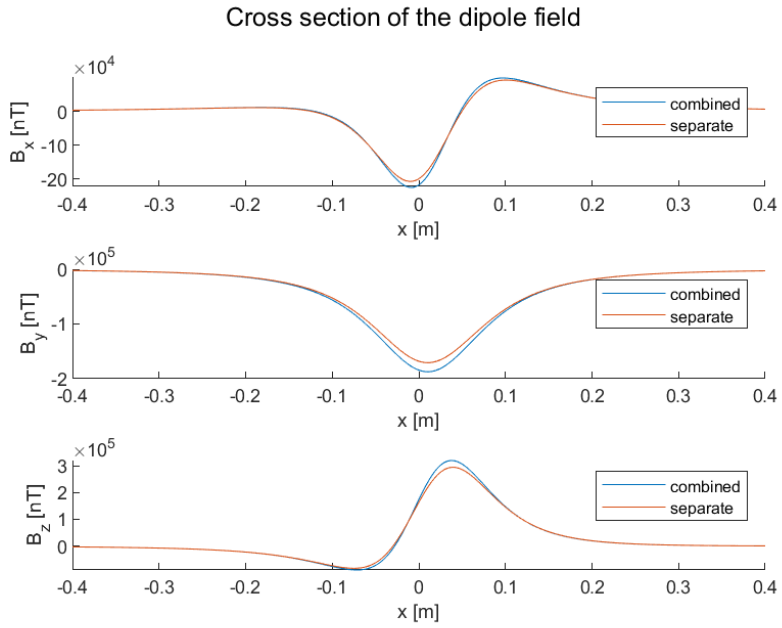

Figure 5.11: Magnetic field components at a cross section parallel to the x-axis of one averaged magnetic dipole (combined), and of the four magnetic dipoles placed in a square, 0.02 m apart, that form the average magnetic dipole (separate). Measurements are done at a height of 0.1 m.

5.5. Alternative field estimation algorithm using SVD-basis

In the main algorithm, the focus is on classifying a magnetic dipole (estimating its location and moment). A field estimate follows directly from the system $\mathbf{x} = \mathbf{\Psi s}$, but just as an additional result. If the goal is not classification but field approximation, an alternative algorithm might work better.

In this version, the basis is initialised as before, but then immediately replaced by a reduced SVD-basis $\mathbf{\Psi}_r$. The exact construction has been described in Section 5.3.2. There are two choices for how to reduce this basis: one can use a predefined rank r, for example such that the reduced basis contains x percent of energy of the original basis. Or one can use the same reduced basis as in Section 5.3.2, where the rank r is set equal to the number of measurements p. Then, the QR pivoting method is always applicable using this same basis, and it does not need to be reduced again.

In either case, the idea is that noise in measurements has such high frequency, that it cannot be described by the first r modes in $\mathbf{\Psi}_r$ that have a relatively low frequency. The frequency of the modes from the SVD-basis increases, as is visualised in Fig. 5.12.

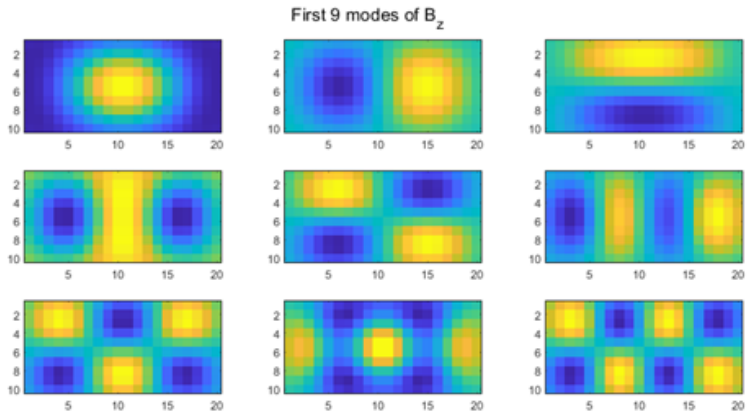

Figure 5.12: Visualisation of the z component of the first 9 modes in $\mathbf{U}$ obtained from applying a SVD to the basis $\mathbf{\Psi}$. The elements of the respective columns of $\mathbf{U}$ are reshaped to give a 2D interpretation.

6

Results and discussion

This chapter presents the results of the implemented localisation algorithm. First, the algorithm is tested and analysed using several simulations. The results of these are presented in Section 6.1. Additionally, the algorithm has been verified using measurement data. This analysis is given in Section 6.2

6.1. Simulation results

In this section, several simulation results of the implemented localisation algorithm are given. The influence of important variables and parameters of the algorithm is analysed: the noise in the sensor measurements, the location of the magnetic dipole that needs to be found, the number of sensors available, whether sensors are chosen randomly or using QR pivoting, different minimisation problems, and number of iterations.

6.1.1. Noise

In general, the larger the noise is compared to the measurement, the harder it is to reconstruct the noise-free field. It is expected that this also holds for this algorithm: decreasing the noise level should make results better and better. Indeed, this is the case, as shown in Fig. 6.1. Here, a histogram of location errors is shown for four different noise levels. These noise levels give rise to a signal-to-noise ratio, that is defined as

$$SNR := \frac{A_{\text{signal}}^2}{\sigma^2}, \tag{6.1}$$

where A_{signal} is the RMS amplitude of the signal (measurement) and σ is the noise standard deviation. From large to small, these ratios in the figures are 24.0, 4.8, 2.4, and 1.2. As expected, results get worse when lowering the signal-to-noise ratio. Still, the error is not too large for a ratio of 1.2, considering that the search area is 60 cm x 30 cm x 20 cm, against an average location error of about 10 cm. This means that although the location cannot be precisely estimated with such noise levels, the algorithm can still point in the right direction. The similar shape of all histograms can also be explained. One could expect that the x-coordinate of the estimated locations forms a distribution that is shaped somewhat like the Gaussian distribution, centred around the true x-coordinate. Hence, the error in the x-coordinate would be Gaussian-shaped around 0 m. If this is the case for the x, y and z-coordinates of the estimate, then the resulting location error has approximately as a chi distribution with parameter 3 (because there are three dimensions). Indeed, this distribution is similarly shaped to the histograms seen in Fig. 6.1.

For 150 runs, individual location errors for a range of signal-to-noise ratios have also been plotted against these ratios in Fig. 6.2. It is clear that the average error, as well as the maximum encountered error, decreases for larger signal-to-noise ratios, as expected. An offset of multiple centimetres is visible, even for larger signal-to-noise ratios. It turns out that this is largely caused by an error in the z-coordinate estimate of the magnetic dipole. In Fig. 6.3, the same scatter plot is produced, but now for just horizontal location error on the y-axis. Indeed, the errors now have a range that extends to 0 m.

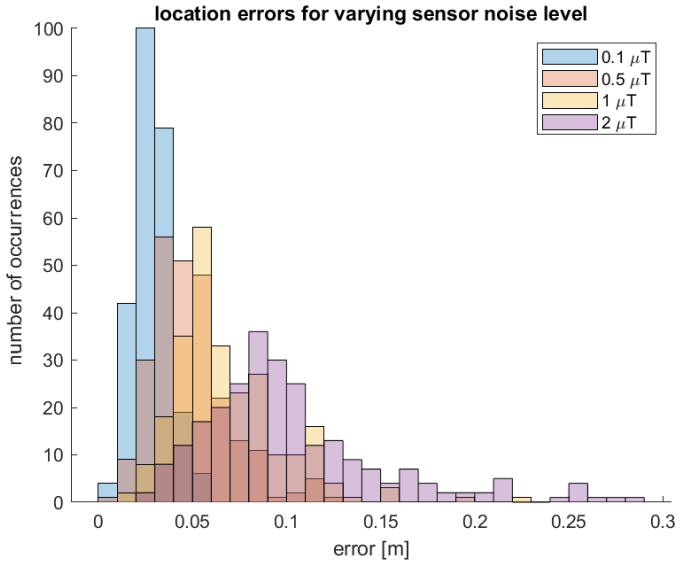

Figure 6.1: Histogram of location errors for 150 runs with 16 chosen sensors using QR pivoting, comparing four different noise levels (indicated by standard deviation). The corresponding average signal-to-noise ratios are, from large to small (blue to purple), 24.0, 4.8, 2.4, and 1.2.

Figure 6.2: Scatter plot of location errors versus average signal-to-noise ratio, for 150 runs with 8 chosen sensors using QR pivoting.

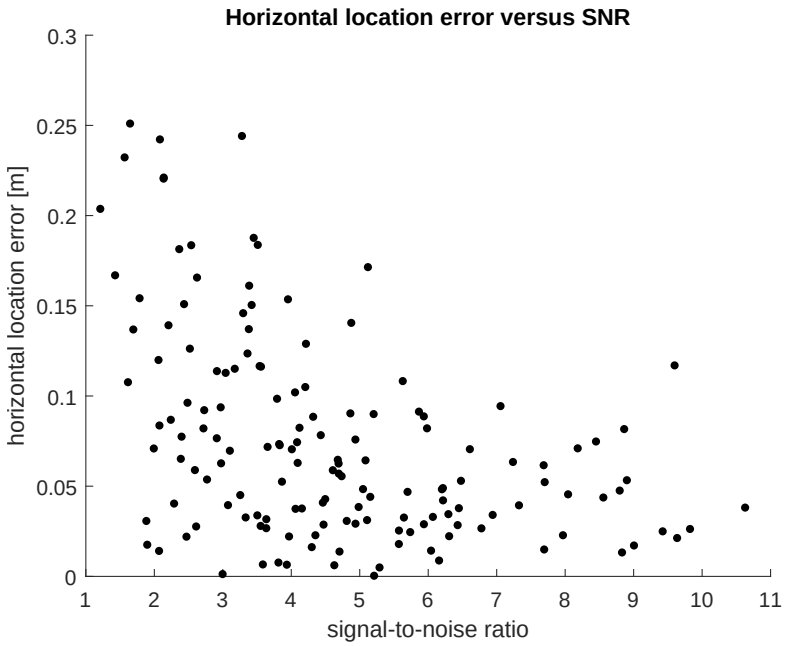

Figure 6.3: Scatter plot of horizontal location errors versus average signal-to-noise ratio, for 150 runs with 8 chosen sensors using QR pivoting.

6.1.2. Number of sensors and sensor choice

From the explanation in Section 5.3.2, an optimal QR sensor placement is expected to perform better than a random sensor choice. Especially for a small number of measurements p, random sensor choice can be poor. If just a few sensors are placed, there is a reasonable chance that those sensors are all located considerably far away from the location of the magnetic dipole. If this is the case, the measured signal is small compared to noise. An example of a particularly bad reconstruction due to a poor choice of sensors is shown in Fig. 6.4. The error is given as NRMSE: normalised root mean squared error. It is the root of the mean of all field errors, divided by the range of the original signal: its maximum value minus its minimum value. For a vector $\mathbf{x}$ of length N and an estimate $\hat{\mathbf{x}}$ of it:

$$\mathrm{NRMSE}(\mathbf{x},\hat{\mathbf{x}}) = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\hat{x}_i)^2}}{\max(\mathbf{x})-\min(\mathbf{x})}. \tag{6.2}$$

The magnitude of the magnetic moment of the particular magnetic dipole in this example is $0.88\,\mathrm{A\,m^2}$. Here, the sensors are all located away from the magnetic dipole location in the top right corner, measuring only small signals.

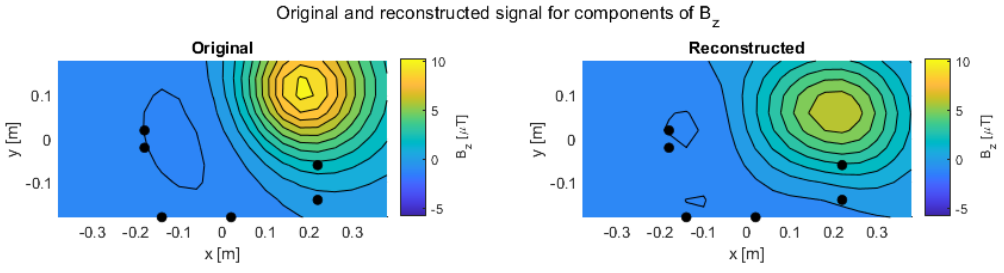

Figure 6.4: Reconstruction of the z-component of the magnetic field using 6 randomly chosen sensors, indicated by the black dots. The reconstruction has NRMSE = 0.12.

In the QR placement method, on the other end, sensors are chosen such that an optimal low-rank approximation can be made. Practically, this means the chosen sensors are spread over the whole sensor array, greatly reducing the probability of not picking up strong signals.

The expectations are tested by comparing location errors, moment angle errors and moment magnitude errors, for 4, 8, and 16 sensors, and for random sensor choice and placement using QR pivoting. To show the range of results for both the QR pivoting method and random sensor placement, 150 test runs were executed for both methods, for 4, 8, and 16 available sensors. For each of the 150 runs, a different magnetic dipole is placed randomly within the search area as seen in Fig. 5.2. Furthermore, its moment has a random orientation and random magnitude between 5 and 15 $\mathrm{A\,m^2}$. Sensor noise was set to 0.5 µT. In the case of the QR pivoting method, the chosen sensors are in the same position for every run, since the basis remains unchanged between runs. However, for random placement, a different subset of sensors is chosen in each run.

In Fig. 6.5, random sensor placement is compared to the QR pivoting method for a choice of 8 sensors. The errors in location, moment angle, and moment magnitude are shown in error distributions. The moment magnitude errors are given as relative numbers: a relative offset of -0.2 means that it is 80 percent of the true size, and a relative offset of 0.1 means it is 110 percent of its true size. From the error distributions, it becomes clear that the QR pivoting method performs slightly better. This is best visible for the moment angle error. All distributions corresponding to QR pivoting are centred around a smaller error than that of the random choice method. Also, the distributions corresponding to QR pivoting contain less large outliers than for random choice. The latter shows for instance an outlier of approximately 30 cm in the location error. These results are in line with the expectations for both methods.

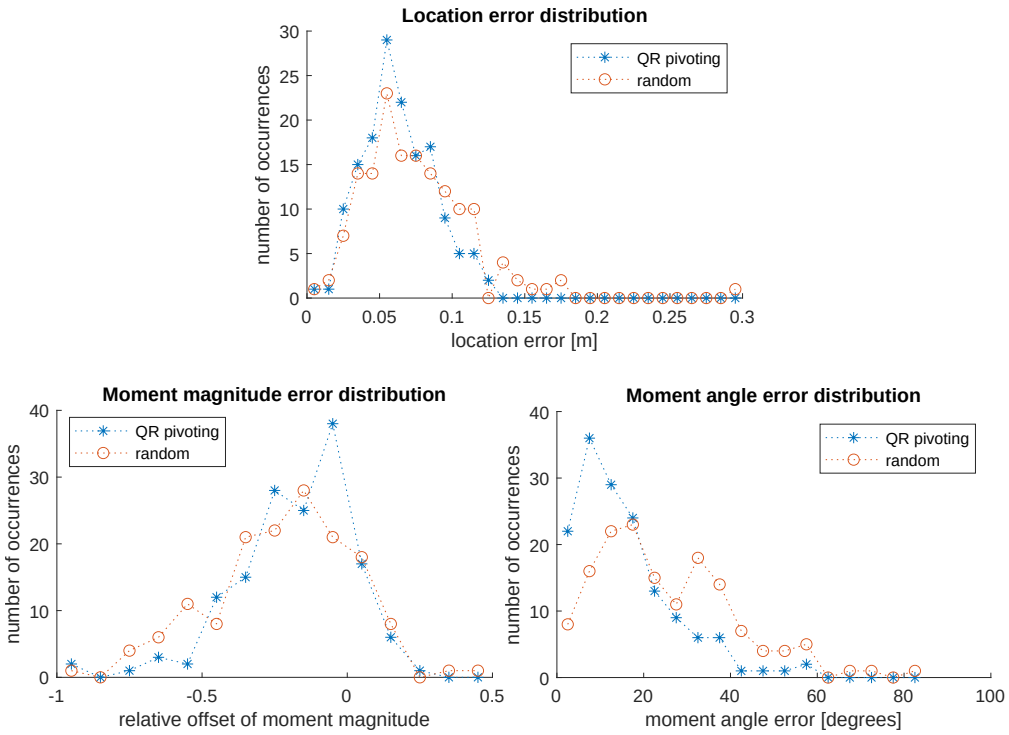

Figure 6.5: Distributions of errors for 150 runs with 8 sensors, comparing a random placement and QR placement method.

When the number of sensors is increased, one would expect a shift of these error distributions towards zero. Furthermore, the method of random sensor placement is expected to produce less outliers. For example, 16 sensors should be able to cover the area in many cases. These expectations are indeed met when running the algorithm for 4, 8, and 16 sensors with both methods, of which the results are displayed in Fig. 6.6 and Fig. 6.7. Going from 4 to 8 sensors, both the results for random and QR sensor choice improve visibly. When looking at the random choice location errors, it can be seen that indeed all large outliers disappear. When doubling the number of sensors again to 16 sensors, especially the results for random sensors choice are improved. Results for QR pivoting improve slightly, but not as much as for random placement. This is expected, since choosing sensors optimally already makes sure that a terrible sensor choice can not occur.

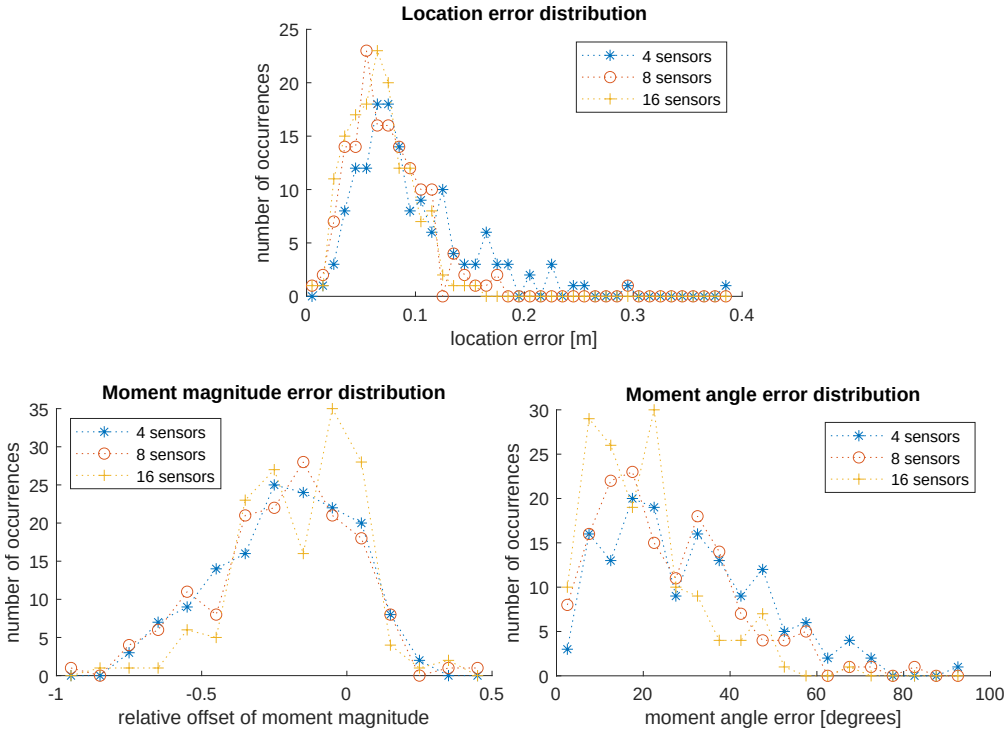

Figure 6.6: Distributions of errors for 150 runs with random sensor choice, comparing 4, 8, and 16 sensors.

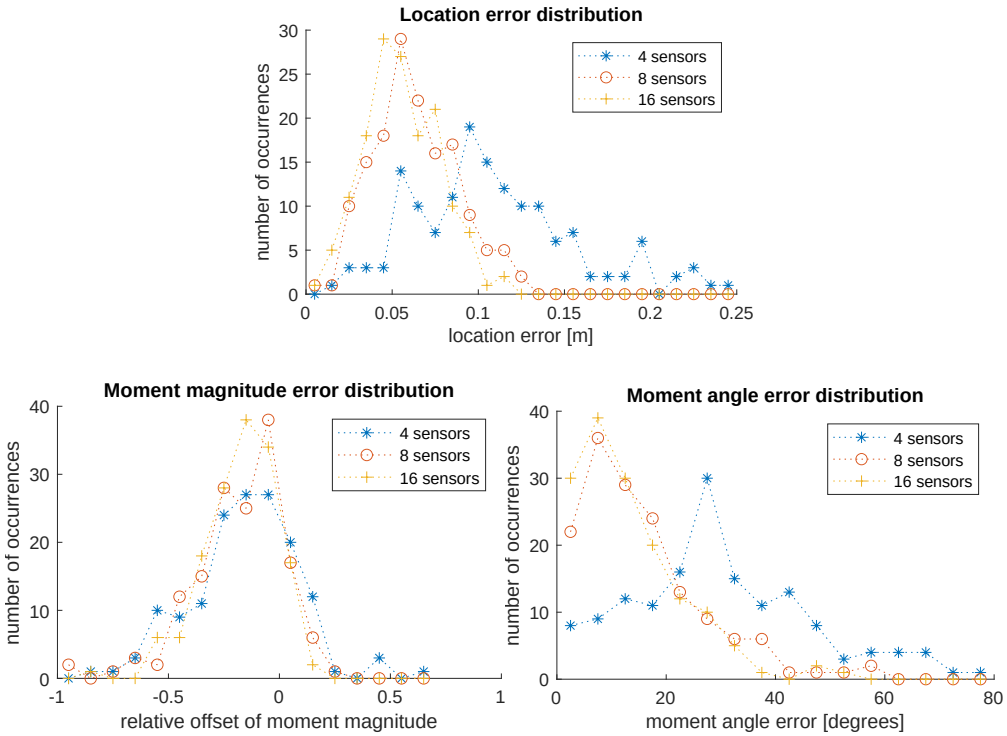

Figure 6.7: Distributions of errors for 150 runs with QR pivoting sensor choice, comparing 4, 8, and 16 sensors.

6.1.3. True magnetic dipole location

To see if the true magnetic dipole location influences the results, the scatter plot in Fig. 6.2 has been reproduced to show the influence of three parameters: vertical distance of the true magnetic dipole to the sensor array, horizontal distance of the true magnetic dipole to the origin, and distance of the true magnetic dipole to the closest initialisation grid point.

First, the vertical distance to the sensor array. The goal of the penalty vector introduced in Section 5.3.4 is to avoid the preference for choosing magnetic dipoles too close to the sensor array. Therefore, it is expected that for similar signal-to-noise ratio, there are similar location errors for magnetic dipoles close by and magnetic dipoles far away from the sensor array.

In Fig. 6.8, the scatter plot is coloured by vertical distance to the sensor array. The first thing appearing here, is the larger signal-to-noise ratio for magnetic dipoles closer to the sensor array. This, of course, makes sense: the closer to the array, the larger the field measurements for equal noise. Now, is it possible to see if the location errors are similar for each dipole height? Looking at the coloured dots along a vertical line, so for equal signal-to-noise ratio, there is no clear concentration of blue or yellow dots on the lower or higher end of the line. This indicates that adding the penalty vector to the minimisation problem successfully eliminated the preference for magnetic dipoles close to the sensor array.

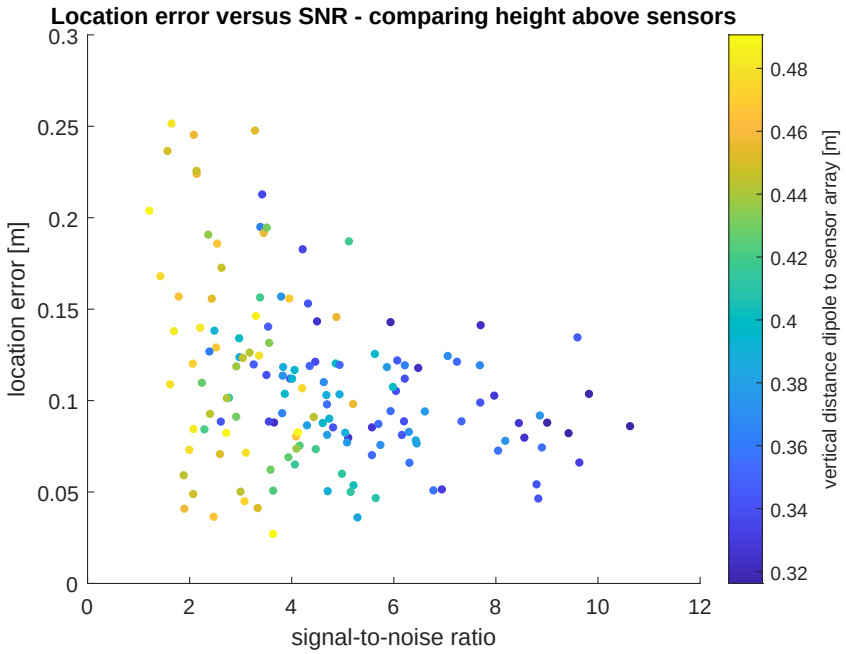

Figure 6.8: Scatter plot of location errors versus average signal-to-noise ratio, for 150 runs with 8 chosen sensors using QR pivoting. Colours indicate the vertical distance between the true magnetic dipole location and the sensor array.

Next, the focus is on horizontal location of the true magnetic dipole. It could be the case that the algorithm performs better or worse for magnetic dipoles near the edges of the search area. This can be quantified by taking the horizontal distance from the true magnetic dipole location to the origin, which is the middle of the search area. The larger this distance, the closer the true magnetic dipole is to one of the edges of the search area.

The coloured results are shown in Fig. 6.9. It is again clear that on a vertical line of equal signal-to-noise ratios, there is no concentration of either colour on the top or bottom half of the graph, which means that the algorithm does not perform better or worse for magnetic dipoles closer to the edges.

The last considered factor is how close the true magnetic dipole is to one of the initialisation locations (grid points). The results are shown in Fig. 6.10. Again, no clear distinction in location error is visible between true magnetic dipoles close to one of the grid points and magnetic dipoles that are more in between several different grid points.

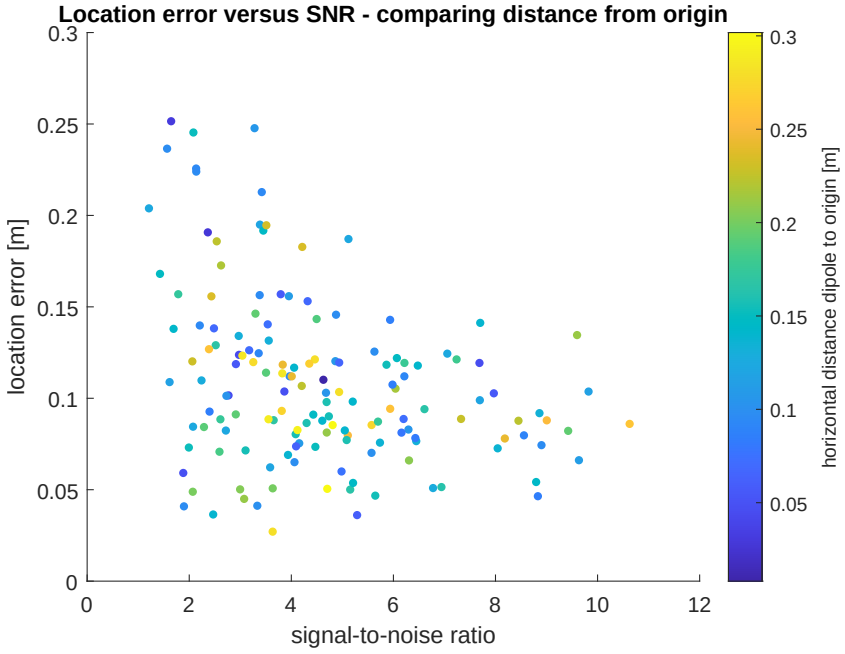

Figure 6.9: Scatter plot of location errors versus average signal-to-noise ratio, for 150 runs with 8 chosen sensors using QR pivoting. Colours indicate the horizontal distance between the true magnetic dipole location and the origin.

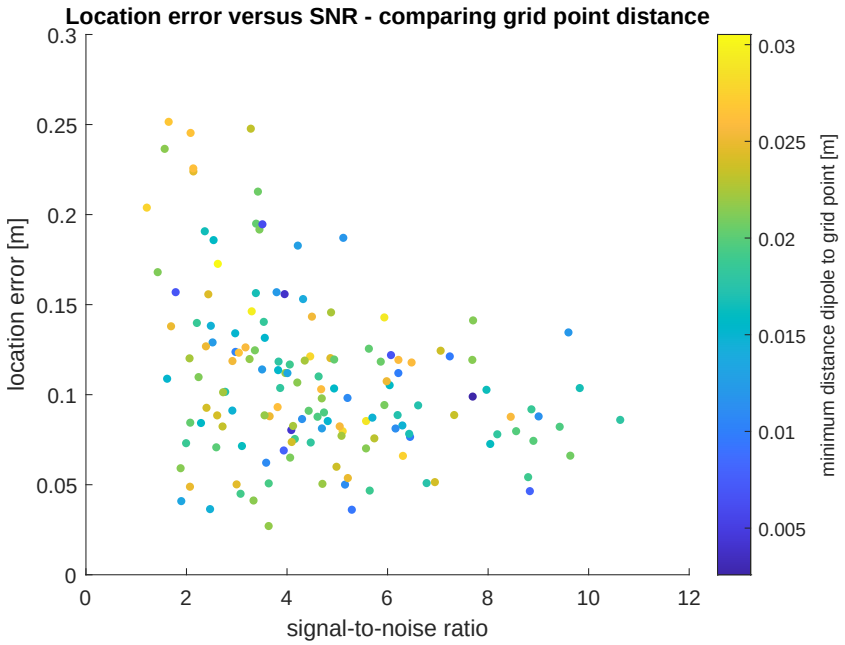

Figure 6.10: Scatter plot of location errors versus average signal-to-noise ratio, for 150 runs with 8 chosen sensors using QR pivoting. Colours indicate the distance between the true magnetic dipole location and the closest initialisation grid point.

6.1.4. Iterations

To illustrate how iterations help in finding the correct location, the distribution location and magnetic dipole moment errors of 150 runs has been plotted in Fig. 6.12. The realisations are made with 8 sensors that are chosen with QR pivoting, and noise with 0.5 µT standard deviation. The first iteration is completed using the full basis, over the full search area. In the second and third iteration the total length of the search box is made 4 cm smaller.

The average location error is gradually reduced, as expected. By eliminating the possibility to activate dipoles too far away from the estimated location, the estimate becomes more precise. Furthermore, new sensors are chosen using QR pivoting in every iteration, based on the smaller bases. The activated sensors concentrate towards the estimated location every iteration, to better describe the field originating from a magnetic dipole in that area. This movement is nicely visible over the three iterations in the location error distribution in Fig. 6.12.

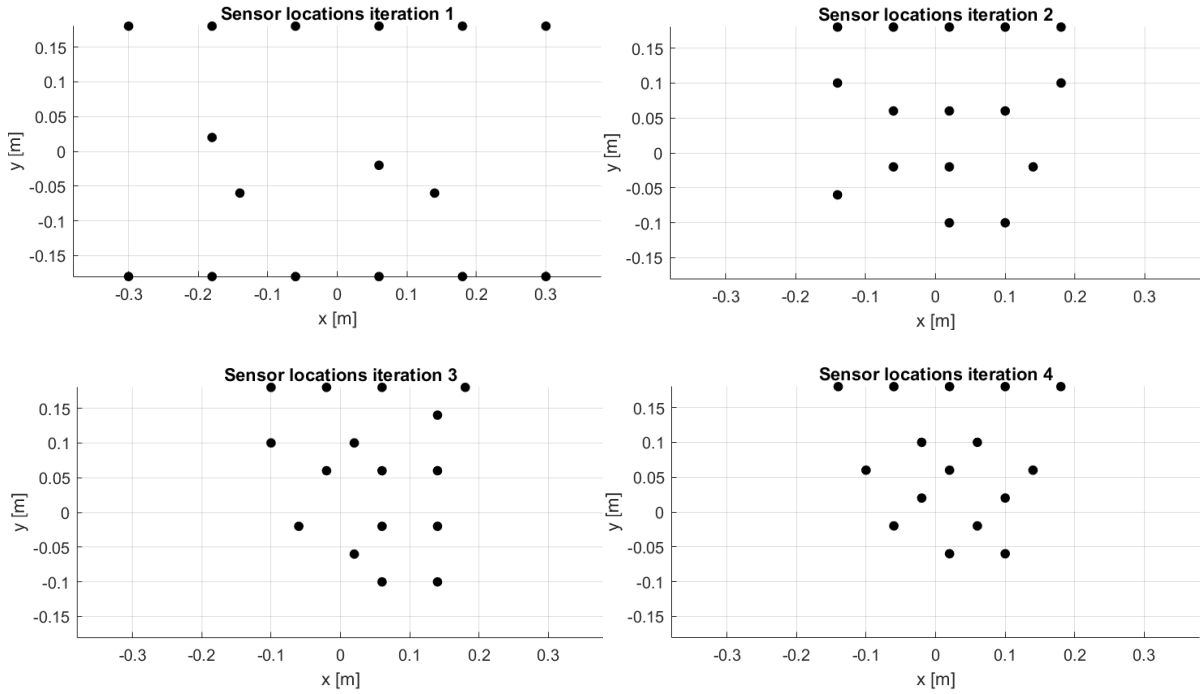

Figure 6.11: Horizontal coordinates of 16 chosen sensors using QR pivoting for four iterations. Here, the true magnetic dipole is located at approximately (x, y, z) = (0.03 m, 0.07 m, -0.12 m).

The shrinking of the search area is done gradually: although the estimate is already quite good on average after the first iterations, the number of times the estimate is relatively far away from the real location is not negligible. If the box is shrank too aggressively, the real magnetic dipole location might end up outside the search box, eliminating the chances of finding this true location. Therefore, to create a robust algorithm, only one layer of initial magnetic dipoles is removed on each side of the search box in every iteration.

On average, the moment magnitude is estimated really well, staying within 10 percent above or below the true value. It is interesting to note the structural error - although not too large - in moment angle. This error is by definition always positive, so it might match with the average structural error in location. Naturally, these influence each other: if the magnetic dipole location estimate contains an error, the algorithm must have compensated for this by rotating the estimated magnetic dipole bit, to still be able to match the resulting field with the obtained measurement.

Comparing to similar results for randomly chosen sensors, depicted in Fig. 6.13, it appears that much of the reason why these iterations improved results, is the better sensor choice each iteration. Now that the sensors are chosen randomly in each iteration, the location estimates do not improve as much as before. However, the moment magnitudes and angles are improving with each iteration. This indicates that some activated moments fall outside the search box after iterating, forcing the algorithm to choose moments from a smaller space around the previous location estimate. Hence, there is less room for creative activation of magnetic dipole moments over the full search space to match noisy measurements.

6.2. Verification using measurements

For a verification of the proposed algorithm, also real-world measurement data has been used. This section first describes the experimental setup and then present the results and a discussion.

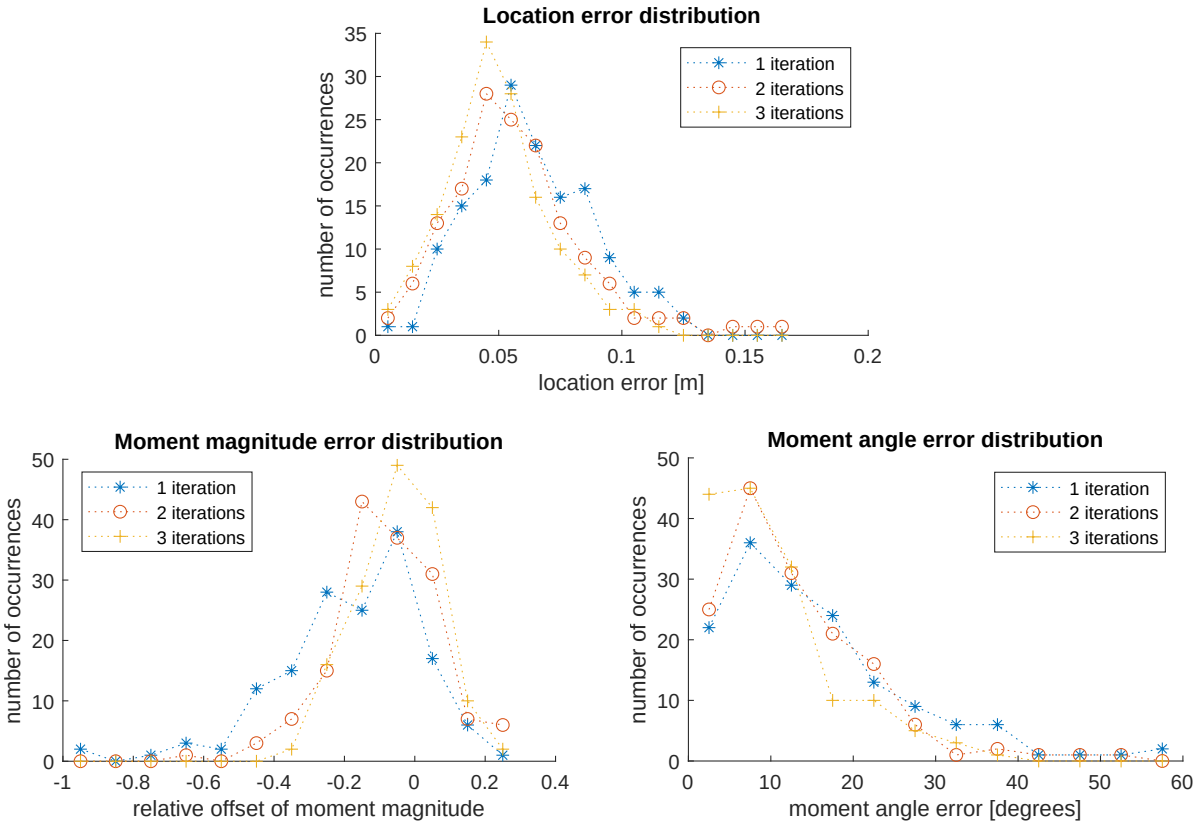

Figure 6.12: Error distributions for 150 runs, for 8 sensors chosen using QR pivoting, comparing 1, 2, and 3 iterations of the algorithm.

6.2.1. Experimental setup

The experimental setup is very similar to the one provided in the problem description in Section 5.1, and the illustration in Fig. 5.2. The measurement setup also consists of a horizontally placed sensor array, now placed at $z = 0.045$ m. It spans between $x = -0.38$ m and $x = 0.38$ m and between $y = -0.18$ m and $y = 0.18$ m with 0.04 m distance between sensors in both directions. This comes down to 200 sensors in total, each measuring all three magnetic field components. Since each field component is measured by a different physical sensor at a slightly different location, there can be a small offset in the sensor locations per component (a few mm). Sensor choice was simulated in this test using only the measurements of 10 chosen sensors, even though all data was collected.

A small magnet was moved over a grid in an S-shaped path. The grid was 9 grid positions wide and 7 grid positions long, from $x = -0.38$ m to $x = 0.38$ and from $y = -0.18$ to $y = -0.18$. The magnet that is used produces a field close to that of a perfect magnetic dipole, with a magnetic moment of approximately $0.05\,\mathrm{A\,m^2}$. This estimation is produced by running the algorithm developed in this thesis, and taking moment estimates where the location estimates were excellent (within 1 cm distance). The magnetic moment is not given as an input to the algorithm in any way - it is just a resulting output.

The sensor noise level has been estimated using two initial measurements of the background field. 200 sensors have each measured the three magnetic field components twice. If no sensor noise would be present, the two measurements would be equal. However, we assume that white Gaussian noise is added, with a normal distribution centred around 0, with a certain standard deviation σ. By calculating the difference between the 600 first and second measurements, one can derive the value of σ. Suppose the first measurement is $X_1 \sim \mathcal{N}(0,\sigma^2)$, and the second $X_2 \sim \mathcal{N}(0,\sigma^2)$. Then also by symmetry: $-X_2 \sim \mathcal{N}(0,\sigma^2)$. Hence:

$$X_1 - X_2 \sim \mathcal{N}(0, 2\sigma^2). \tag{6.3}$$

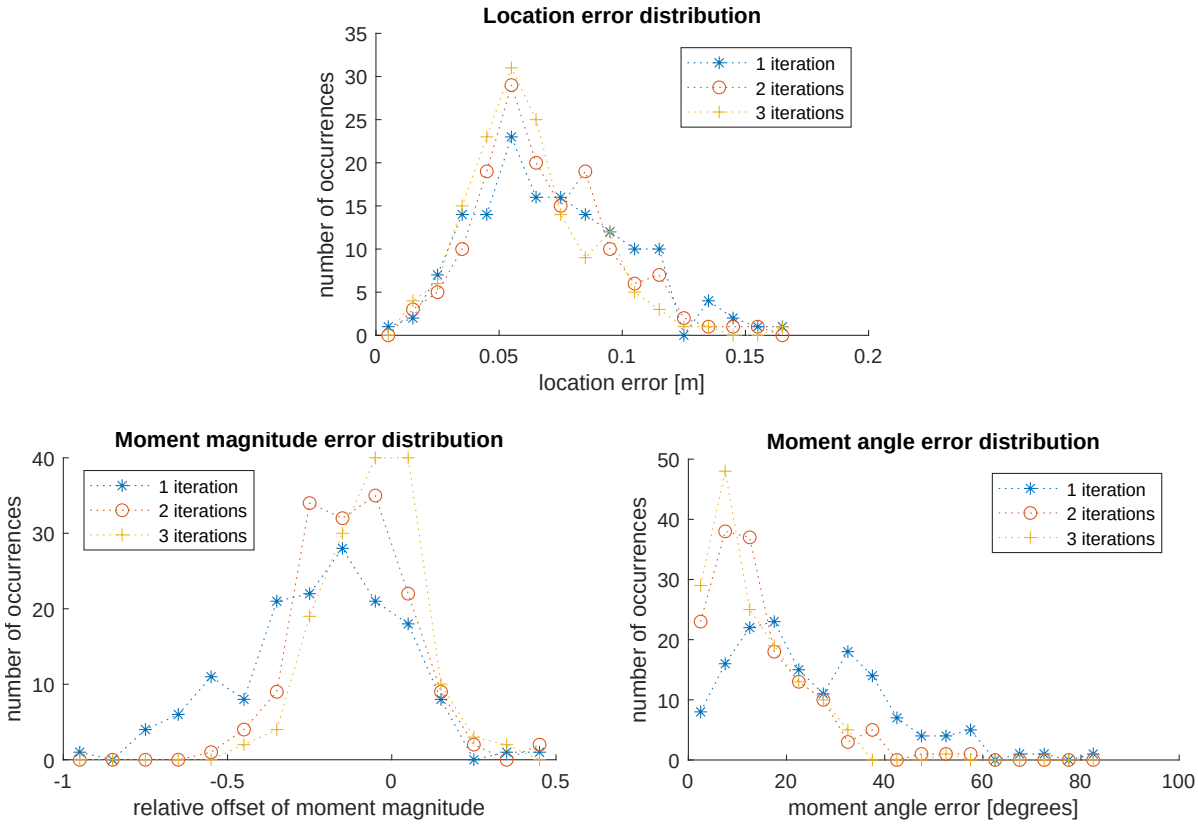

Figure 6.13: Error distributions for 150 runs, for 8 randomly chosen sensors, comparing 1, 2, and 3 iterations of the algorithm.

Therefore the following computation is used to determine the sensor noise standard deviation:

$$\sigma = \sqrt{\frac{1}{2}\mathrm{Var}(X_1 - X_2)}. \tag{6.4}$$

For the performed measurements, the computed value for σ is approximately equal to 21 nT.

For basis construction, a grid distance of 0.04 m was used. Following the discussion in Section 5.3.5, the grid discretisation error needs to be estimated. Since the magnetic dipole was held at 0.24 m from the grid, interpolation of the values in Table 5.1 gives a RMSE of around 600 nT. However, since the magnetic dipole moment here is about 40 times smaller, the scaled RMSE value is estimated to be between around 15 nT. Since we consider not just averaging within a horizontal plane, but averaging within a 3D cube, the RMSE value was increased somewhat to 25 nT. This choice was also the result of some testing of the algorithm, where this value gave the best performance. Here, neither the sensor noise nor the grid discretisation error is dominant in the optimisation. One option would be to decrease the grid distance in basis construction. However, this increases both size of stored matrices and computing power needed to optimise exponentially. Therefore, an adjusted value for ϵ is chosen according to Eq. (5.21), where σ_{grid} is taken equal to 25 nT.

6.2.2. Results

As mentioned before, the tests have been conducted using the measurements of 10 sensors at the same time. Both random and optimal sensor choice using QR pivoting has been applied. The results for 1 iteration of the algorithm are compared to the results for 3 iterations (only using QR pivoting). The magnetic moment was not measured or estimated independently from this algorithm, and therefore only location estimates and errors are discussed.

The resulting location estimates for random sensor choice are shown in Fig. 6.14, for 1 iteration with optimal sensor choice in Fig. 6.15, and for 3 iterations they are shown in Fig. 6.16. These plots show the difference between the true magnetic dipole location and the location estimate, in 3D and from above, following the location path of the measurements.

An interesting behaviour that stands out immediately is the fact that estimates are almost always placed a bit higher (smaller z) than they should. We perform ℓ_1 optimisation within error bounds, which means that every possibility will used to minimise the ℓ_1 norm term. A somewhat large feasible region is used to make sure the algorithm can also produce a good estimate when the data is noisier than average. This also means that when the data is not as noisy as anticipated, the space will be taken by the ℓ_1 minimisation. Normally that would result in higher placed magnetic dipoles, with a smaller magnetic moment, since only the moment count for the minimisation. However, we compensated for this using the vector $\mathbf{c}$. These results show that either the height of the magnetic dipole was not measured correctly (however, an error of more than 1 cm is highly unlikely), or the compensation vector does work exactly as planned. The latter is most likely, since the compensation term that scales with $\frac{1}{r^3}$ was a simplification.

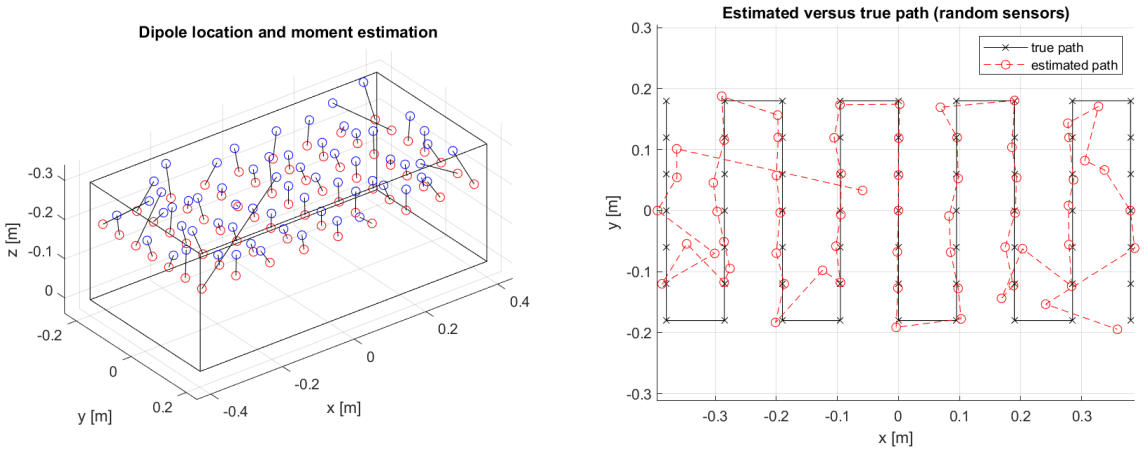

Figure 6.14: True locations of the magnetic dipole (red) and location estimate by the algorithm after 1 iteration (blue), shown in 3D and from a top view. Random sensor choice has been used.

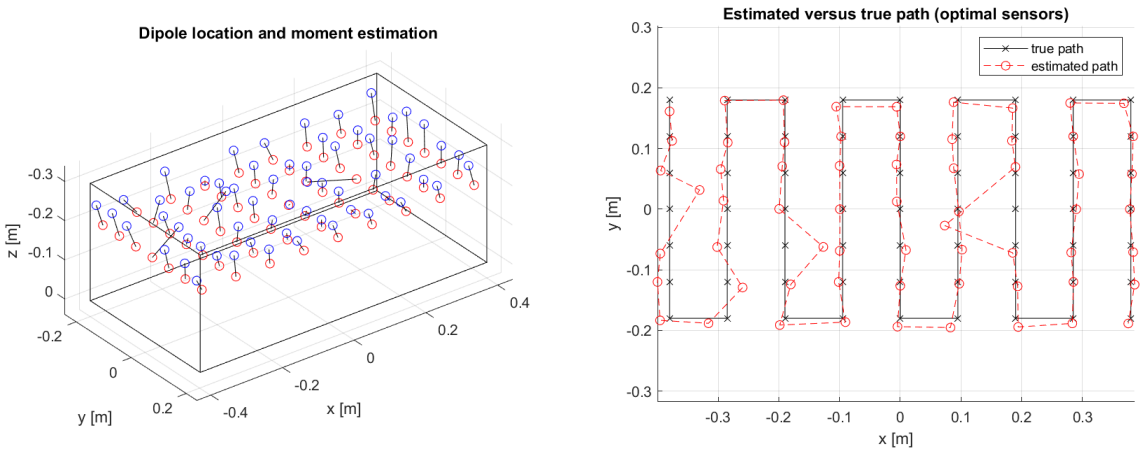

Figure 6.15: True locations of the magnetic dipole (red) and location estimate by the algorithm after 1 iteration (blue), shown in 3D and from a top view. Optimal sensor choice using QR pivoting has been used.

As expected from the numerical experiments, the algorithm including optimal sensor choice performs better than with random choice. In Fig. 6.14, it is clear that random choice may produce outliers, when sensors are chosen in such a poor pattern that reconstruction is almost impossible. This is more common when the magnetic dipole is located at the borders of the search area, since that leaves a larger area with a small signal. The performance of the algorithm with optimal sensor choice improves when 3 iterations are used. Indeed, when looking at the paths, some of the worse estimates for 1 iteration are improved when performing 2 extra

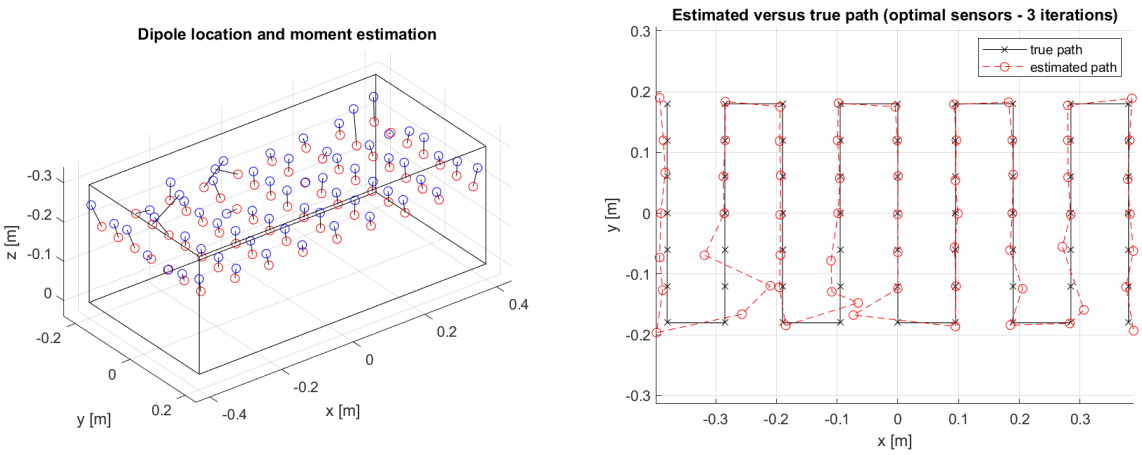

Figure 6.16: True locations of the magnetic dipole (red) and location estimate by the algorithm after 3 iterations (blue), shown in 3D and from a top view. Optimal sensor choice using QR pivoting has been used.

iterations. However, some of the good estimates are becoming worse.

The effects described above are also reflected in the location error histogram which compares the three methods, shown in Fig. 6.17. Clearly the same type of histograms are resulting from the test as in Section 6.1. The outliers of the random sensor choice model are visible, and the improvement of 3 iterations over 1 iteration is clear as well. Only location errors are shown, since the magnetic dipole moment was not measured independently from the algorithm.

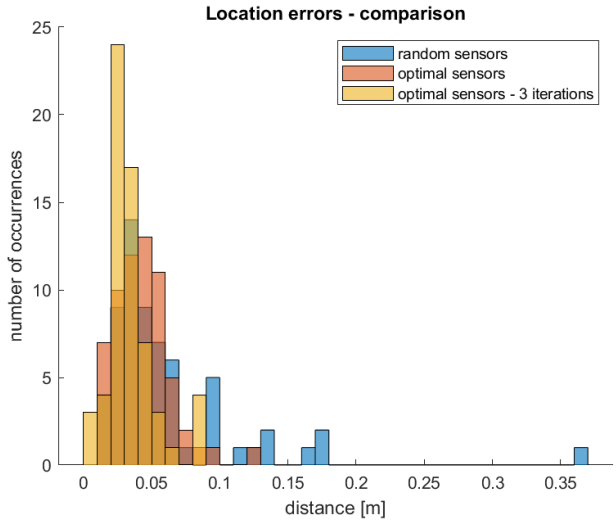

Figure 6.17: Histogram of location estimate errors, compared between random sensor choice with 1 iteration and optimal sensor choice with 1 and 3 iterations.

7

Conclusions and recommendations

7.1. Conclusions

Compressed sensing is an interesting method for processing relatively small amounts of data. An algorithm for localisation of a magnetic dipole in a 3D space was implemented. For sensor choice, two options were considered: random sensor choice, and optimal choice using QR pivoting. The algorithm is able to make accurate predictions of location and magnetic dipole moment of a randomly placed magnetic dipole. This is shown by simulation results, and verified using real-world measurements using a small magnet.

As expected, including noise in the measurements influences results: a larger noise standard deviation correlates with a large location error when estimating. Doubling the noise standard deviation roughly corresponds to a twice as large average location error. When the signal to noise ratio rises above 1, the average location error decreases: a signal to noise ratio of 1 results in about 10 cm error, where a ratio of 3 corresponds to a 5 cm average error.

The number of sensors and sensor choice method also influence the results. For a similar number of sensors, optimal choice using QR pivoting is especially beneficial compared to random choice if the number of sensors is small. In that case, the probability of choosing mostly 'bad' sensors is relatively large. For larger numbers of sensors, the difference between both choice methods vanishes. Comparing different numbers of sensors for equal choice methods, larger numbers of sensors perform slightly better. Again, the difference is largest for smaller numbers of sensors: the benefit of increasing from 4 to 8 sensors is way larger than that of increasing from 16 to 32 sensors.

The implemented algorithm features an option for iteration in combination with optimal sensor choice using QR pivoting. Instead of performing all measurements, and then making one estimate, it can also loop: let a few sensors measure, approximate the magnetic dipole location, and determine a new optimal set of sensors to measure. The new set of sensors is a consequence of shrinking the search area after each estimate. Iterating can indeed improve localisation performance.

Lastly, the influence of the source magnetic dipole location was analysed to discover a potential bias in the algorithm. First, it can be concluded that the minimum distance of the magnetic dipole to one of the grid points used for initialisation does not influence results. This indicates that the grid distances are chosen properly. Second, the distance to the origin of the search space is also not of any importance for the localisation accuracy. And third, the height of the magnetic dipole - determining how close the magnetic dipole is to the horizontal sensor array - does have a correlation with location error. However, this is solely due to a different signal to noise ratio which is directly related to distance to the sensor array. When correcting for signal to noise ratio, the height of the magnetic dipole does not influence results. All in all, the algorithm does not present any bias towards certain dipole locations.

7.2. Recommendations

For future research, there are some recommendations. First of all, there have been made some assumptions and simplifications in the algorithm that might have influenced the results. For instance, sensor arrays that are not horizontal have yet to be considered. Furthermore, a compensation vector has been introduced to avoid unwanted bias towards certain magnetic dipole locations. The values of this vector are now based on the vertical distance between the initialisation dipoles and the sensor array, which is a simplification of the actual situation where distances of the initialisation locations to *every* sensor must be taken into account. Besides, another way of computing this compensation must be found if the sensor array is not a horizontal plane.

The grid discretisation error has been discussed in this thesis. Based on an estimated magnetic dipole moment, the initialisation grid distance, and the estimated height above the sensor array, it can be estimated how large this discretisation error will be. However, in real-life applications the moment and height are often not known. Therefore, a method needs to be developed to deal with the discretisation error based on the actual magnetic field measurements. For example, an initial estimate of height and magnetic dipole moment can be made, after which additional actions can be taken such as considering a finer grid or increasing the error bound ϵ in the optimisation.

As discussed briefly in Section 4.4, the use of other methods to approximate the sparsest solution can be explored. One option is the use of greedy algorithms, such as an iterative matching pursuit algorithm. Another possibility is approximating the ℓ_0-norm by smoothing functions, making the optimisation problem that must be solved convex.

Regarding the iterative algorithm, the convergence properties must be proved. The approach taken in this thesis is heuristic, and produces good results. However, mathematically it is not yet clear if there will always be convergence to the true location of the magnetic dipole, and if so, under which conditions and at what rate. Moreover, different search shapes apart from the considered box shape can be explored to potentially improve results.

An important next step that can be taken is to consider not only the localisation problem, but also the detection problem. Multiple questions come to mind: what if it is not sure yet whether a magnetic dipole is present? Think for example of the container problem. It has to be found first, before it can be localised. Also, what if multiple sources are present? Can we find all of them, and which adaptations are needed to the proposed algorithm? Finding answers to these question is crucial on the way to actual application of this localisation algorithm.

II

Magnetic signature translation

8

Magnetic signatures of a ship

In this chapter, the problem of magnetic signature translation is introduced, and the physics of magnetic signatures of ships are discussed. First, the problem setup is given, followed by a brief description of the physical model used to describe it.

8.1. Problem description

In this part of the thesis, magnetic signatures of a ship are considered. A ship, made of ferromagnetic material, is placed in a homogeneous magnetic background field $\mathbf{B}_b$ - for example the Earth magnetic field. Apart from the background field, there are no magnetic field sources. Ferromagnetism inside the ship creates an induced magnetic field $\mathbf{B}_s$. This induced field is what is meant by the magnetic signature of the ship.

A first goal could be to estimate this signature based on a limited amount of (potentially noisy) measurements. To this end, imagine a sensor array present above the ship. This array is a 2D horizontal grid of sensors. The grid is typically placed on a distance above the ship of the same order of magnitude as the height of the ship. Some of the sensors in the array will measure all three components of the magnetic field, some of them will not. In reality, these sensors would be measuring the total magnetic field $\mathbf{B}_b + \mathbf{B}_s$. However, since it is fairly easy to measure the background field independently, and subtract it from the measurements, we assume that just $\mathbf{B}_s$ is measured. With the measurements above the ship, the goal is then to estimate the full magnetic field at all sensor locations - even the ones that did not measure. The field estimate will therefore be discrete, and not continuous.

The second goal, which is more interesting and will be the focus of this thesis, is to perform measurements above the ship, and use these to estimate the field below the ship. For this, a second 2D horizontal sensor array is positioned under the ship. Its distance below the ship is similar to the distance of the upper array to the ship. The setup is depicted in Fig. 8.1.

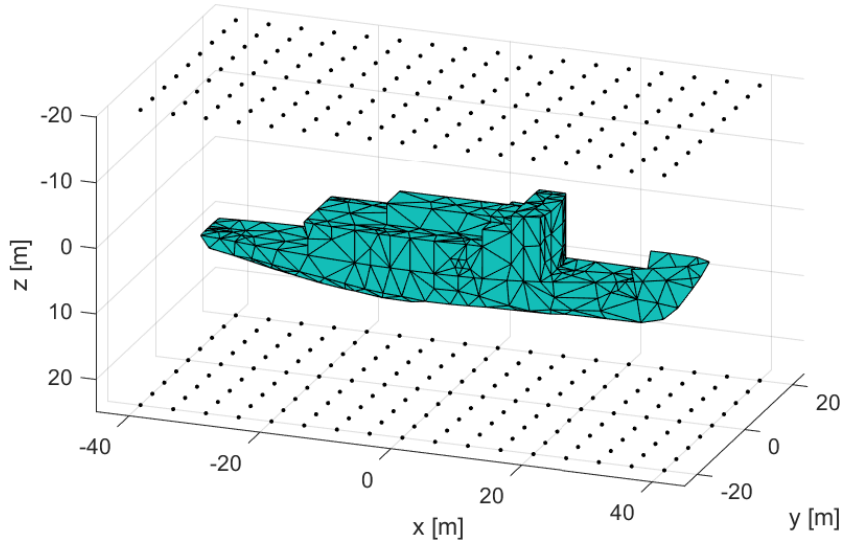

Figure 8.1: Setup of the ship and sensor arrays.

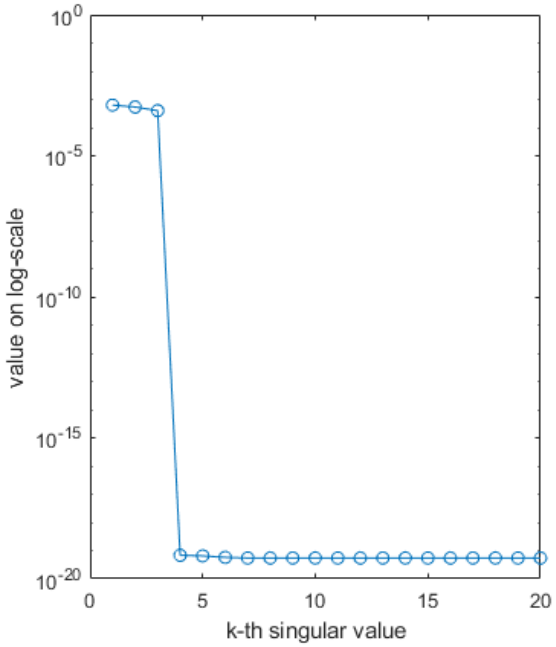

Figure 8.2: First 20 singular values for linear magnetisation.

8.2. Physical model

Since it is rather hard and costly to perform experiments with a real ship, a simulation is used. This simulation is able to calculate induced magnetic field values at certain points in space, using the geometry and properties of a ferromagnetic object. The simulation uses the Method of Moments (MoM), based on [25]. A realistic, generic ship geometry was used. Not a full model, but a simplified model without internal structure was used, to speed up computations. This model is the one shown in Fig. 8.1.

In the implementation of the MoM, two options for magnetisation are used. The first option is linear magnetisation. As in Eq. (2.28), magnetism of each part of the ship scales linearly with the background field. In that case, the singular values become as in Fig. 8.2: three singular values have similar non-zero value, and the

others are equal to zero. These three values correspond to fields in three orthogonal directions, and a linear combination of those can create any induced field.

The other option for the magnetisation includes hysteresis. For that, Rayleigh's hysteresis model [2] has been implemented. This is based on the Rayleigh law, which describes how magnetisation M depends on the applied magnetic field H. The model is gives by two equations:

$$M(H) = \chi_0 H + \alpha_R \mu_0 H^2 \tag{8.1}$$

and

$$M(H) = \chi_0 H - \alpha_R \mu_0 H^2, \tag{8.2}$$

where χ_0 is the initial susceptibility and α_R is called the Rayleigh constant. This model describes two branches: Eq. (8.1) describes an upper branch, where H is decreasing, and Eq. (8.2) describes a lower branch, where H is increasing. This law gives rise to a hysteresis loop: the induced magnetic field by the ship is no longer just the result of the background field, but also of its previous magnetisation. α_R determines the shape of this loop. An illustration of a hysteresis loop is shown in Fig. 2.2. The corners of the loop correspond to turning points of the applied field signal. This is where this signal changes direction and hence the loop changes branches. Consequently, many more variations are possible in the magnetic field above and below the ship. Therefore, the singular value spectrum has a smoothly decreasing shape, as can be seen in Fig. 8.3.

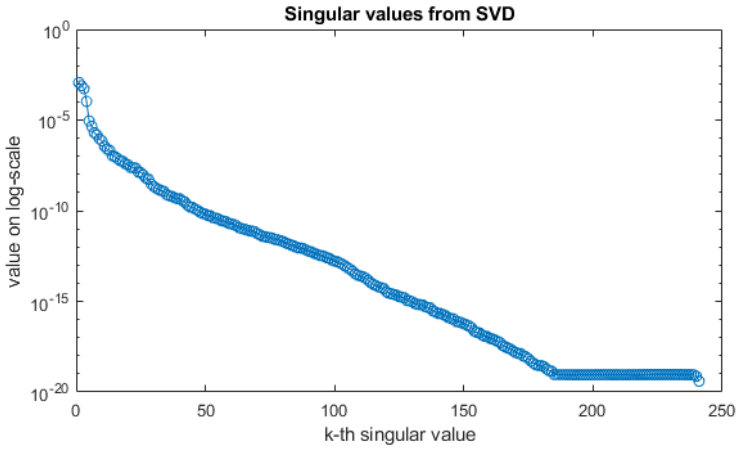

Figure 8.3: Example of singular values.

For the analysis that follows, always two datasets are constructed. The first set is used to construct the basis, and if applicable to choose optimal sensor locations. The second set is used for the actual validation and analysis. This set contains data that the algorithm has never 'seen' before.

Both sets are initialised using a different permanent magnetisations of all element in the geometry. These follow from real world measurements of a ship. In some of these measurements, a background field was applied, and in others, none was present. First, the volume of steel in the model was matched with the actual volume by adjusting the thickness of the triangular elements. Then, using some of the measurements with background field, the model geometry was aligned with the measurement data. Additionally, the magnetic susceptibility χ_m of the steel was estimated - the resulting value was 155, which is reasonable for steel. Using all the acquired properties, a measurement without background field was taken, which means that the full magnetic field present is caused by permanent magnetisation. Using an inversion technique, the field was translated into a permanent magnetisation for each element of the geometry. This was done for two separate ship measurements, which resulted in two permanent magnetisations.

After initialisation, the first dataset was constructed by applying randomly chosen background fields. The second dataset, for analysis, was made by simulating a round trip around the Earth: the Earth magnetic field present at each of the world locations served as background fields in the simulation. Here, hysteresis was also included in the model.

9

Gappy POD

As described in the Chapter 8, the goal of the signature problem is to perform some measurements above a ship or another object, and process these to estimate the ship's magnetic signature above and below it. Specifically, imagine two grid layers of field sensors, placed above and below the ship. Only part of the sensors above the ship perform measurements (possibly noisy), and we aim to reconstruct the field in both layers of sensors with these measurements. The first of two ways to solve this problem is explored in this chapter: Gappy POD. First, a background to the proper orthogonal decomposition (POD) and the adaptation of Gappy POD is given. Then, a method to choose sensor locations is described, and the application of Gappy POD to the magnetic field signature problem is explained. Lastly, the results of the implemented algorithms are given.

9.1. POD: Proper orthogonal decomposition

The main idea of the proper orthogonal decomposition (POD) is to decompose a certain dynamic physical field in orthogonal components:

$$\mathbf{u}(x,t) = \sum_{k=1}^{n} \mathbf{a}_k(t)\psi_k(x), \tag{9.1}$$

where $\mathbf{u}$ is the field, ψ_k are orthogonal basis functions, and $\mathbf{a}_k$ are the corresponding coefficients, or activations of these basis functions. When considering static problems, the coefficients $\mathbf{a}_k$ can be taken constant in time:

$$u(x) = \sum_{k=1}^{n} a_k\psi_k(x). \tag{9.2}$$

Alternatively, since we are exploring discrete magnetic fields in vector form, and not functions, this equation can be rewritten in the form of vectors:

$$\mathbf{u} = \sum_{k=1}^{n} a_k\boldsymbol{\psi}_k, \tag{9.3}$$

or as a system:

$$\mathbf{u} = \boldsymbol{\Psi}\mathbf{a}. \tag{9.4}$$

In discovering orthogonal components in sets of data, the singular value decomposition (SVD) naturally comes to mind. Refer to Section 3.3 for a detailed explanation of this method. Say that the field of interest is the field in a layer above the ship. Then all sensor measurements from that layer could be grouped in

one column $\mathbf{x}_1$. One can gather measurements $\mathbf{x}_k$ of many different fields, and add them as columns to a matrix $\mathbf{X} = [\mathbf{x}_1 \cdots \mathbf{x}_n]$. Applying the SVD gives

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*, \tag{9.5}$$

where $\mathbf{\Sigma}$ contains all singular values, and $\mathbf{U}$ contains the left-singular vectors of $\mathbf{X}$. In other words, $\mathbf{U}$ forms an orthogonal basis for the fields in $\mathbf{X}$. The singular value σ_k corresponding to singular vector $\mathbf{u}_k$ can be seen as a measure of energy, such as described earlier with Eq. (5.7). The larger σ_k, the more influential $\mathbf{u}_k$ is in describing the original fields $\mathbf{x}$. An actual example of singular values from the application to ship signatures is shown in Fig. 8.3.

The basis $\mathbf{U}$ can give rise to a reduced basis $\tilde{\mathbf{U}}$, containing not n, but less basis vectors. Since $\tilde{\mathbf{U}}$ essentially contains the basis fields ψ_k, it makes sense to rename the matrix to $\mathbf{\Psi}$.

The goal is to reconstruct a field $\mathbf{x}$ from a measurement $\tilde{\mathbf{x}}$, which is for the moment equal to $\mathbf{x}$. First, the coefficients $\hat{a}_k$ are estimated, by solving a system as in Eq. (9.4):

$$\tilde{\mathbf{x}} = \mathbf{\Psi}\hat{\mathbf{a}}. \tag{9.6}$$

Since $\mathbf{\Psi}$ is an orthogonal basis, the coefficients a_k can simply be found by projecting the measurement $\tilde{\mathbf{x}}$ onto each of the orthogonal basis vectors $\boldsymbol{\psi}_k$. In other words, taking the inner product:

$$\hat{a}_k = \langle \tilde{\mathbf{x}}, \boldsymbol{\psi}_k \rangle . \tag{9.7}$$

Then, the lower rank approximation of $\mathbf{x}$ is

$$\hat{\mathbf{x}} = \mathbf{\Psi}\hat{\mathbf{a}} = \sum_{k=1}^{r} \hat{a}_k \boldsymbol{\psi}_k. \tag{9.8}$$

Note that in general, the system Eq. (9.6) will be underdetermined. If the complete basis $\mathbf{\Psi} = \mathbf{U}$ is used, instead of a reduced basis, the system Eq. (9.6) is not underdetermined, hence the reconstruction will be perfect: $\hat{\mathbf{x}} = \mathbf{x}$.

9.2. Gappy POD: gappy proper orthogonal decomposition

It might be impossible or less practical to measure all elements of the magnetic field vector. For example, if one sensor is used to measure at all grid positions, and there is not enough time to cover all of them. In this case, the collected data is said to be *gappy*. The POD technique can be extended to this situation. The resulting method is appropriately called *Gappy POD*.

Now, the measurement $\tilde{\mathbf{x}}$ does not cover the complete vector $\mathbf{x}$, but only part of it:

$$\tilde{\mathbf{x}} = \mathbf{P}\mathbf{x}. \tag{9.9}$$

Here, $\mathbf{P}$ is a measurement matrix, which contains selected rows of the identity matrix. Note that this definition is similar to that of the measurement matrix $\mathbf{C}$ in Part I, just with a different symbol. If the vector entry is measured, the corresponding row is added to the matrix. For instance, when measuring a field vector of length 6, and measuring only elements 1, 3, 4, and 5 of $\mathbf{x}$, results in the measurement matrix

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{9.10}$$

Like in the original POD method, the coefficients a_k of the basis fields need to be estimated to reconstruct the original field $\mathbf{x}$. However, the basis fields $\boldsymbol{\psi}_k$ are not orthogonal in the measurement domain. To see how this

can be the case, consider the measurement matrix $\mathbf{P}$ in Eq. (9.10). Examples of two orthogonal field vectors $\mathbf{x}_1, \mathbf{x}_2$ and the corresponding measurements $\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2$ are:

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}; \quad \tilde{\mathbf{x}}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tilde{\mathbf{x}}_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{9.11}$$

Obviously $\langle \mathbf{x}_1, \mathbf{x}_2 \rangle = 0$, hence the fields were orthogonal in the original domain. The resulting measurements, however, are not orthogonal: $\langle \tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2 \rangle = 1$.

The basis vectors are in general not orthogonal any more, and therefore it is not possible to estimate the coefficients using a simple inner product as in Eq. (9.7). A reasonable way to estimate them in this situation would be to minimise some norm of the difference between measurement entries and estimated field entries. For that, it is needed to compare an incomplete vector and a full vector, only at the entries the incomplete vector covers. We use the measurement matrix $\mathbf{P}$ to make the complete vector just as incomplete as the other vector. For a measurement $\tilde{\mathbf{x}}$ and an estimate $\hat{\mathbf{x}}$, the error to be minimised is:

$$\epsilon = \left\| \tilde{\mathbf{x}} - \mathbf{P}\hat{\mathbf{x}} \right\|_2^2 = \left\| \tilde{\mathbf{x}} - \mathbf{P} \sum_{k=1}^{r} \hat{a}_k \boldsymbol{\psi}_k \right\|_2^2 = \left\langle \tilde{\mathbf{x}} - \mathbf{P} \sum_{k=1}^{r} \hat{a}_k \boldsymbol{\psi}_k, \tilde{\mathbf{x}} - \mathbf{P} \sum_{k=1}^{r} \hat{a}_k \boldsymbol{\psi}_k \right\rangle. \tag{9.12}$$

To minimise ϵ, one can differentiate ϵ with respect to each of the $\hat{a}_j$, and set the result equal to zero:

$$\frac{\partial \epsilon}{\partial \hat{a}_j} = 0 \implies \left\langle \tilde{\mathbf{x}} - \mathbf{P} \sum_{k=1}^{r} \hat{a}_k \boldsymbol{\psi}_k, \mathbf{P}\boldsymbol{\psi}_j \right\rangle = 0 \qquad j \in \{1, 2, \ldots, r\}. \tag{9.13}$$

Note that the inner product is linear in the first argument, hence:

$$\left\langle \mathbf{P} \sum_{k=1}^{r} \hat{a}_k \boldsymbol{\psi}_k, \mathbf{P}\boldsymbol{\psi}_j \right\rangle = \left\langle \tilde{\mathbf{x}}, \mathbf{P}\boldsymbol{\psi}_j \right\rangle, \tag{9.14}$$

or

$$\sum_{k=1}^{r} \left\langle \mathbf{P}\boldsymbol{\psi}_k, \mathbf{P}\boldsymbol{\psi}_j \right\rangle \hat{a}_k = \left\langle \tilde{\mathbf{x}}, \mathbf{P}\boldsymbol{\psi}_j \right\rangle. \tag{9.15}$$

This defines the following linear system to solve for the coefficient vector $\hat{\mathbf{a}}$:

$$\mathbf{M}\hat{\mathbf{a}} = \mathbf{f}, \tag{9.16}$$

where for $i, j \in \{1, 2, \ldots, r\}$:

$$M_{ij} = \left\langle \mathbf{P}\boldsymbol{\psi}_i, \mathbf{P}\boldsymbol{\psi}_j \right\rangle; \tag{9.17}$$

$$f_i = \left\langle \tilde{\mathbf{x}}, \mathbf{P}\boldsymbol{\psi}_i \right\rangle. \tag{9.18}$$

The matrix $\mathbf{M}$ is square: $r \times r$. In the case that all entries of $\mathbf{x}$ are measured ($\mathbf{P} = \mathbf{I}$), it follows that $\mathbf{M} = \mathbf{I}$, since $\boldsymbol{\Psi}$ is orthonormal. For increasingly dense measurements, vectors $\mathbf{P}\boldsymbol{\psi}_i$ and $\mathbf{P}\boldsymbol{\psi}_j$ become increasingly more similar to $\boldsymbol{\psi}_i$ and $\boldsymbol{\psi}_j$. Consequently, $\mathbf{M}$ converges to the identity matrix.

To solve for $\hat{\mathbf{a}}$, one must calculate the entries of $\mathbf{M}$ and $\mathbf{f}$ and then invert the system in Eq. (9.16). Since $\mathbf{M}$ is square, there is in general a unique solution to $\hat{\mathbf{a}}$.

9.2.1. Tikhonov regularisation

Apart from simple inversion, Tikhonov regularisation is explored. This enables taking the energy, or weight, of the basis vectors into account. Usually, Tikhonov regularisation is used in cases where a linear system is underdetermined. In our situation, we can use the method to create a better estimate in case the measurements contain some error. Normally, every measurement error is contained in $\mathbf{f}$ directly, and propagates into the solution $\hat{\mathbf{a}}$. However, we know that some basis vector describe generally a larger part of the solution than others. Using Tikhonov regularisation, these properties can be used.

For an underdetermined linear system $\mathbf{Ax} = \mathbf{b}$, usually a simple least squares optimisation is used:

$$\mathbf{x} = \arg\min_{\mathbf{y}} \left\|\mathbf{Ay} - \mathbf{b}\right\|_2^2. \tag{9.19}$$

Tikhonov regularisation adds an extra regularisation term to this optimisation, related to the solution vector by some matrix $\mathbf{\Gamma}$:

$$\mathbf{x} = \arg\min_{\mathbf{y}} \left\|\mathbf{Ay} - \mathbf{b}\right\|_2^2 + \lambda \left\|\mathbf{\Gamma y}\right\|_2^2. \tag{9.20}$$

Then, a closed form solution of this minimisation problem (with parameter λ) is

$$\mathbf{x}_\lambda = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{\Gamma}^T\mathbf{\Gamma})^{-1}\mathbf{A}^T\mathbf{b}. \tag{9.21}$$

In this situation, we want to make sure that basis modes with a higher energy are used relatively more than modes with low energy. Since the regularisation term acts as a penalty in the optimisation, $\mathbf{\Gamma}$ should describe something like inverse energy. From the SVD, a measure for energy of each of the basis fields (singular vectors) is already present: singular values. The larger a singular value, the more energy that corresponding mode contains. For $\mathbf{\Gamma}$, we can therefore use a diagonal matrix, with entries equal to one over the singular values from the SVD:

$$\mathbf{\Gamma} = \begin{bmatrix} \frac{1}{\sigma_1} & 0 & \dots & 0 \\ 0 & \frac{1}{\sigma_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{1}{\sigma_r} \end{bmatrix}, \tag{9.22}$$

where σ_i denotes the singular value of basis mode $\boldsymbol{\psi}_i$.

The question that remains is how to choose the regularisation parameter λ properly. Too small and it has virtually no effect on the optimisation result, too large and only the very few basis modes with a high energy get a non-zero coefficient. A helpful tool in selecting a 'good' λ is the so called L-curve criterion. In the L-curve, the residual $\left\|\mathbf{Ax} - \mathbf{b}\right\|_2^2$ is compared to the regularisation term $\left\|\mathbf{\Gamma x}\right\|_2^2$. The idea, which involves some heuristics, is that these should be 'in balance'.

Suppose $\mathbf{A}$, $\mathbf{b}$, and $\mathbf{\Gamma}$ are given. In the L-curve, a range of values for λ are explored. For every λ, a different solution $\mathbf{x}_\lambda$ can be calculated by Eq. (9.21). Then for every $\mathbf{x}_\lambda$, the residual $\left\|\mathbf{Ax}_\lambda - \mathbf{b}\right\|_2^2$ and regularisation term $\left\|\mathbf{\Gamma x}_\lambda\right\|_2^2$ are computed. These values define one point in the L-curve, usually plotted on a logarithmic scale. An example of such a curve is shown in the left plot of Fig. 9.1.

Now, good values for λ give a right balance between the residual (the values on the x-axis) and the regularisation term (values on the y-axis). L-curves - as the name suggests - have the shape of the letter 'L', with a steep vertical side and an almost horizontal side. A relatively sharp corner is connecting both sides. Choosing a value for λ inside the corner makes sure that neither the residual term nor the regularisation term dominates. Then how do we know which λ is most inside the corner? Hansen and O'leary suggest to use the value of λ at which the L-curve has maximum curvature. At this point, the residual term and regularisation term can be seen as most 'in balance'. Moving away from this point on the curve will either be a very vertical or very horizontal movement. This indicates that one of the term is decreased slightly, but at the expense of a large increase in the other term. When defining

$$\eta = \log\left\|\mathbf{\Gamma x}_\lambda\right\|_2^2; \tag{9.23}$$

$$\rho = \log\left\|\mathbf{Ax}_\lambda - \mathbf{b}\right\|_2^2, \tag{9.24}$$

and η', ρ', η'', ρ'' as their first and second derivatives with respect to λ, the curvature is given by:

$$\kappa = 2\frac{\rho'\eta'' - \rho'\eta''}{(\rho'^2 + \eta'^2)^{\frac{3}{2}}}. \tag{9.25}$$

In Fig. 9.1, next to an example of an L-curve, its curvature is plotted. The curvature clearly shows a peak, at the top of which the value of λ is chosen. This value corresponds to the point most 'inside the corner' in the L-curve.

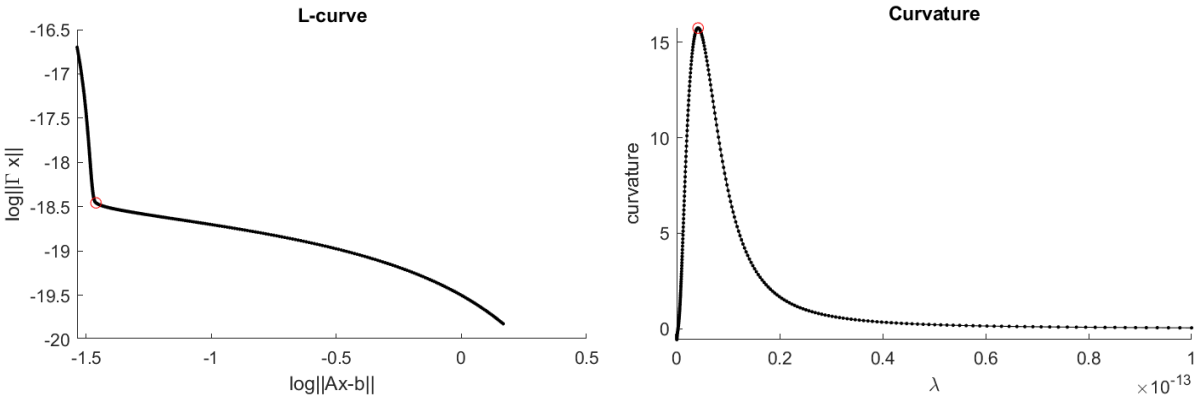

Figure 9.1: Example of an L-curve, and its curvature with respect to λ. The red circle indicates the value of $\lambda \approx 4.2 \cdot 10^{-15}$ that was chosen here.

9.3. Choosing measurements

Up to this point, $\mathbf{P}$ has been an abstract measurement matrix. Imagine a scenario in which there is a fixed number of measurements available. One could of course simply choose a random set of vector entries to measure. However, it might very well be that certain entries provide much more information than others about the activation of basis modes. This section describes how measurements, or sensor locations, can be chosen in a smart way.

In the Gappy POD method, basis mode coefficients are estimated by solving a system $\mathbf{M}\hat{\mathbf{a}} = \mathbf{f}$, as in Eq. (9.16). From there, a field estimate can be made, but all information is essentially contained in the estimate of coefficients $\hat{\mathbf{a}}$. Therefore, the quality of the final field estimate depends fully on how good the inversion of the linear system Eq. (9.16) is. In particular, noise in the measurements should have minimal influence on the solution. As described in more detail in Section 3.5, the condition number of $\mathbf{M}$ indicates how sensitive the approximation is to measurement noise. A way to choose measurements is to choose ones that minimise the condition number $\kappa(\mathbf{M})$.

We consider a set of available sensor locations, where a sensor at each individual location can produce multiple measurements. In this case, not the best measurements, but the best sensor locations must be chosen. To achieve this goal, Willcox [37] suggests using a greedy approach. For every new measurement that must be chosen, the proposed algorithm loops over all possible measurements. For each, a new $\mathbf{M}$ is constructed, and its condition number $\kappa(\mathbf{M})$ is computed. The new measurement that resulted in the smallest value of $\kappa(\mathbf{M})$ is chosen. The algorithm is greedy in the sense that at every step, the optimal sensor is chosen, but the results of all sensors together is never considered. In other words: a local minimiser is chosen at each step. The steps are as follows:

1. To find the best new sensor location, for each uncovered sensor location:
   (a) Place a sensor at that location.
   (b) Construct the matrix $\mathbf{M}$ using new and previous sensors.
   (c) Calculate the condition number $\kappa(\mathbf{M})$.

2. Identify the sensor location which resulted in the smallest value of $\kappa(\mathbf{M})$.

3. Add this sensor, delete from the list of available locations, and repeat.

The algorithm has two main downsides, but they can both be surpassed more or less. First of all, the routine is computationally expensive, since it loops over every available sensor location for every new sensor that must be chosen. The good news, however, is that this can all be done beforehand. Once the basis modes $\boldsymbol{\psi}_i$ are known, the matrices $\mathbf{M}$ can be constructed. Specific measurement data or other information is not needed.

Second, the algorithm does not guarantee an optimal result in terms of the condition number. Even though the sensor location resulting in the smallest condition number is chosen, there might be another equally large set of sensors that together produce an even lower condition number. Although the results might not be optimal, they are almost always good in practice [37]. Typically, the condition number decreases rapidly while adding extra sensors. If the number of measurements chosen is approximately equal or larger than the number of modes covered in the basis $\boldsymbol{\Psi}$, then the condition number becomes low enough. More optimisation is in that case not needed.

9.4. Application of Gappy POD to magnetic field signatures

As described in Chapter 8, the goal in this part of the thesis is magnetic signature translation from above to ship to below it. In order to reconstruct a field below the ship using Gappy POD with just measurements above the ship, the fields above and below need to be coupled. This is done when constructing the basis $\boldsymbol{\Psi}$. Measurements are collected of the full sensor array above and below the ship. Each measurement is a column vector $\mathbf{x}_i$, of which the upper half contains measurements from above the ship, and the lower half of below the ship. A measurement vector has the following structure:

$$\mathbf{x}_i = [B_{u1,x}, B_{u1,y}, B_{u1,z}, B_{u2,x}, \ldots, B_{um,z}, B_{l1,x}, \ldots, B_{ln,z}]^T, \tag{9.26}$$

where for example $B_{u1,x}$ is the x-component field measurement of the first $u$pper sensor, and $B_{ln,z}$ is the z-component field measurement of the n^{th} $l$ower sensor. These measurement vectors are put together into a large matrix $\mathbf{X}$ and the SVD is applied to it. Following the procedures from Section 9.1 and Section 9.2, a basis $\boldsymbol{\Psi}$ is created. This basis then consists of basis fields $\boldsymbol{\psi}_i$ that contain both the field above and below the ship. Therefore, when using Gappy POD, the estimated mode coefficients in $\hat{\mathbf{a}}$ not only describe the field above the ship, but also below the ship.

9.5. Results

In this section, results are shown of applying the Gappy POD method to the ship magnetic signature problem. First, some example results are shown to illustrate what the obtained results typically look like. Then, a number of properties are varied and their influence is analysed: sensor choice, the number of modes, the number of sensors, the height of the sensor arrays, and measurement noise. Throughout the subsection, the application of Tikhonov regularisation is explored as well. All the results in this section are created using the MoM implementation with the Rayleigh hysteresis model and parameter $\alpha = 0.1$. This value introduces a relatively small hysteresis effect.

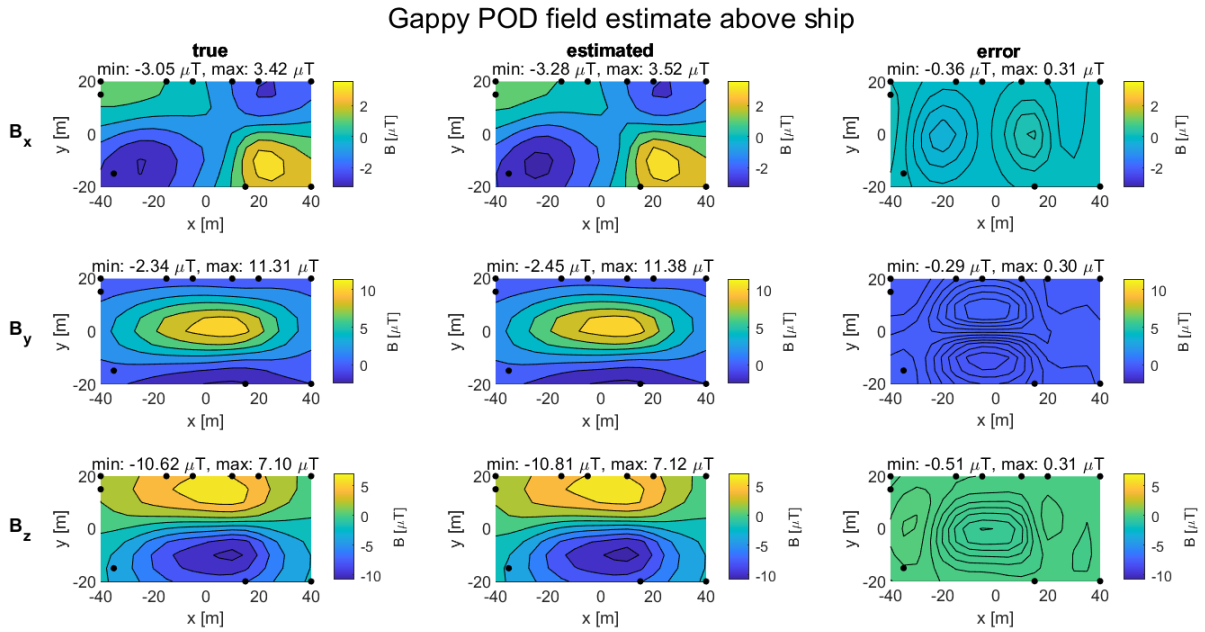

Figure 9.2: Example of a Gappy POD estimate of the field above the ship, for 10 modes and 10 optimally chosen sensors (depicted by black dots).

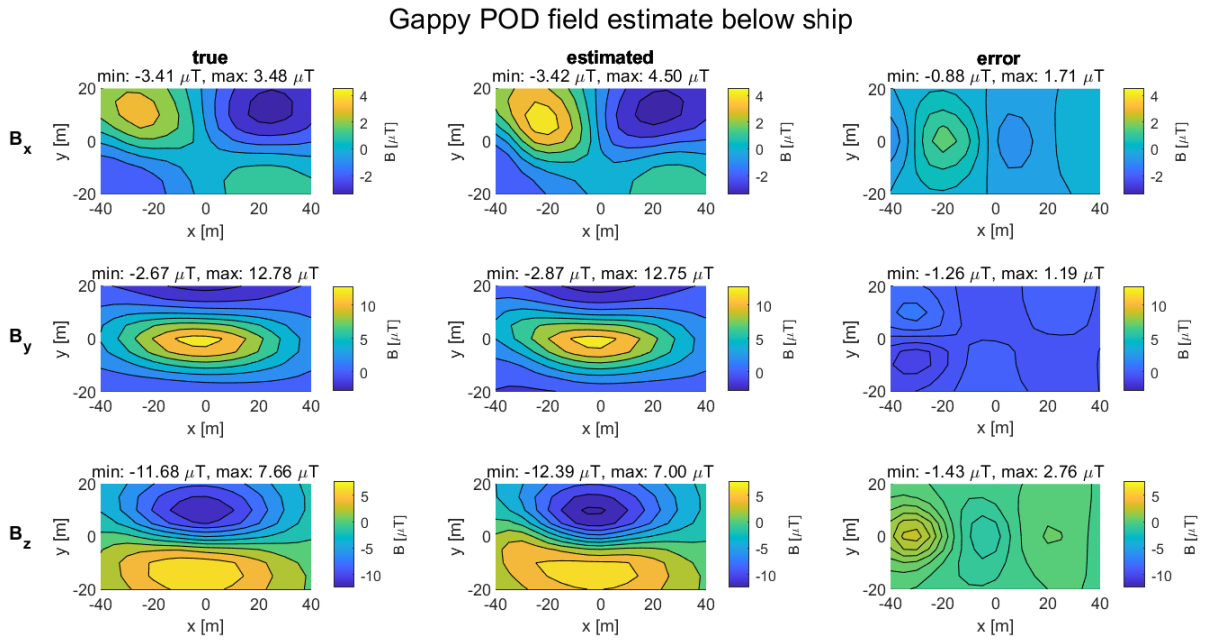

Figure 9.3: Example of a Gappy POD estimate of the field below the ship, for 10 modes and 10 optimally chosen sensors (depicted by black dots).

9.5.1. Examples

In Fig. 9.2 and Fig. 9.3, an example of applying Gappy POD to a measurement using 10 random sensors is shown. Fig. 9.2 shows the three field components above the ship, the locations where the field was measured (chosen using the algorithm from Section 9.3), the estimated field components above the ship, and the error. Fig. 9.3 shows the corresponding estimated field components below the ship (computed from the same estimated coefficients $\hat{\mathbf{a}}$) and the error.

Fig. 9.4 and Fig. 9.5 shows the exact same true field and measurements, but now the field reconstruction is performed using Gappy POD including Tikhonov regularisation. Compared to the classic Gappy POD

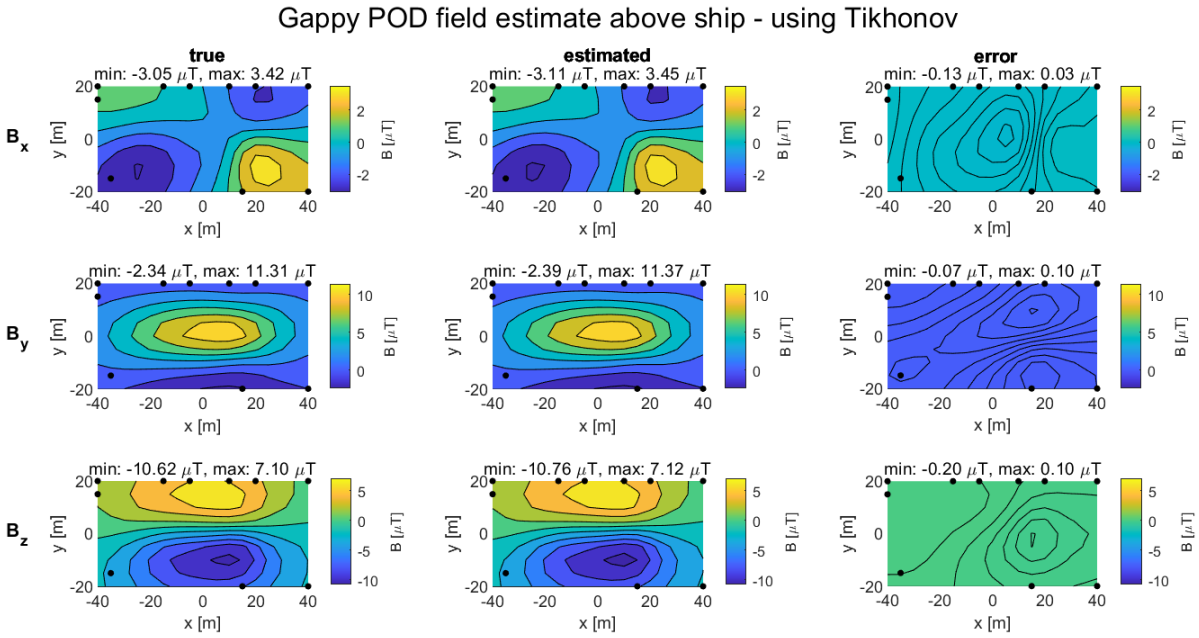

Figure 9.4: Example of a Gappy POD estimate of the field above the ship, for 10 modes and 10 optimally chosen sensors (depicted by black dots). Tikhonov regularisation is used.

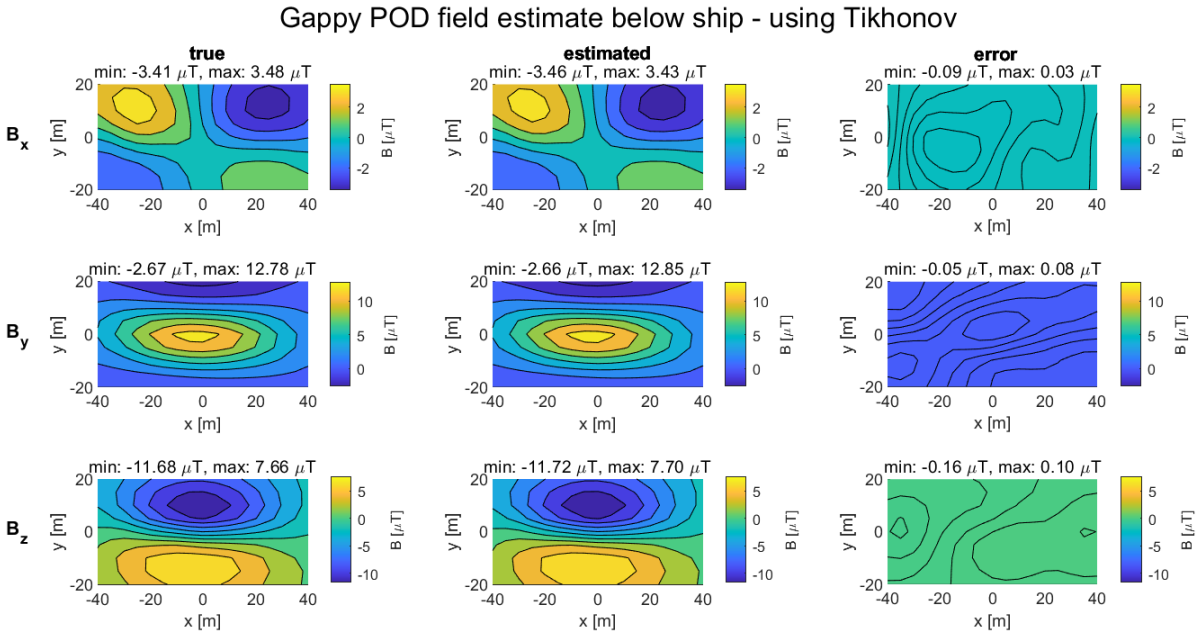

Figure 9.5: Example of a Gappy POD estimate of the field below the ship, for 10 modes and 10 optimally chosen sensors (depicted by black dots). Tikhonov regularisation is used.

method, the reconstruction is much better, with the errors being around three times smaller in every field component above the ship, and about ten times smaller below the ship.

In Fig. 9.6, an example of mode activation estimation is given, where in one estimate Tikhonov regularisation is used, and in the other estimate classical Gappy POD is used. It is interesting to see how close the estimate with Tikhonov gets, while classical Gappy POD is especially far off for the higher modes, which have much lower energy.

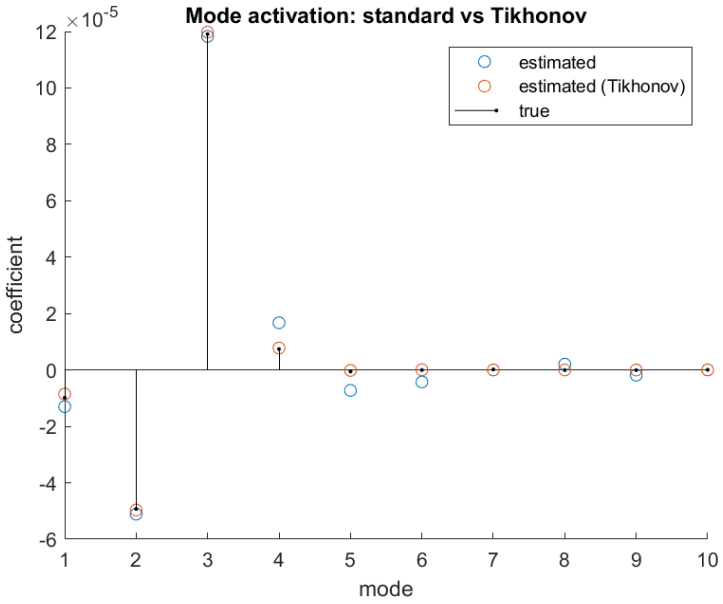

Figure 9.6: Comparison of the mode activation estimates when Tikhonov regularisation is used and not used.

9.5.2. Sensor choice

The behaviour of the optimal sensor choice algorithm described in Section 9.3 was analysed. Fig. 9.7 shows how the sensor grid is being generated, and Fig. 9.8 shows the corresponding condition number of the matrix **M**. It is clear that after just as much as four chosen sensors, the condition number of **M** drops spectacularly. This is expected: in general, an equal or larger number of sensors is needed than the number of considered basis fields [7]. Only with a larger or equal number of measurements than modes it is possible to determine the coefficients of all basis fields. With less measurements, the system that is solved is underdetermined. With three chosen sensors, nine measurements are available, which is still too few. After choosing the fourth sensor, the inversion is possible without too large perturbations of the solution.

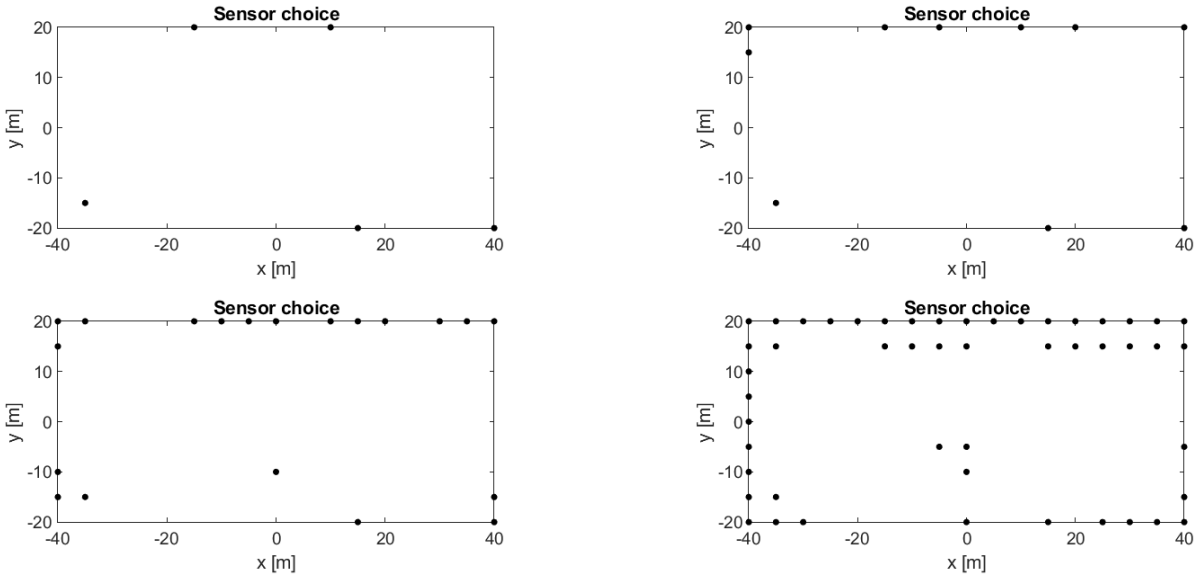

Figure 9.7: Sensor choice after choosing 5, 10, 20, and 50 sensors from the grid.

Interesting to note is the order of sensor choice: most of the sensors are chosen on the outside of the region, far away from the centre of the ship. Apparently, the distinction between the first 10 basis modes is best made using measurements near the edges of the grid.

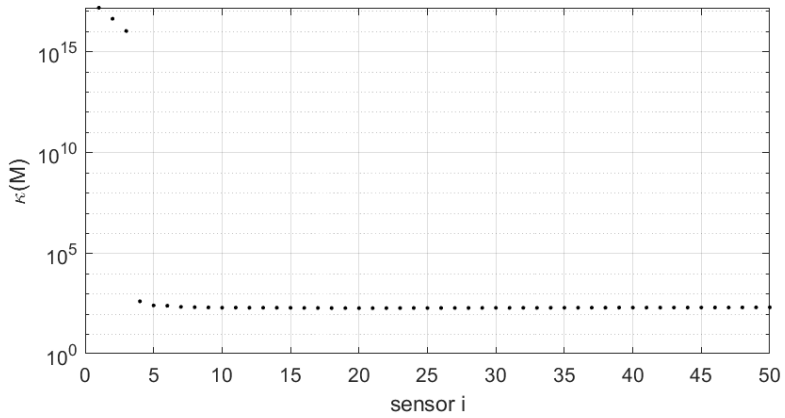

Figure 9.8: Condition number after each i^{th} sensor choice.

9.5.3. Number of modes and sensors

In this subsection, the influence of the number of modes and the number of sensors is analysed. The number of modes was varied from 1 to 25 or 30 depending on configuration. The number of sensors has been chosen between 1 and 100. The white noise added to the signal was taken with a standard deviation of 5 percent of the total signal reach (max - min). For each combination of a number of modes and sensors, 100 runs were performed using Gappy POD. In each run, a new field was applied to the ship, resulting in new measurements and a new estimate for the field below the ship. The root mean square error of the field below the ship was computed, and subsequently the normalised root mean square error (NRMSE) was calculated. This error is defined as the RMSE divided by the range of the true field (its maximum value over the whole sensors array minus its minimum value).

The results are shown in Fig. 9.9 to Fig. 9.12. The experiment was done for random sensors choice and optimal choice, and for classic Gappy POD and including Tikhonov regularisation. All of the four methods are given the same fields for reconstruction, so that the four figures are directly comparable. Since 100 runs were performed for each 'block' in the figures, the result can seem slightly non-smooth, due to some randomness in the presented fields and measurements.

Figure 9.9: NRMSE error of field below ship for varying number of modes and number of sensors. Standard Gappy POD with random sensor choice is used for reconstruction. Each block displays the average NRMSE over 100 runs with Gappy POD. All yellow blocks have NRMSE 1.0 or above.

The first thing that immediately stands out, is the large difference in error between classical Gappy POD and Gappy POD including Tikhonov regularisation. In equal situations, Tikhonov performs in almost all cases way better. Especially in situations where the number of sensors is (way) smaller than the number of modes, Tikhonov regularisation outperforms. The only situations where classical Gappy POD gives acceptable results is in the case of very few basis modes (maximum around 8), and a relatively large number of sensors. This can be explained by the idea that if more modes are available, chances of overfitting are larger. The extra modes always have a smaller singular value, meaning that in general, these modes do not contribute much to the signals. However, classical Gappy POD considers them equally to the first few very important modes. Tikhonov regularisation solves this issue by taking into account the energy of each basis mode.

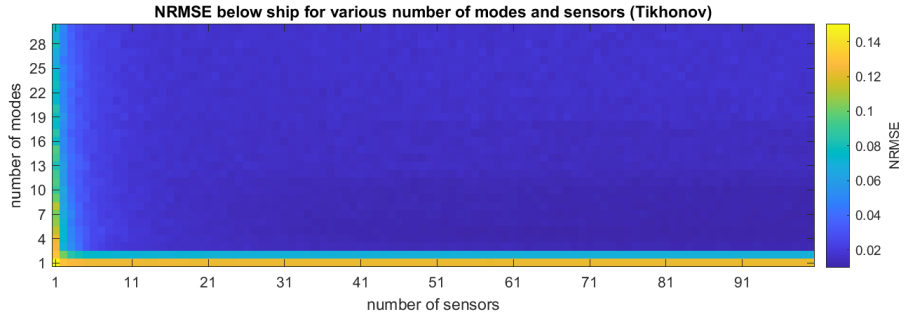

Figure 9.10: NRMSE error of field below ship for varying number of modes and number of sensors. Gappy POD with Tikhonov regularisation, with random sensor choice is used for reconstruction. Each block displays the average NRMSE over 100 runs with Gappy POD.

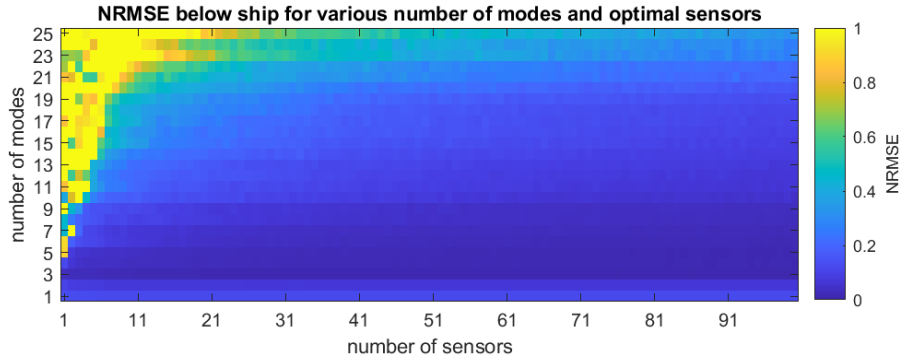

Figure 9.11: NRMSE error of field below ship for varying number of modes and number of sensors. Standard Gappy POD with optimal sensor choice is used for reconstruction. Each block displays the average NRMSE over 100 runs with Gappy POD. All yellow blocks have NRMSE 1.0 or above.

Another behaviour that is clearly visible from the figures, is the improvement of results when the number of sensors is increased (for equal number of basis modes). This makes complete sense: the more measurements are taken, the more information is available to determine the basis modes coefficients $\hat{\mathbf{a}}$. Consequently, a better field estimate can be made.

Third, optimal sensor choice improves the results for classical Gappy POD compared to random choice, especially when few sensors are chosen. This is visible by the border of the yellow part (NRMSE > 1.0) in Fig. 9.11 being clearly further to the left than in Fig. 9.9. This is not strange: if only four sensors are randomly selected, chances are relatively large that it is hard to distinguish the different basis modes from from the measurements. If more sensors are chosen, the randomly selected sensors are likely to be more or less spread out over the region, which lowers the need for smartly chosen sensors.

On the other hand, when Tikhonov regularisation is included, the optimal sensor choice does not make much difference. This is quite a spectacular result and emphasises how powerful Tikhonov regularisation can be. Even when choosing four random sensors, Gappy POD is in general able to estimate the field well with Tikhonov. This is in sharp contrast with classical Gappy POD, where reconstruction results can be ruined because of badly chosen sensors.

9.5.4. Height of sensor arrays

The height of both the upper and lower sensor arrays as shown in Fig. 8.1 was varied. This was done for a fixed number of 10 modes and 10 randomly chosen sensors (so 30 measured field components). The standard deviation of the Gaussian noise added to each measurements was for each run 5 percent of the total signal range above the ship (max - min). In this way, the influence of a changing signal to noise ratio while varying heights was minimised. Tikhonov regularisation was included. Each block in the figure corresponds to the average NRMSE of the field below the ship of 10 runs.

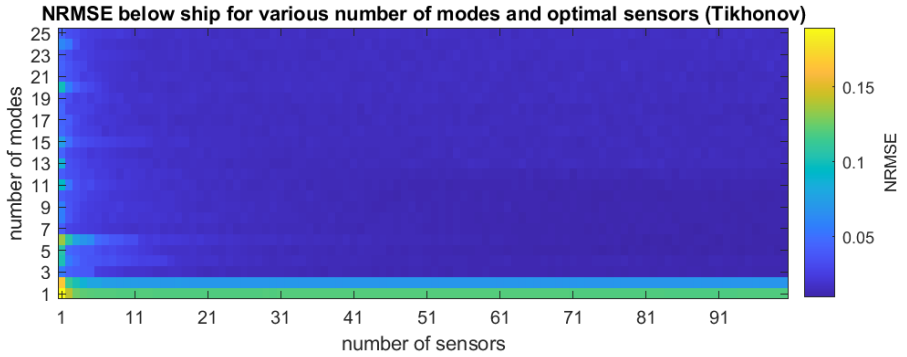

Figure 9.12: NRMSE error of field below ship for varying number of modes and number of sensors. Gappy POD with Tikhonov regularisation, with optimal sensor choice is used for reconstruction. Each block displays the average NRMSE over 100 runs with Gappy POD.

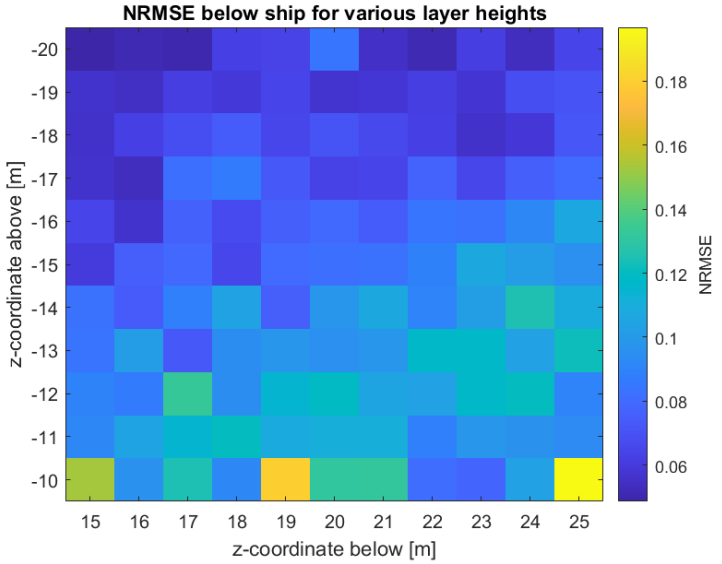

Figure 9.13: NRMSE error of the field below the ship for a range of heights for the upper and lower sensor array. The left bottom corner corresponds with both sensors arrays close to the ship. Each block displays the average NRMSE over 100 runs with Gappy POD.

Striking is that errors are so small when the sensor array above the ship is far away from the ship, and get larger when the array is moved towards it. Probably, the strong local field effects of the ship influence the measurements more, causing the Gappy POD reconstruction to get worse. When the sensor array is further away, it is better possible to estimate the basis field coefficients. It is also interesting that this is the only clear relation between error and sensor array heights. There seems to be no relation for the error between the distances of the arrays above and below the ship. For instance, it might have been expected that a combination of an array far away above and an array far away below would work better than one array close and another far away. However, this is not the case.

9.5.5. Noise

In Fig. 9.14, the influence of measurement noise on the reconstruction errors is explored. Again, this was done for a fixed number of 10 modes and 10 randomly chosen sensors. Tikhonov regularisation was used. The normalised noise standard deviation was computed by dividing the standard deviation by the range of the signal (max - min). As expected the potential reconstruction error increases when the noise is increased.

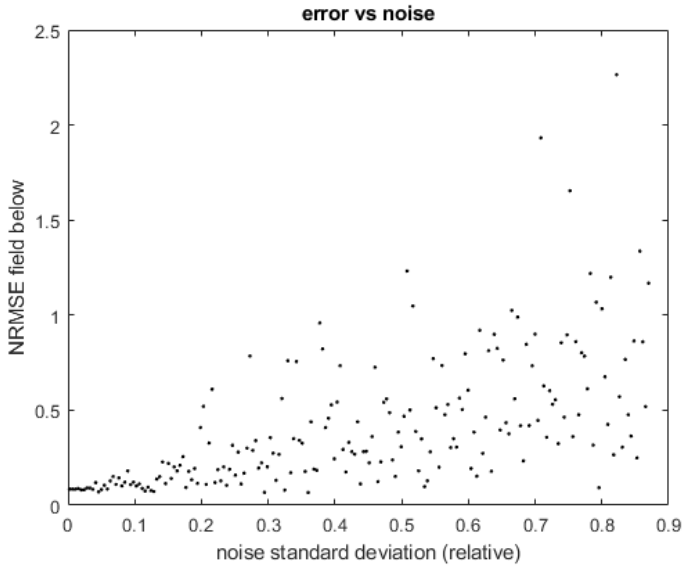

Figure 9.14: NRMSE error of field below ship versus normalised noise standard deviation.

10

Neural networks

Neural networks are a specific kind of machine learning models. The overall goal in every neural network is to predict some output based on a certain input. The output can be predicted by the network because it has been trained before, with similar data. The output can for instance be a value, a set of values, or a label. Based on the training data, consisting of inputs and corresponding targets (desired outputs), the network tries to learn during the training process how to predict outputs from inputs. A neural network is no ordinary numerical model, in the sense that it is not predefined by the author how the prediction must be done, but instead the network trains itself. This chapter describes how neural networks have been applied to the magnetic field signature estimation problem, and the results.

10.1. Introduction

There are many types of neural networks, each fitting for their specific purpose. Two main purposes of these networks are classification and regression. For classification, one can think of recognising objects in images, or face recognition. Regression, however, is the process of approximating a certain value or set of values. The neural networks used in this research are all regression neural networks: we try to estimate field data, hence we are performing a regression. The upcoming sections will discuss the general architecture of neural networks - what *are* they, and what do they consist of - followed by an explanation of the training process. Here, the focus will be on regression neural networks.

10.1.1. Architecture

Neural networks are networks of connected *neurons*. A very simple network is shown in Fig. 10.1. This is an example of a *feedforward neural network*, the first, simplest, and most important form of neural networks. When an input is fed into the network, all the neurons in the network will get a value, or *activation*. It starts at the left of the network: in Fig. 10.1 there are three input values (the black dots). Each of these values is sent to the first layer of three neurons (the column of gray circles). Each neuron receives three input values. Based on these input values, the neuron calculates its own activation. For now, suppose this is linear. Then, taking three input values x_1, x_2, and x_3, the neuron calculates is own activation y as

$$y = a_1 x_1 + a_2 x_2 + a_3 x_3 + b. \tag{10.1}$$

The coefficients a_i are called the *weights*. Every connection - each arrow in the image - has a weight. These determine how important each piece of information is in the following step. The value b is called a *bias*. Every neuron has a bias value. The weights and biases together form the learnable parameters: these will change during the training process. Now, every neuron in the first layer has calculated its activation value, and these values are propagated to the next layer, where the process repeats. This network is called feedforward, because all information is only propagated from the start of the network to the end - no loops exist and no

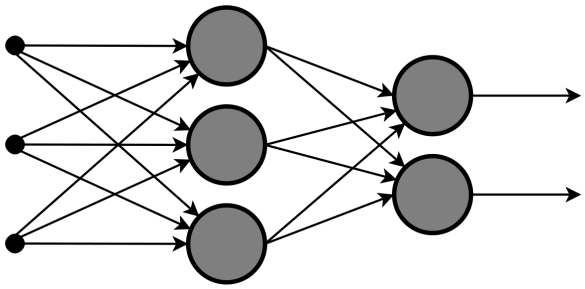

Figure 10.1: A very simple neural network architecture, consisting of neurons and connections between them [28].

information is transferred back to a previous layer or part of the network. Additionally, the example network is *fully-connected* (or *dense*), since all neurons in each layer are connected to all neurons in the following layer.

In the previous explanation, it was assumed that the activation is a linear combination of the inputs. In reality, the power of neural networks lies in their non-linearity. This non-linearity can be achieved by first calculating the linear combination y of inputs as in Eq. (10.1). Then, some *activation function* is applied to y: $z = f(y)$. The value z is then the final activation of the neuron. Activation functions are in general non-linear. Four widely used activation functions are shown in Fig. 10.2. Tanh and sigmoid are often used if the input needs to be scaled between for instance 0 and 1 or -1 and 1. Linear activation functions do not alter the neuron activations. The ReLU function ensures that the activations can only be non-negative. Sometimes, the ReLU function is adapted slightly to a leaky ReLU function: $f(x) = \max(\alpha x, x)$, where $\alpha \in (0, 1)$ is typically close to 0.

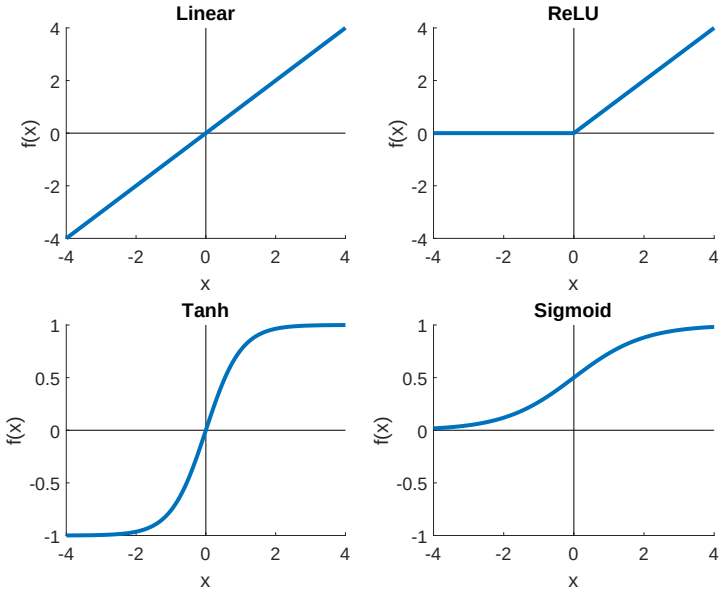

Figure 10.2: Four activation functions, adapted from [23]. The sigmoid function is defined by $\sigma(x) = \frac{1}{1+\exp(-x)}$. For ReLU, $\alpha = 1$ is taken.

Finally, the activations of the last layer of neurons - the output layer - provide the output value(s). In the training process, these output values are compared to the targets, and based on this comparison, the weights and biases in the network are altered. This optimisation continues until the outputs are no longer improving. The training process is described in more detail in the following section.

10.1.2. Training

As explained, the goal of training the neural network is to make sure any input can be converted as well as possible to the corresponding desired output. This is done by optimising the learnable parameters in the

network: the weights of each connection between neurons, and the biases of every neuron. The optimisation is based on a comparison between the output and the target (desired output). In particular, a *loss function* quantifies this: the closer output and target are, the smaller the loss. The loss function usually is a function of just the output and target. Training algorithms aim to minimise this loss $L(\mathbf{y}, \hat{\mathbf{y}})$, where $\mathbf{y}$ is the target and $\hat{\mathbf{y}}$) is the output. In regression problems, mostly the mean squared error (MSE) is used as a loss function:

$$L(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2, \tag{10.2}$$

where n is the length of the output vector.

At first, all weights and biases need to be initialised. Usually, it suffices to take for example random weights and zero biases, and let the training algorithm find its way in optimising them. If it turns out that the training does not progress initially, it might be needed to specifically assign initial learnable parameters. This was never needed during this research, hence a discussion of this topic is beyond the scope of the project.

Then, the training algorithm aims to minimise the loss $L(\mathbf{y}, \hat{\mathbf{y}})$ iteratively. To do this, it needs to know how changing the weights and biases a little influences the loss. In particular, the algorithm needs the gradient of the loss function to each learnable parameter p_i:

$$\frac{\partial L}{\partial p_i} = \sum_{j=1}^{n} \frac{\partial L}{\partial y_j} \frac{\partial y_j}{\partial p_i}. \tag{10.3}$$

The chain rule is used for the summation. This can be rewritten as the product of the Jacobian matrix of $\mathbf{y}$ with respect to $\mathbf{p}$ and the gradient of L to $\mathbf{y}$:

$$\frac{\partial L}{\partial p_i} = \left(\frac{\partial y}{\partial p} \right)^T \nabla_{\mathbf{y}} L. \tag{10.4}$$

The gradient can of course be calculated beforehand, if the loss function L is known. The Jacobian, however, needs to be determined every iteration. This can be done in an efficient way by using a *backpropagation* algorithm. When the gradient with respect to the learnable parameters is known, the algorithm knows in which direction to change these parameters to find a minimum.

One of the algorithms that performs the above process is stochastic gradient descent (SGD). Out of all training samples (combinations of inputs and targets), it chooses a few random ones that form a *minibatch*. For each of the samples, gradient descent is performed. In the opposite direction of the gradient, the loss function is minimised most. The gradient descent algorithm then moves a small distance in that direction. If all goes well, a minimum is found iteratively. This depends on how large the step size is taken - in machine learning also called the *learning rate*. As illustrated in Fig. 10.3, a learning rate that is too large may cause moving away from the minimum. An appropriate learning rate moves slowly into the minimum. A learning rate that is very small obviously causes a very long running time of the algorithm. Sometimes, it might be beneficial to use an adaptive learning rate, such as one that is large in the beginning, and becomes smaller towards the end of the optimisation. The SGD algorithm takes an average over all the individual directions of steepest descent of the samples, and moves in that direction. In this way, a too large influence of a single sample is avoided.

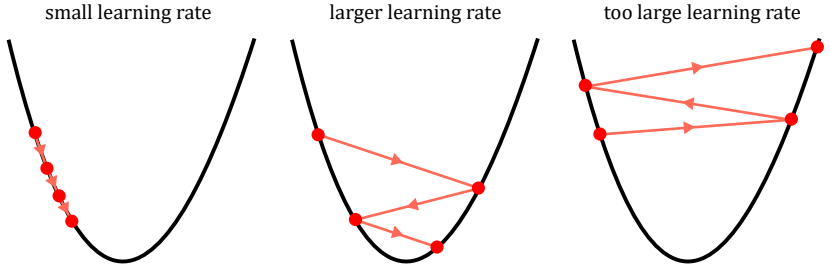

Figure 10.3: Illustration of the effect of the learning rate on gradient descent performance, adapted from [12].

In this research, the Adam training algorithm is used. This is an extension of the SGD algorithm. The main extra function of Adam is an adaptive learning rate that is individual per learnable parameter. It does so using estimates of the first and second moments of the gradients (mean and uncentred variance). Detailed information can be found in [22].

During training, a main risk is overfitting. In this context, it means that the network learns very well how to handle the training inputs, but at the same time performs way worse on similar but unseen data. To avoid this, a common practice is to use an independent training set and validation set. As long as the performance on data from the validation set increases (measured by a decreasing loss function), the training continues. If at some point the training performance keeps improving, while validation performance becomes worse for a certain number of consecutive iterations, the training must be stopped. Final performance of the network is always measured using test data that is independent from the training data.

10.2. Autoencoder

The *autoencoder* is a special type of neural network. It is designed to reproduce as its output the exact input. Typically, one hidden layer in the middle of the autoencoder network has less nodes then inputs. This means that all the information of the input is at some point stored in the activations of fewer hidden nodes. In other words, the network tries to summarise the input data. This is especially useful if for example data can be represented by a limited amount basis modes, as is the case for magnetic signatures of a ship. The neural networks used in this project are based on the concept of the autoencoder.

The autoencoder consists of an input layer and an equally sized output layer - since the input needs to be exactly reconstructed. In between, a few hidden layers are placed. A schematic of the network architecture is shown in Fig. 10.4. The activations in the bottleneck hidden layer are together called the *code*. The first half of the network is said to *encode* the input, and the second half *decodes* back to a signal. Typically, the number of neurons per layer is decreasing towards the middle.

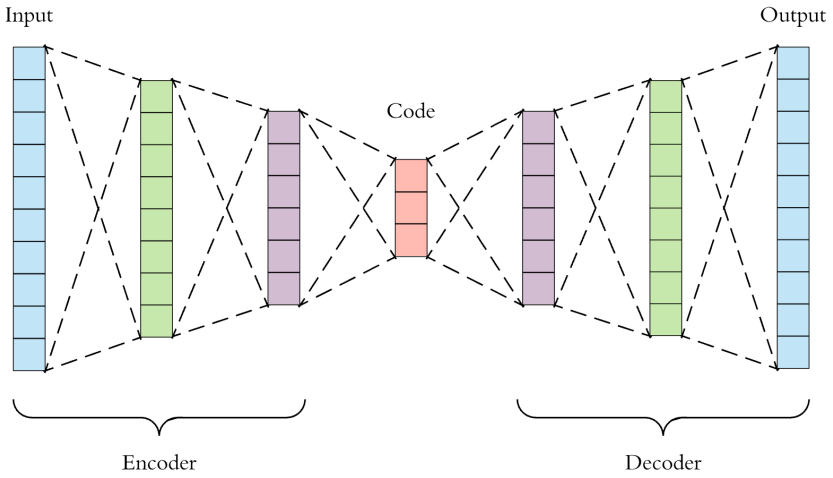

Figure 10.4: Schematic of an autoencoder network architecture [11].

The *denoising autoencoder* is a network similar in architecture, but it is used to process a noisy input. The input can be noisy in two ways: noise can be added to the individual elements of the input vector, and the input vector can be gappy. In practice the latter is equivalent to setting some of the input vector elements to zero. When training the denoising autoencoder, the idea is to start with an original input vector $\mathbf{x}$, which is then altered such that it becomes noisy: $\mathbf{x}_n$. The noisy input propagates through the network and creates a predicted output $\hat{\mathbf{x}}$. The output is compared to the original input $\mathbf{x}$, and based on this result, the network is trained.

At first, such a denoising autoencoder was used to reconstruct the full magnetic field above the ship from noisy and gappy measurements. As an adaptation, the network was trained with noisy and gappy input data

from above the ship as inputs, and the full field below the ship as targets. The setup for the sensors is equal to that for Gappy POD, as in Fig. 8.1. Just like in Section 9.4, the field vectors had the following structure:

$$\mathbf{x}_i = [B_{1,x}, B_{1,y}, B_{1,z}, B_{2,x}, \ldots, B_{n,z}]^T, \tag{10.5}$$

where $B_{i,x}$ is the x-component field measurement of the i^{th} sensor.

10.3. Implementation

In designing the neural networks, many parameters were tuned. Think of network architecture (number of layers and neurons) but also learning rate, activation functions and loss functions. For some of these, finding an optimal configuration is a matter of trial and error. For example, tuning the learning rate can be done by inspecting the learning curve - performance versus iterations. For other, more fundamental choices, different possibilities have been explored. An overview of these is given in the upcoming sections, and the results of the analyses is given in Section 10.4.

10.3.1. Network purpose, architecture, and activation functions

The networks in this research are divided in two groups: one with as purpose reconstructing the field above a ship using potentially noisy and gappy measurements above the ship. This is done using autoencoders. The second purpose is estimating the field below the ship using measurements above the ship. The latter are no longer autoencoders, but modified versions.

Many different architectures were tested, but all had some things in common. Each network consists of an input layer, a similarly sized output layer, and one or more hidden layers. If multiple hidden layers are used, the network is symmetric, and the number of neurons decreases towards the middle bottleneck layer which contains the code, similar to the schematic in Fig. 10.4. A larger code can be seen as a larger amount of basis fields considered, comparing to the Gappy POD method.

As activation functions, linear functions and leaky ReLU functions with slope 0.1 were used. If all activation functions in the network are linear, then the output is just a linear combination of the input. It is expected that results of such a network are therefore similar to Gappy POD, which is also solving a linear system. Using leaky ReLU function, a non-linearity is introduced in the system.

10.3.2. Loss function: physics guided neural network

As a standard regression loss function, almost always the mean squared error between output and target is used, see Eq. (10.2). Additionally, an extension to a physics guided neural network [20] was explored. We know that a valid output field of the network must adhere to some governing physics laws. In this situation, where we consider magnetism, these might be Maxwell's equations. Therefore, it might be beneficial to introduce a penalty in the loss function which becomes zero as the network output looks more like a valid magnetic field.

From Maxwell's equations, for the magnetic field at a sensor array outside of the ship, the following must hold:

$$\begin{aligned} \nabla \cdot \mathbf{B} &= 0; \\ \nabla \times \mathbf{B} &= \mathbf{0}. \end{aligned} \tag{10.6}$$

Additionally, under the ship, the magnetic field magnitude must decay when moving away:

$$\frac{\partial |\mathbf{B}|^2}{\partial z} \leq 0. \tag{10.7}$$

Note that the z-axis is defined to point in the downward direction. Of course, above the ship, this relation would be similar with $\geq$.

Not all equations from Eq. (10.6) can be checked when measuring in a horizontal plane. However, from the zero curl equation it immediately follows that

$$g(B) = \frac{\partial B_y}{\partial x} - \frac{\partial B_x}{\partial y} = 0 \tag{10.8}$$

at all points within the horizontal plane.

Rewriting Eq. (10.7), gives

$$B_x \frac{\partial B_x}{\partial z} + B_y \frac{\partial B_y}{\partial z} + B_z \frac{\partial B_z}{\partial z} \leq 0, \tag{10.9}$$

and using the fact that the field needs to be divergence-free, together with the zero curl, gives the following relation:

$$h(B) = B_x \frac{\partial B_z}{\partial x} + B_y \frac{\partial B_z}{\partial y} + B_z(-\frac{\partial B_x}{\partial x} - \frac{\partial B_y}{\partial y}) \leq 0. \tag{10.10}$$

This leaves us with two relations Eq. (10.8) and Eq. (10.10) that depend only on the horizontal derivatives of the field, which can be calculated. These must hold at every location in the horizontal plane. The loss function is extended using two additional physics terms:

$$L(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \alpha \sum_{j,k} (g_{j,k}(\hat{\mathbf{y}}))^2 + \beta \sum_{j,k} \text{ReLU}(h_{j,k}(\hat{\mathbf{y}})), \tag{10.11}$$

where j, k denote the x and y locations of the sensor array. $g_{j,k}$ denotes the evaluation of the function g at the location (j, k). For the calculation of its value, also other measurements may be used, hence the $\hat{\mathbf{y}}$ inside. The square in the first physics term is chosen such that the term has a minimum in 0. For the second physics term, it only needs to be optimised when it is larger than zero. For every negative value it should be zero, hence the ReLU function was chosen.

The derivatives are approximated using the forward difference method. This means that on the edges of the sensor array, the relations cannot be checked. For training purposes, also the gradient of the loss function with respect to each of the field components is needed. The gradient vector entries were also approximated using the forward difference scheme.

10.4. Results

In this section, the results of the implemented neural networks will be discussed. First, a comparison is made between linear and non-linear networks, and at the same time between a 'classic' loss function and a physics-guided loss function. Then, the number of nodes and sensors are varied, and the implications for the network performance are discussed. Lastly, the influence of varying the height of the upper and lower sensor array is analysed.

10.4.1. Linearity of the neural network and physics-guidance

In this section, the results are shown and analysed of applying the neural network for estimating the magnetic field below the ship. For these results, a network with the following structure was used: 100 - 50 - 10 - 50 - 100 hidden neurons. Now, a comparison is made between the results with linear activation function and ReLU activation function. Furthermore, the physics guided neural network with adapted loss function is compared to the standard regression network with mean squared error loss.

For the first analysis, a noisy and gappy input signal was used, where Gaussian noise with standard deviation 10 nT was added, and 10 percent of measurements was set to zero. In Fig. 10.5 and Fig. 10.6, the training and validation results are shown, respectively.

When looking at the training performance, it is clear that the physics loss function is not improving results at all. When checking the physics conditions for results from the network without physics guidance, they are indeed already satisfied. Clearly, the network learnt from the target data how the results should look, and copies that in the output. And since the targets are satisfying the derived physics rules, the outputs will too.

Other than that, the training performance of the network with nonlinear leaky ReLU activation functions between all neurons is better than that of the linear networks. However, when comparing the validation data, it becomes clear that this is in fact just overfitting. While the results are improved for the data that the network has seen during training, they are not for the unseen data. The validation results of the linear networks are even slightly better.

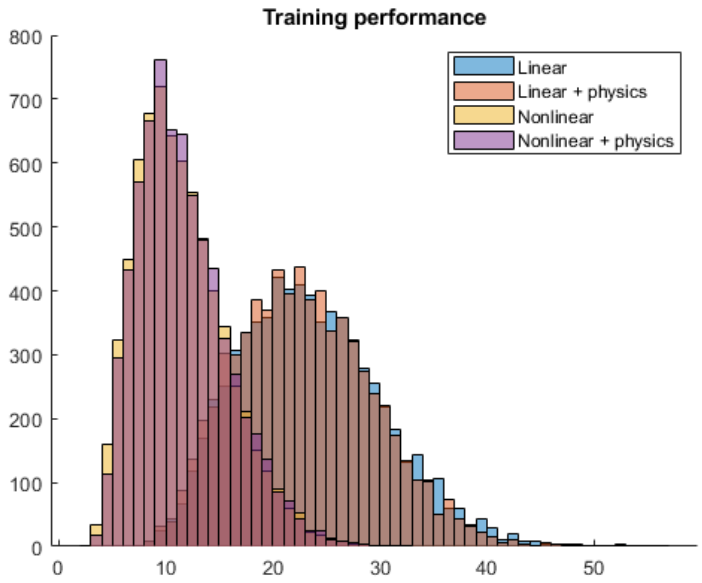

Figure 10.5: Training performance of neural networks with linear and nonlinear activation functions, and with and without physics terms in the loss function. A noise with standard deviation 10 nT is added, and 10 percent of measurements is set to 0. The histogram shows the occurrences of RMSE in nT of all samples in the training set.

For the second analysis, a noisy and gappy input signal was used, where Gaussian noise with standard deviation 50 nT was added, and 50 percent of measurements was set to zero. Fig. 10.7 and Fig. 10.8 show the training and validation results.

As expected, the errors are in general larger than for the previous scenario, because the measurements are 5 times more noisy and gappy. Other than that, the results are virtually the same. In the remainder of this chapter, only linear neural networks with a classic MSE loss function are considered. Judging from these first results, this network structure is most effective.

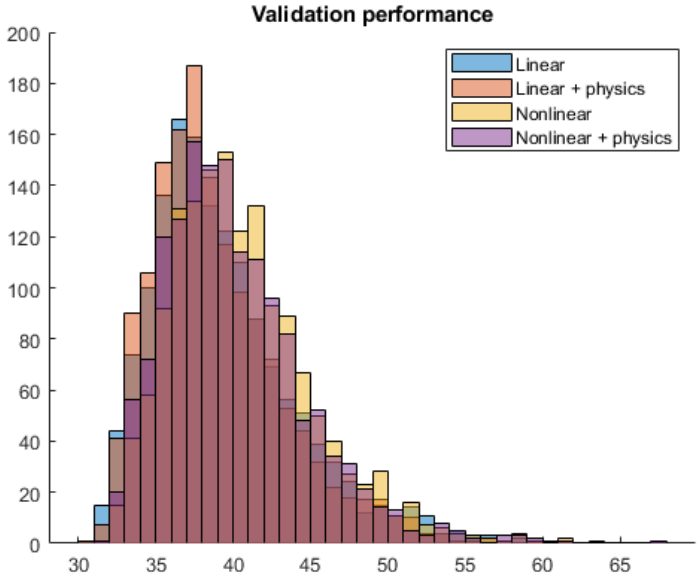

Figure 10.6: Validation performance of neural networks with linear and nonlinear activation functions, and with and without physics terms in the loss function. A noise with standard deviation 10 nT is added, and 10 percent of measurements is set to 0. The histogram shows the occurrences of RMSE in nT of all samples in the validation set.

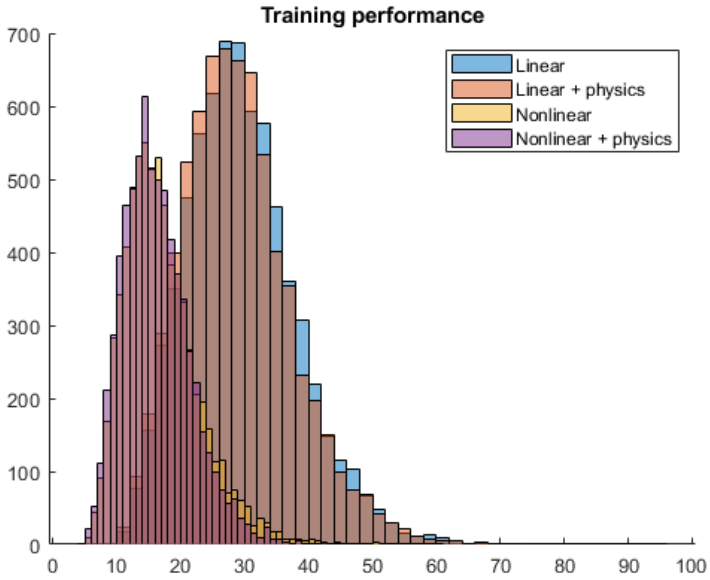

Figure 10.7: Training performance of neural networks with linear and nonlinear activation functions, and with and without physics terms in the loss function. A noise with standard deviation 50 nT is added, and 50 percent of measurements is set to 0. The histogram shows the occurrences of RMSE in nT of all samples in the training set.

10.4.2. Number of nodes and sensors

In the same way as in Section 9.5.3, the neural network performance is analysed for varying number of sensors, and for a varying number of nodes in the internal layer of the neural network. Note that the linear network that is considered here consists of an input layer of full size, an internal layer of varying size, and an output layer of full size. The activation functions between nodes of these layers are linear.

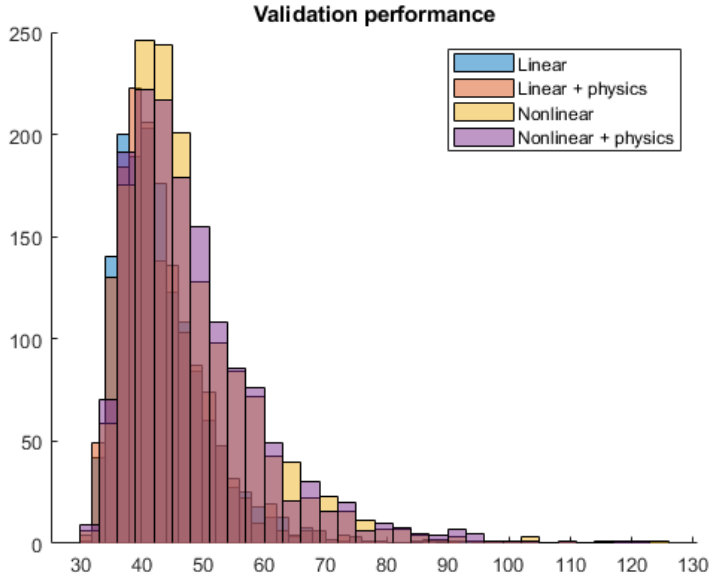

Figure 10.8: Validation performance of neural networks with linear and nonlinear activation functions, and with and without physics terms in the loss function. A noise with standard deviation 50 nT is added, and 50 percent of measurements is set to 0. The histogram shows the occurrences of RMSE in nT of all samples in the validation set

The number of internal nodes can be seen as similar to the number of modes considered in Gappy POD: they both determine how many degrees of freedom are available when fitting a field to the available measurements. If there are three internal nodes, there are effectively only three 'basis fields' that the network will try to fit. In this way, the results are comparable to the results of Gappy POD in Fig. 9.9 to Fig. 9.12.

In Fig. 10.9, the results for varying number of internal nodes and sensors are shown. Each block in the figure represents one trained neural network with corresponding number of internal nodes, trained on input data from the corresponding number of sensors. The other input data is set to zero, just as in Section 10.4.1. The NRMSE value that is shown is the average NRMSE over the validation set, which is independent from the training set.

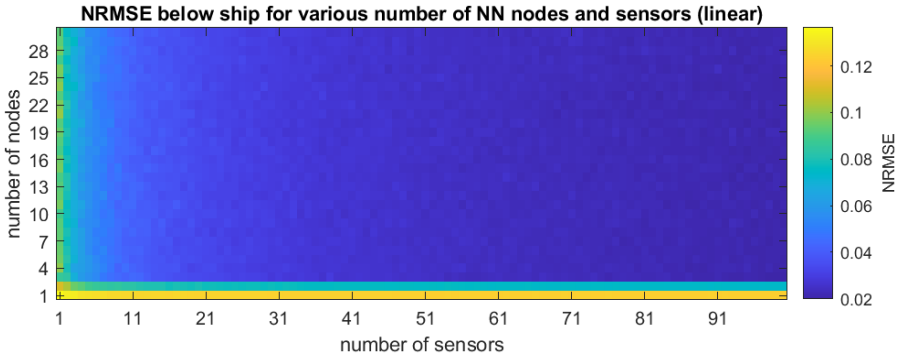

Figure 10.9: NRMSE error of the field the below ship for a varying number of nodes in the internal layer and number of sensors. A linear neural network with random sensor choice is used for reconstruction. Each block displays the average NRMSE of the validation set after training.

Of course, the results for only one or two internal nodes (similar to one or two basis modes) are not great. As seen in Fig. 8.2, the data contains three basis modes and hence three degrees of freedom are needed. Apart from that, it is clearly visible that results improve when the number of sensors increases. This makes sense, because a larger number of measurements gives more certainty on the actual magnetic field. Noise can also be separated better from the signal when more measurements are available. A comparison to the results of Gappy POD presented in Section 9.5.3 will be made in Section 10.5.

10.4.3. Height of sensor arrays

Again in the same way as for Gappy POD in Section 9.5.4, various heights of the upper and lower sensor array are compared. The results have been produced with a linear neural network with 10 internal nodes, and 10 sensors that produced measurements (30 measurements). And again, the NRMSE values are the average NRMSE of the validation set. These results are shown in Fig. 10.10.

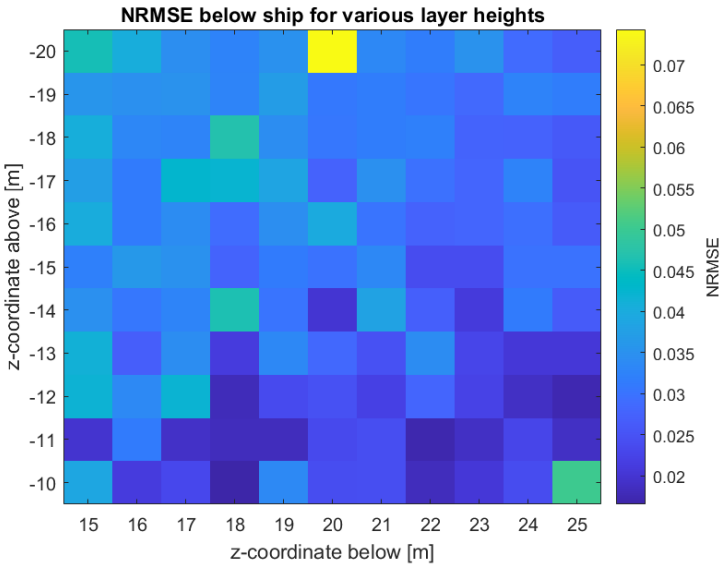

Figure 10.10: NRMSE error of the field below the ship for a range of heights for the upper and lower sensor array. The left bottom corner corresponds with both sensors arrays close to the ship. Each block displays the average NRMSE of the validation set after training of the neural network.

Overall, all errors are quite small. It is interesting to note that there is one larger error, for the sensor arrays 20 m below the origin and 20 m above. It is not clear what causes this deviation. In general, the errors are smaller for sensor arrays closer above the ship. Apparently, the neural network can handle local effects well, and use them for a better estimate below the ship. This is in contrast to the performance of Gappy POD, as explained further in Section 10.5.

10.5. Comparison between neural networks and Gappy POD

In this section, we will discuss the differences and similarities between Gappy POD and linear neural networks. First, the practical implications are discussed: what needs to be done for both methods to be brought in practice? After that, the relations between the results from the simulations as discussed in Section 9.5 and Section 10.4 are analysed.

10.5.1. Practical implications

Both the Gappy POD method and neural networks need initialisation. In this thesis, the methods are initialised using simulation data. To train the neural network, noise-free and non-gappy data is needed. Therefore, it is hard to do this with only field measurements, and it may be an obstacle when only measurements are available. If no model to generate data is available, it might be a better choice to use Gappy POD. When constructing the basis using the SVD, in principle noisy data can be used. The noise will be represented in some higher order basis modes resulting from the SVD. If a proper regularisation is used, such as the Tikhonov regularisation proposed in this work, this does not need to be a problem in applying Gappy POD.

In initialisation, training a neural network costs significantly more time and computing power than initialising the Gappy POD basis using the SVD. However, all of this work can be done beforehand using simulated or real life measured data, so the impact is not large. However, if in a true application of these methods updates

need to be done to the basis or network on the fly, it may be a factor of interest. This can for example be the case to correct for model errors when the training or initialisation data was generated.

Applying the methods to measurements is somewhat less efficient in Gappy POD, since a linear system needs to be inverted. The network creates a linear function between input and output, of which the result is easier to computer. Unless the Gappy POD basis is enormous, however, this will not be a problem.

10.5.2. Simulation results

The results of varying number of sensors and number of modes (in Gappy POD) or internal nodes (in neural networks) are remarkably similar if we compare Gappy POD including Tikhonov regularisation. Looking at the two relevant figures, Fig. 9.10 and Fig. 10.9, the results are almost the same, except for some effects of randomness in the presented fields. This means that in the process of training the linear neural network, some automatic regularisation is applied.

That effect is probably due to how input and target data are presented to the network during training. The input data is always noisy and gappy, but each time with a different noise and different gaps. The target data, however, is always 'perfect': it is the full expected field, and without noise. After a while, the network learns to ignore the noise in the data. In the training process, the internal parameters of the network (biases and weights), are only altered slightly in the right direction after a new set of inputs and targets. This is due to the principle of stochastic gradient descent. If noise or gappyness would influence the parameters in one way (overfitting), the next set of inputs and targets will perhaps try to influence them the other way. Over a large number of inputs, the influence of noise can be averaged out in this way.

In varying heights, there is some difference between the two methods. We compare the results in Fig. 9.13 and Fig. 10.10. Two effects are interesting. First, the area where errors are small are for Gappy POD more located where the distance of the sensor array above the ship is larger. However, for the linear neural network, the errors are smaller when this sensor array is closer to the ship. Second, there are some outliers with a significantly larger error than for other array heights. For Gappy POD, these are when the sensor array above the ship is very close. In the neural network results, there is one outlier, where the sensor array above is far away. These locations corresponds to the general error results. When analysing the error scales closely, the larger error at -20 m and 20 m is occurring in both figures. The only difference between the figures seems to be caused by a different way of regularisation. The neural network is able to handle local effects in the field better than Gappy POD.

11

Conclusions and recommendations

11.1. Conclusions

For magnetic signature reconstruction of ships, two different approaches were taken. The first of these is the implementation of a Gappy POD algorithm. In this algorithm, magnetic field data of a ship, resulting from simulations, is used to find orthogonal modes. Only a restricted number of these are used in the remainder of the algorithm. After this initialisation, these modes are fitted to gappy and noisy measurement data. Just as in the localisation algorithm, a method for choosing good sensors was included. This time, it focuses on minimising the condition number of the matrix used in the system that must be solved. As an additional feature, Tikhonov regularisation was added as an option. When fitting a large number of modes to noisy data, a risk of overfitting arises, and Tikhonov regularisation aims to prevent this from happening.

Overall, the algorithm is able to accurately reconstruct the magnetic field of a ship, especially when Tikhonov regularisation is used in the process. This was tested using simulated data of a simplified model of a ship. Results varied based on the number of sensors used and how they were chosen, the number of modes considered, height of the sensor arrays, whether Tikhonov regularisation was used, and measurement noise.

For Gappy POD without Tikhonov regularisation, the results are quite well in the case where the number of considered basis modes is very small (10 or smaller) and the number of used sensors is not too small (larger than 5). Performance when 5% noise is added to the measurements shows a NRMSE of about 10%. However, the results quickly become worse outside this specific area, with even a NRMSE above 1 for large numbers of modes in combination with small numbers of sensors. In this situation, the method is overfitting on the noisy data. This behaviour is completely solved by applying Tikhonov regularisation in the process. Modes with a small average energy in the original data are not considered equally to modes with a large average energy. Results show that the NRMSE when Tikhonov regularisation is included is as low as 2% to 4%.

Secondary findings from the Gappy POD results include that reconstruction performance below the ship is better when the sensor array above the ship is not placed too close. This way, the field does not contain too many local effects, improving the estimation of basis mode coefficients. Also, increasing the measurement noise decreases reconstruction accuracy, as expected.

Second, a neural network was designed and trained. The architecture was based on that of an autoencoder, with layers of neurons that create two funnels towards a centre layer with the least neurons. With the amount of neurons in the middle layer, one can control how much information is allowed to be stored - similar to choosing the number of modes in Gappy POD. Four variations were designed and analysed: combinations using linear and non-linear activation functions, and with or without physics guidance. For the non-linear activation function, a leaky ReLU function was used. Physics guidance was created using a custom loss function. If the output field is not satisfying known physics rules for magnetism, a penalty is applied, forcing the network to learn fields that satisfy physics. In this way, the neural network turns from a fully data-driven method into a hybrid method: both data and underlying physics are used.

In equal conditions, the non-linear neural network performs better than its linear counterpart in training. However, its validation performance is slightly worse than for the linear network. This indicates overfitting: the non-linear network is able to better learn how to process the given inputs, but this is less generalisable to unseen inputs.

Neural network performance does not change when adding physics guidance. Even when no physics rules are given to the network, it learns what a valid field looks like. This is possibly due to combining several basis modes in the final output: if each of those fields satisfy physics rules, then the linear combination will too. This behaviour was confirmed by checking the physics penalty terms, which were indeed (close to) zero.

The network performance does not differ much when altering the number of sensors or the number of middle layer neurons. Overall, performance is better than Gappy POD without Tikhonov regularisation. This is true especially when a large number of modes is chosen in Gappy POD, corresponding to a large number of middle layer neurons. The neural network results remain consistent relative to a smaller number of 'modes', while the Gappy POD error increases spectacularly. Apparently, some regularisation is automatically applied when training the network. However, the neural network performance is worse than Gappy POD with Tikhonov regularisation.

The variation of number of internal nodes and number of sensors in a linear neural network gives results very similar to Gappy POD with Tikhonov regularisation. The network turns out to be capable of recognising noise in the training input, and avoid overfitting. This automatic regularisation comes from the way of training the network: noisy and gappy data is fed into the network, but the target data is always noise-free and gap-free. After being confronted with many noisy and gappy inputs, and perfect output data, the network is after a while able to ignore input noise.

11.2. Recommendations

First, this thesis is somewhat theoretical, and not yet going into the actual application. The final goal of the signatures part of this thesis would be to use actual magnetic field measurements conducted from an aerial drone to estimate signatures. For that, some steps need to be taken. The signal will be measured in time, so some processing needs to be done to get a spatial signal. This is not trivial, since speed can have a significant impact on the signal in time, and errors in location data must be considered. Also, the height of the drone may vary, so measurements in a 3D space need to be considered. This is of course an opportunity as well: with height included, it can be very interesting to look at what optimal paths around a ship can be taken to estimate its signature as well as possible.

Additionally, it would be interesting to look into time series neural networks and convolutional neural networks to catch hysteresis effects. In this thesis, just a brief exploration of non-linear neural networks was done. However, deep structures were not considered. Using these, it might for example be possible to involve the expected hysteresis effects in signature estimations.

Finally, throughout this thesis, only white noise has been considered as sensor noise. In real life, other errors can play a role: there can be other magnetic sources, or there can be errors in location data. To compensate for these is a lot harder than to compensate for white noise, which can eventually be averaged out. If a structural error is present, it needs to be determined whether it belongs to the ship or is indeed an error that needs to be compensated for. In solving these problems, the physics-guided neural network may come into play. Further development of that model could potentially result in better error detection and suppression in the final estimates.

Bibliography

[1] AHMED, N., NATARAJAN, T., AND RAO, K. R. Discrete Cosine Transform. *IEEE Transactions on Computers C-23*, 1 (1974), 90–93.

[2] BALDWIN, J. Rayleigh hysteresis - A new look at an old law. *IEEE Transactions on Magnetics 14*, 2 (1978), 81–84.

[3] BALL, P. The tyranny of simple explanations. https://www.theatlantic.com/science/archive/2016/08/occams-razor/495332/, aug 2016.

[4] BARANIUK, R., DAVENPORT, M., DEVORE, R., AND WAKIN, M. A simple proof of the restricted isometry property for random matrices. *Constructive Approximation 28*, 3 (2008), 253–263.

[5] BJÖRCK, Å. *Numerical methods for least squares problems*. Society for Industrial and Applied Mathematics, 1996.

[6] BJÖRCK, Å. *Numerical methods in matrix computations*, vol. 59 of *Texts in Applied Mathematics*. Springer, 2015.

[7] BRUNTON, S. L., AND KUTZ, J. N. *Data-driven science and engineering: machine learning, dynamical systems, and control*. Cambridge University Press, 2019.

[8] CANDÈS, E. J. The restricted isometry property and its implications for compressed sensing. *Comptes Rendus Mathematique 346*, 9-10 (2008), 589–592.

[9] CANDES, E. J., AND WAKIN, M. B. An introduction to compressive sampling. *IEEE Signal Processing Magazine 25*, 2 (2008), 21–30.

[10] CLARK, E., KUTZ, J. N., AND BRUNTON, S. L. Sensor selection with cost constraints for dynamically relevant bases. *IEEE Sensors Journal 20*, 19 (2020), 11674–11687.

[11] DERTAT, A. Applied deep learning - part 3: Autoencoders. https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798, oct 2017.

[12] DONGES, N. Gradient descent: an introduction to 1 of machine learning's most popular algorithms. https://builtin.com/data-science/gradient-descent, jun 2019.

[13] ECKART, C., AND YOUNG, G. The approximation of one matrix by another of lower rank. *Psychometrika 1*, 3 (1936), 211–218.

[14] GRANT, M., AND BOYD, S. Graph implementations for nonsmooth convex programs. In *Recent Advances in Learning and Control*, V. Blondel, S. Boyd, and H. Kimura, Eds., vol. 371 of *Lecture Notes in Control and Information Sciences*. Springer-Verlag Limited, 2008, pp. 95–110.

[15] GRANT, M., AND BOYD, S. CVX: Matlab Software for Disciplined Convex Programming, version 2.2. http://cvxr.com/cvx, 2014.

[16] GRIFFITHS, D. J. *Introduction to Electrodynamics*, 4th ed. Prentice Hall, Upper Saddle River, N.J., 2017.

[17] HOLMES, J. J. *Exploitation of a ship's magnetic field signatures*, vol. 9. Morgan & Claypool, 2006.

[18] JACKSON, J. D. Magnetostatics, Faraday's law, quasi-static Fields. In *Classical Electrodynamics*, 3rd ed. Wiley, New York, N.Y., 1998, ch. 5.

[19] JIN, H., GUO, J., WANG, H., ZHUANG, Z., QIN, J., AND WANG, T. Magnetic anomaly detection and localization using orthogonal basis of magnetic tensor contraction. *IEEE Transactions on Geoscience and Remote Sensing 58*, 8 (2020), 5944–5954.

[20] KARPATNE, A., WATKINS, W., READ, J., AND KUMAR, V. Physics-guided neural networks (pgnn): An application in lake temperature modeling. *ArXiv abs/1710.11431* (2017).

[21] KE, W., ZHANG, X., YUAN, Y., AND SHAO, J. Compressing sensing based source localization for controlled acoustic signals using distributed microphone arrays. *Mathematical Problems in Engineering 2017* (2017).

[22] KINGMA, D. P., AND BA, J. Adam: a method for stochastic optimization. In *ICLR* (2015).

[23] KOBRAN, D., AND BANYS, D. Activation Function. https://docs.paperspace.com/machine-learning/wiki/activation-function, 2020.

[24] MEHROTRA, S. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization 2*, 4 (1992), 575–601.

[25] MORANDI, A., FABBRI, M., AND RIBANI, P. L. A modified formulation of the volume integral equations method for 3-D magnetostatics. *IEEE Transactions on Magnetics 46*, 11 (2010), 3848–3859.

[26] NOS. Container met gifstoffen uit MSC Zoe gelokaliseerd. https://nos.nl/artikel/2278154-container-met-gifstoffen-uit-msc-zoe-gelokaliseerd, mar 2019.

[27] NOS. Laatste restanten MSC Zoe-containers in Waddenzee opgeruimd. https://nos.nl/artikel/2327802-laatste-restanten-msc-zoe-containers-in-waddenzee-opgeruimd, mar 2020.

[28] OFFNFOPT. A neural network with multiple layers. https://commons.wikimedia.org/wiki/File:Multi-Layer_Neural_Network-Vector.svg, apr 2015.

[29] PAPERNO, E., SASADA, I., AND LEONOVICH, E. A new method for magnetic position and orientation tracking. *IEEE Transactions on Magnetics 37*, 4 (2001), 1938–1940.

[30] QAISAR, S., BILAL, R. M., IQBAL, W., NAUREEN, M., AND LEE, S. Compressive sensing: From theory to applications, a survey. *Journal of Communications and Networks 15*, 5 (2013), 443–456.

[31] SHANNON, C. E. Communication in the presence of noise. *Proceedings of the IRE 37*, 1 (1949), 10–21.

[32] SPIEGEL, M. R., LIPSCHUTZ, S., AND LIU, J. *Schaum's outline of mathematical handbook of formulas and tables*, 3rd ed. McGraw-Hill, 2008.

[33] TOH, K.-C., TODD, M. J., AND TÜTÜNCÜ, R. H. SDPT3 — a Matlab software package for semidefinite programming. *Optimization Methods and Software 11* (1999), 545–581.

[34] TOH, K.-C., TODD, M. J., AND TÜTÜNCÜ, R. H. On the implementation and usage of SDPT3– A MATLAB software package for semidefinite-quadratic-linear programming, version 4.0. *International Series in Operations Research and Management Science 166* (2012), 715–754.

[35] TROPP, J. A. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information Theory 50*, 10 (2004), 2231–2242.

[36] VAN DEN BERG, J. Hoe de MSC Zoe 342 containers verloor (en de kapitein het pas vijf uur later doorhad). https://www.volkskrant.nl/nieuws-achtergrond/hoe-de-msc-zoe-342-containers-verloor-en-de-kapitein-het-pas-vijf-uur-later-doorhad~b38fe9af/, jun 2020.

[37] WILLCOX, K. Unsteady flow sensing and estimation via the gappy proper orthogonal decomposition. *Computers & Fluids 35*, 2 (2006), 208–226.

[38] WILSON, C. A. From Kepler's laws, so-called, to universal gravitation: Empirical factors. *Archive for History of Exact Sciences 6*, 2 (1970), 89–170.

[39] ZUREK, S. Magnetic domains in a single grain. https://www.e-magnetica.pl/doku.php/magnetic_domain, apr 2021.