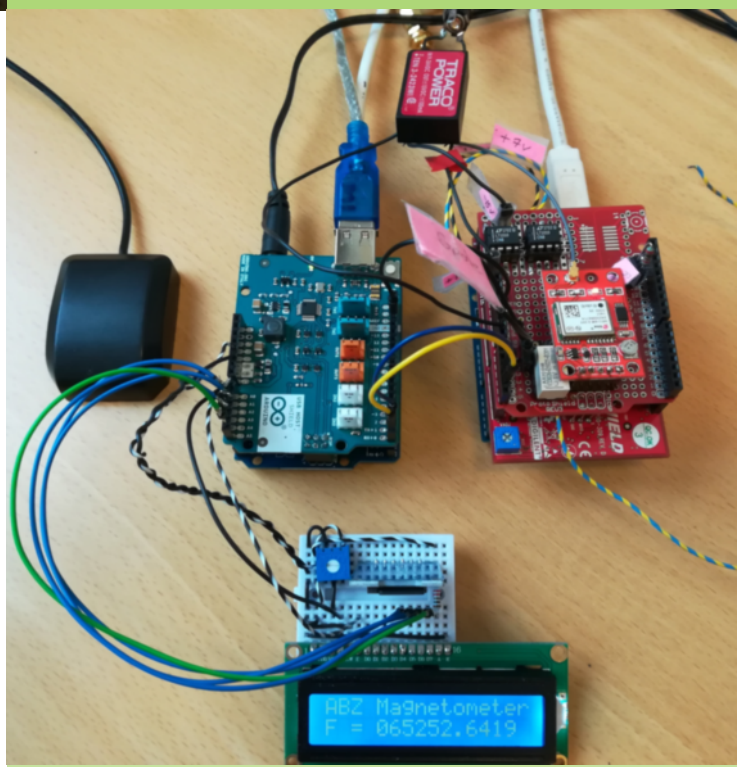
Optimizing the ABZ magnetometer by solving the inverse problem generated by the continuous varying current in the Z-coil

Master's Thesis

Tobias Bjerg
July 29, 2018

Supervisors:
Lars William Pedersen
Chris Finlay



DTU Space
National Space Institute

DTU Space
National Space Institute
Technical University of Denmark

Elektrovej, building 327
DK-2800 Kgs. Lyngby
Phone (+45) 4525 9500
Fax (+45) 4525 9575

www.space.dtu.dk

Abstract

The most common approach for measuring the geomagnetic field at an observatory is by means of a relative vector magnetometer and a D.I. flux theodolite. It usually involves a weekly baseline measurement by a trained specialist, and is therefore constrained by the accessibility of the observatory. This master thesis make an effort to enhance the development of the automatic absolute ABZ magnetometer. The basic idea of the ABZ magnetometer is a scalar magnetometer inside a freely hanging vertical-coil, which enables absolute estimate of the horizontal, vertical and total magnetic intensities.

A new approach is presented to the determination of the geomagnetic elements using the ABZ magnetometer. This involves first estimating the misalignment and sensitivity of the vertical-coil. This information is then used to estimate the horizontal, vertical and total magnetic intensities, by solving the inverse problem related to the continuous varying current in the coil.

Tests were carried out to demonstrate how the coil misalignment and sensitivity can be estimated. These revealed an unexpected coil current offset of $-1.4\,\mathrm{mA}$. This stray coil current offset has not yet been accounted for. By use of a simple test set-up similar to ABZ magnetometer, but with a bigger coil diameter, it is successfully demonstrated that absolute estimates of the horizontal and vertical intensity can be obtained. It was possible to conduct a field prediction within ± 10 nT, and with a 3 minutes estimation window.

Related to the ABZ magnetometer, a computer system has also been developed. The computer system arranges a synchronization of a controlled measurable coil current and data from the scalar magnetometer. The most important factors are a multiple serial-channel, GPS time-stamps and to improve the coil current measurements.

A new proposal to improving the regular determination of observatory absolute baselines, is also investigated. This builds on an idea from Soloviev et al. (2018), where a routine baseline improvement is made by applying additional information of the total vector field. The proposal is demonstrated on data from the geomagnetic observatory in Thule. This proposal could be extended with information from the ABZ magnetometer, as a possible application.

Preface

This master's thesis (35 ECTS credits) was prepared at the Department of Geomagnetism, National Space Institute at the Technical University of Denmark. The thesis is representative for the requirements for acquiring a master's degree in Earth and Space Physics and Engineering. The study was supervised by Chris Finlay and Lars William Pedersen both from the National Space Institute, DTU.

July 29, 2018

Tobias Bjerg (s124309)

Acknowledgements

First of all, thanks to Lars and Chris for highly qualified technical support, it have been extremely motivating to work with you. Somehow you always have time for a discussion about the project, which has been very helpful. With 12 trips to Brorfelde, I have discovered that you can fit an endless amount of equipment in a Citroën C3, thanks for the ride Lars.
It would not have been any fun to do the thesis, without my fellow student. A good chat and loads of coffee have help me through long days of data processing, I hope the good atmosphere will continue among the space students. Niels Skødt, Rasmus Joost, Ida Egdalen, Mick Kolster and Kristoffer Runge I really appreciate your help, if it is proof reading or just a coffee it is nice that I always can count on you. Thanks to Lukas Christensen for helping with the computer system. Thanks to Anna Naemi Willer with the Baseline improvement. Lastly, I would like to give thanks to Veronica, whom provide me with a lot of motivation.

Contents

List of Figures

List of Tables

1

Introduction

The geomagnetic field is one of the most important information sources we have to explore the Earth's interior, measurements of the geomagnetic field have been carried out for centuries. The first big geomagnetic breakthrough gave the possibility to navigate the world, with no more than a magnetic needle. However, the geomagnetic field can tell much more than direction. Today, a wide spectrum of measuring techniques grant information on everything from solar wind driven currents to flow of liquid iron in Earth's outer core. The opportunities can seem limitless, but in order to get more geomagnetic information, the instrumentation has to follow the scientific demands.

Many different instruments and measurement platforms have been developed with the intention of obtaining better geomagnetic data. Here among the highly advanced Swarm satellite constellation, which produces global data coverage with its three orbiting satellites. Despite many advantages of satellite measurements, there is still a demand for stationary ground observatories. The demand comes from the limitation of satellites. Where satellites provide global coverage, the stationary ground observatories give local continuous data of the geomagnetic field. Furthermore, stationary ground observatories have been measured continuously for more than 100 years, compared to the satellites that do not exceed 20 years' continuous data. The organization INTERMAGNET is a high quality real-time magnetic observatory network, where the first geomagnetic information node was established in 1991 (`Intermagnet.org`). With approximately 170 magnetic observatories, INTERMAGNET has a good stationary coverage, the data is publicly available and is used both in the private and public sector worldwide.

With the increasing quality in instrumentation, INTERMAGNET raises the bar for the measurement standards. In 2005, an INTERMAGNET user community expressed a desire for one-second data, where only one-minute data had been available before. Attached to this a minimum standard of instrument performance and data quality was required, which demanded new higher standards for the instrumentation (Turbitt et al., 2013). Even though we still see a growing demand for better data acquisition and data dissemination, automatic absolute measurement is rare within stationary ground observatories. The most common approach is by use of a D.I. flux theodolite (Declination Inclination flux theodolite) together with a relative vector geomagnetic instrument, e.g. the FGE (Pedersen and Merenyi, 2016). The D.I. flux theodolite provides a highly precise robust absolute measure of the geomagnetic field, called the baseline, which enables calibration of a relative measurements. However, the

D.I. flux theodolite is a manual instrument and baseline measurements have to be made on the site. The manual baseline measurement has to be done at least once a week to meet the INTERMAGNET requirements.

Automatic absolute measurement of the geomagnetic field is no new science, and the big struggle belongs to determining the declination. The ABZ magnetometer lacks information about declination and can only measure F, H and Z. However, without the declination, a number of important applications are still achievable and this is where the ABZ magnetometer could make an important contribution.

A classical structure is implemented for this thesis report. Firstly, the associated background theory is presented in the "Theory", followed by the method implemented in "Method". The results are shown in "Results". Lastly, the report is rounded off with a "Discussion" and "Conclusion".

A number of appendixes are included, they contain the implementation of computer system, and codes for both the computer system and Matlab script for the data processing. The report will therefore aim to describe ABZ magneometer on a conceptual level, while the appendixes will document the details of how it was implemented. All the material for implementation can also be found on: `ftp://ftp2.space.dtu.dk/pub/ABZ_Magnetometer/`.

1.1 General description of the geomagnetic field

The scientific world has developed various of mathematical schemes to represent the geomagnetic field. For stationary ground observatories, it is highly appropriate to use a Cartesian coordinate system. In Figure 1.1 the geomagnetic field **B** is a three-dimensional vector field, defined by the Cartesian vector element X, Y and Z. Furthermore, the description is extended with four other possible geomagnetic elements H, F, Dcl and Icl.

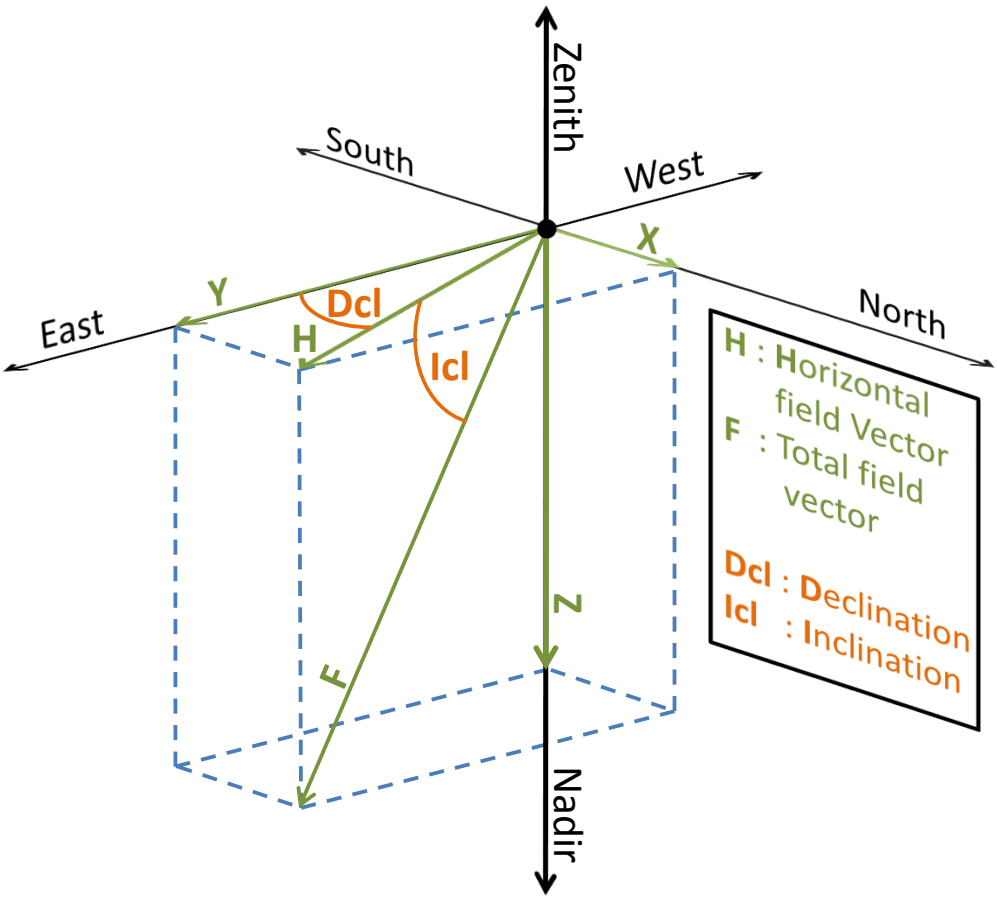


Figure 1.1: *The syntax of the geomagnetic elements. (Bjerg, 2017b)*

In order to describe the geomagnetic field, three independent elements are sufficient, and the rest can be derived from these three independent elements. In Equation 1.1 interrelationships between the geomagnetic elements are described.

$$
\begin{aligned}
X &= H\ \cos(Dcl) \\
Y &= H\ \sin(Dcl) \\
Z &= H\ \tan(Icl) = F \sin(Icl)
\end{aligned}
$$

$$
\begin{aligned}
H &= \sqrt{X^2 + Y^2} \\
F &= \sqrt{X^2 + Y^2 + Z^2}
\end{aligned}
\tag{1.1}
$$

$$
\begin{aligned}
Dcl &= \tan^{-1}\left(\frac{Y}{X}\right) \\
Icl &= \tan^{-1}\left(\frac{Z}{H}\right)
\end{aligned}
$$

The combination F, H and Z does not involve mutually independent elements and it will therefore not be a complete description of the geomagnetic field. However, this does not mean that F, H and Z are useless information.

A former INTERMAGNET observatory is the Danish observatory in Brorfelde. The measurements in this thesis are primarily done in Brorfelde and the values for geomagnetic elements stated in Table 1.1 can be used as an approximate reference.

X	Y	Z	H	F	Dcl	Icl
17.188 nT	797 nT	47.000 nT	17.207 nT	50.051 nT	3° 39'	69° 53'

Table 1.1: *Observed geomagnetic elements in the Danish observatory in Brorfelde in January 2015. (Finlay and Olsen, May 4, 2017)*

In general the declination and inclination is not changing significantly within small distances. Thus if a constant offset is measured in the total intensity, the offset in X, Y and Z can be calculated by fixing the declination and inclination.

When a local magnetic field is described by the magnetic elements from Equation 1.1, it will contain all magnetic signal at the present location. When discussing the geomagnetic field it will be refer to as H_E, F_E Z_E etc.

1.2 Concept of the ABZ magnetometer

The main purpose of the ABZ magnetometer is to derive the geomagnetic elements H_E, Z_E and F_E. The ABZ magnetometer is in its simplest form a scalar magnetometer inside a freely hanging vertical-coil system (See Figure 2.1). When exciting the coil system with a current it affects the scalar magnetometer with a homogeneous magnetic field. The magnetic field from the coil system will be a product of coil sensitivity ($S\,[\frac{nT}{mA}]$) and the coil current ($I\,[mA]$). The coil sensitivity will be nearly constant, and only change slow over long periods. By a measurable change in the coil current the absolute

geomagnetic horizontal (H_E), vertical (Z_E) and total intensity (F_E) can therefore in principle be derived. The deduction can be complex, and that is why a large part of this thesis addresses H_E, Z_E and F_E estimation. In general this involves solving the inverse problem generated by the continuous varying coil current.

No documented attempt of construction or testing a similar setup was found in the literature. The closest related is a more complex version using a tri axial coil system, as evaluated in the Ph.d. Zikmund (2014). But the geometry of a tri axial coil system is different and we do not find the same challenges in the ABZ magnetometer. Hence, we had to start from scratch. The formulation of the concept started with a special course project in June 2017 (Bjerg, 2017a), where the basic idea was investigated. After that a synthesis project about the fundamental construction and testing was conducted (Bjerg, 2017b). These two previous projects leads to the work presented in this report and are available on request.

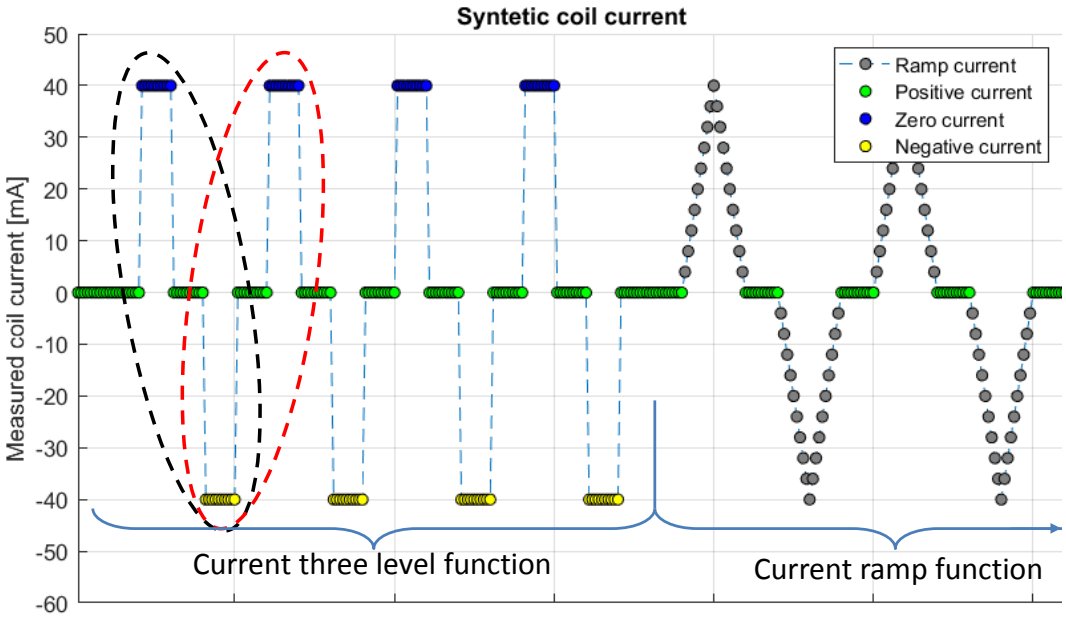


Figure 1.2: *Illustration of the two coil current functions used in the ABZ magnetometer.*

The coil current is an important aspect of the system, and we will use this opportunity to introduce two fundamental coil current functions that are referred to in the rest of the thesis. These two coil current functions are named current three level function and current ramp function they are presented in Figure 1.2 in a synthetic illustration.

Instrument performance and data quality

It is later stated in "Goal for Project" that we strive to achieve INTERMAGNET standards. For the purpose of evaluating the result a definition of specification parameters is listed in Table 1.2. Be aware that these data specifications

are ambitious, as it is also stated in Turbitt et al. (2013). But high ambitions are necessary to ensure that future observatory data is well suited for future applications.
It is not expected to achieve these specifications on the ABZ magnetometer performance and data quality. However, with the specifications in mind, pitfalls of the ABZ magnetometer can be located.

Table 1.2: *One-second Definitive Data Specifications (Turbitt et al., 2013)*

INTERMAGNET One-second Definitive Data Specifications

General Specifications	
Time-stamp accuracy	0.01 s
Phase response	±0.01 s
Maximum filter width	25 seconds
Instrument Amplitude Range	≥ ±4000 nT High Latitude, ≥ ±3000 nT Mid/Equatorial Latitude
Data resolution	1 pT
Pass band	DC to 0.2 Hz
Maximum component orthogonality error	2 mrad
Maximum Z-component verticality error	2 mrad

Pass Band Specifications [DC to 8 mHz (120 s)]	
Noise level	≥100 pT RMS
Maximum offset error	±2.5 nT
Maximum component scaling & linearity error	0.25 %

Pass Band Specifications [8 mHz (120 s) to 0.2 Hz]	
Noise level	≤ 10 pT /$\sqrt{}$Hz at 0.1 Hz
Maximum gain/attenuation	3 dB

Stop Band Specifications [≤ 0.5 Hz]	
Minimum attenuation in the stop band (≥ 0.5Hz)	50 dB

Auxiliary measurements:

- Compulsory full-scale scalar magnetometer measurements with a data resolution of 0.01 nT at a minimum sample period of 30 seconds.
- Compulsory vector magnetometer temperature measurements with a resolution of 0.1 °C at a minimum sample period of one minute.

1.3 Applications

The ABZ magnetometer is not like the traditional scalar- or vector magnetometers, where applications are well defined from decades of research. The ABZ magnetometer provides us with a new combination of absolute geomagnetic information (F_E, H_E and Z_E) and with this combination comes new applications.

There is no doubt that a complete description of the geomagnetic field would be favourable. However, for some applications F_E, H_E and Z_E are enough, and for some applications continuous absolute measurement are a more interesting feature than relative independent geomagnetic elements.

A series of articles by Hiroaki Toh and Yozo Hamano introduces the possibility of providing a seafloor geomagnetic observatory for long term deployments at remote open oceans (Toh and Hamano, 2015). Among other interesting application, the two seafloor geomagnetic observatory detected tsunami signals from the 2006 and 2007 Kuril earthquakes (Toh et al., 2011). It can be seen from the measurements shown in Toh et al. (2011)[Fig: 8], that the land measurement cannot measure the magnetic signal from earthquakes. Hence, if earthquake measurements were a desired magnetic application, it would have to be employed at the seafloor.

For a seafloor geomagnetic observatory it is required to have an automatic absolute measurement, since there is no way to access it while it is deployed. The ABZ magnetometer would be well suited for a purpose like this.

A less extreme example is geomagnetic observatories in Greenland. The geomagnetic nature of polar regions have been an area of interest for a long time, but weather conditions can be a limiting factor with calibrations once a week.

Another application for the ABZ magnetometer could be as extra information to the already existing observations. An application in this context is the proposed routine baseline improvement in Soloviev et al. (2018), here the total intensity measurements are used to update the baseline estimation but the method could be expanded and improved with the ABZ magnetometer. This baseline improvement concept is further explained in "Theory", and one illustrative implementation is shown in "Results".

1.4 Goal for Project

The ultimate goal for the ABZ magnetometer is to automatically measure absolute geomagnetic field values for the horizontal (H_E), vertical (Z_E) and total intensity (F_E) to the level of accuracy required by the INTERMAGNET standards. Applications wise, the ABZ magnetometer is intended to measure in remote places, which demands a robust and trustworthy construction and method.

The reasoning behind the construction of the ABZ magnetometer was formulated in the synthesis project report Bjerg (2017b), and will not be the focus of attention in this thesis. The objective for this thesis is rather to describe in detail how H_E, Z_E and F_E can be derived from the ABZ magnetometer in the an effective and accurate way.

If H_E, Z_E and F_E can be obtained, the description of the prototype ABZ magnetometer will be complete. From the fundamental construction (Bjerg, 2017b) all the way to automatically deriving H_E, Z_E and F_E. However, with a project such as the ABZ magnetometer, optimization is always possible and there are clear opportunities for further development, expansion and optimization. This thesis also aims to highlight these important future avenues.

2

Theory

This chapter deals with the theoretical background related to the ABZ magnetometer. It will be separated into five main sections:

- Geometry of the ABZ magnetometer: The main features of the construction are examined.
- A model to the ABZ magnetometer data: A forward model related to the geometry of the ABZ magnetometer is constructed.
- A forward problem scheme: An simple approach for estimation H_E and Z_E as a forward problem is examined.
- Inverse problem schemes: Two approaches for determining the models parameters by solving the forward problem as a non-linear inverse problem is examined.
- Improving observatory baselines: This involves a mathematical description of a baseline improvement scheme developed in Soloviev et al. (2018), and an extension for a possible application for the ABZ magnetometer.

2.1 Geometry of the ABZ magnetometer

The motivation for the fundamental construction of the ABZ magnetometer and testing of the setup is described in Bjerg (2017b), no changes have been made in the construction. However, a description and definition of the geometry is a necessity for later explanation.

The main features can be seen in Figure 2.1, where the different parts are illustrated. The three main construction parts that are crucial for this thesis, are the scalar magnetometer, the suspension, and the Lee-whiting coil system. The scalar magnetometer is restricted to be either a Potassium magnetometer or an Overhouser proton precession magnetometer (Overhauser ppm). Both of these scalar magnetometers are good options, and both can be used. In this project the Potassium magnetometer is chosen.
The Lee-whiting coil is homogeneous and robust and fulfils its purpose. The idea of the suspension is to be hanging exactly in the vertical direction, but this is not a possibility in reality. Therefore, a definition for the misalignment is required, as detailed in Figure 2.2.

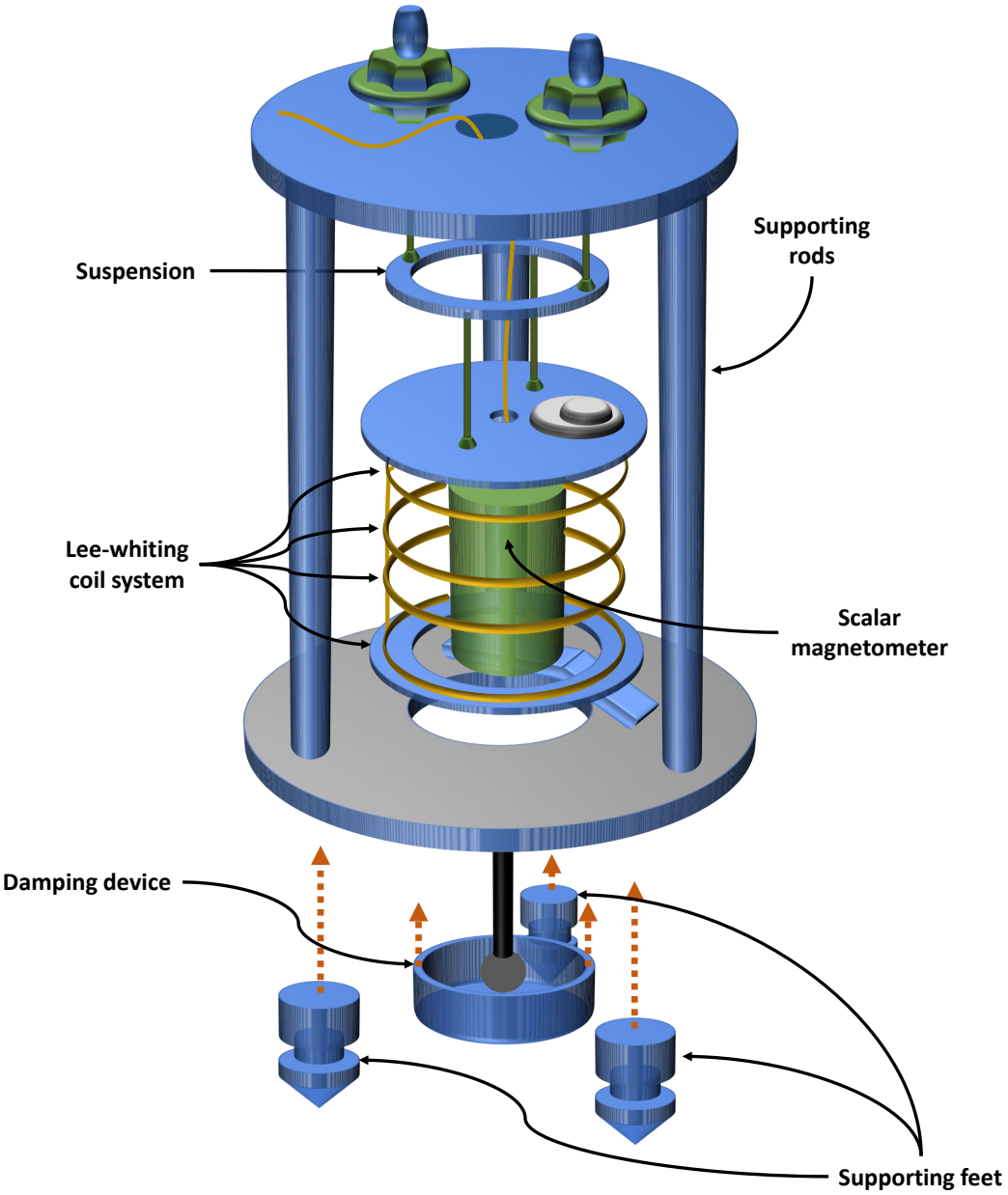


Figure 2.1: *Construction of the ABZ magnetometer (Bjerg, 2017b)*

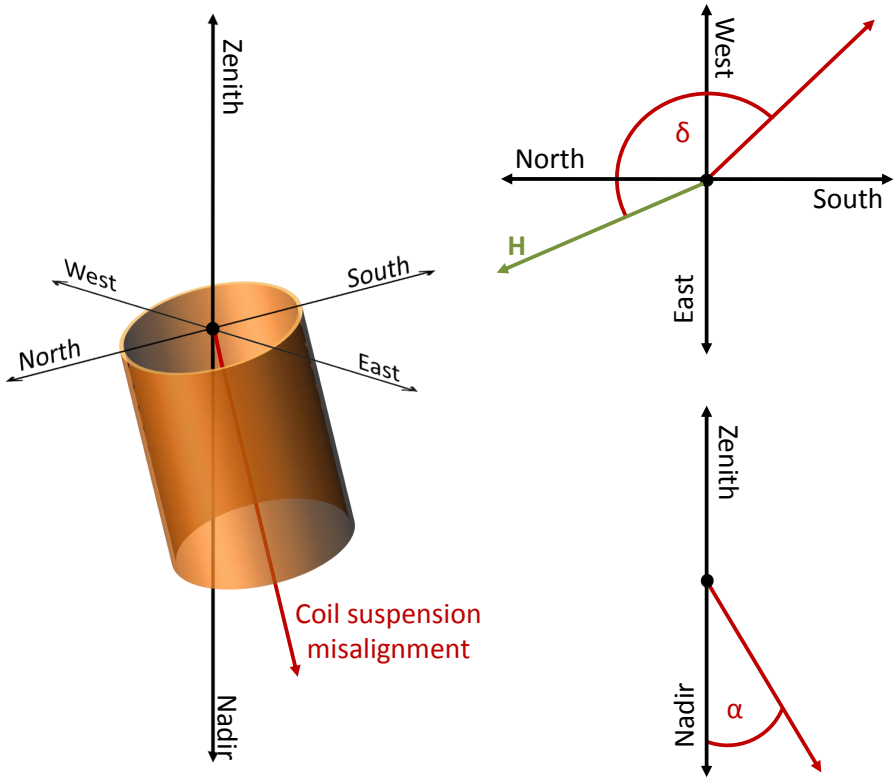


Figure 2.2: *Defination of coil misalignment to vertical (α) and horsontal (δ) direction.*

The definition of α and δ are rather important for understanding the mathematical description of the system. If δ is either $\frac{\pi}{2}$ or $\frac{3\pi}{2}$, the coil field will have no impact on the horizontal geomagnetic field, no matter the value α. However, the estimated vertical geomagnetic field will be unaffected by any variation of δ and therefore only depend on α.

Furthermore, note that δ is closely related to the declination of the geomagnetic field, and information on the declination can not be derived from δ.

2.2 A model to the ABZ magnetometer data

In this section a model describing the ABZ magnetometer data is presented. The model will be related to the geometry of the ABZ magnetometer and the nature of the geomagnetic field. It will not be possible to make a perfect model, but by making appropriate assumptions it can be very closely related to reality. The formulation of the model will aim to link information that can be measured with the model parameters that we wish to estimate.

The simplest form of a model is total intensity, and horizontal- and vertical intensity.

$$\mathbf{f}(H, Z) = \sqrt{H^2 + Z^2} \tag{2.1}$$

Where $f(H, Z)$ is the total intensity. H is the horizontal intensity and Z is the vertical intensity. To be useful in our case H and Z have to be expanded into the geomagnetic field (H_E and Z_E), and the field produced by the coil (S I $\sin\alpha\cos\delta$, SI $\cos\alpha$)

$$\mathbf{f}(H_E, Z_E, S, \alpha, \delta) = \sqrt{(H_E + SI\sin\alpha\cdot\cos\delta)^2 + (Z_E + SI\cos\alpha)^2} \tag{2.2}$$

where S is the coil sensitivity, α is the vertical misalignment and δ is the horizontal misalignment, as visualized in Figure 2.2. I is the coil current, the coil current is measured and is therefore not a model parameter in $\mathbf{f}(H_E, Z_E, S, \alpha, \delta)$.

If the geomagnetic field is not considered constant over the measurement internal, a linear slope in time can be added to the expression

$$\mathbf{f}(H_E, Z_E, S, \alpha, \delta, a, b) = \sqrt{(H_E + a\,n + SI\sin\alpha\cdot\cos\delta)^2 + (Z_E + b\,n + SI\cos\alpha)^2} \tag{2.3}$$

where a and b are slope parameters, and n is the measurement number that is related to the time. a and b will hereby be the linear change between each measurement. Equation 2.3 does not take any non-linear change into account, which will not be a good assumption for a fast change field or with big gap between measurement.

Both Equation 2.2 and Equation 2.3 can be applied as a model for describing the ABZ magnetometer. For fast measurements the slope variable may just over complicate the problem. Equation 2.2 is therefore used as the model for this thesis, nevertheless the extension with a linear slope may prove useful for other applications.

With the model in Equation 2.2, we have a non-linear problem. We can start examine techniques for estimating the model parameter (H_E, Z_E, S, α, δ). The data will be the total intensity measured by the Potassium magnetometer, and the measured coil current, i.e. our data is F and I.

2.3 A forward problem scheme

In section 2.2 a non-linear model describing our data has been located. In order to solve it as a forward problem is have to be reformulated. In the next section a forward problem scheme that is called the Equal steps method is described.

2.3.1 Equal steps (ES) method

The Equal steps method assumes no misalignment in suspension ($\alpha = 0$ and $\delta = 0$), and no linear change ($a = 0$ and $b = 0$). This will simplify the equation a great deal, as seen here.

$$F = \sqrt{(H_E)^2 + (Z_E + SI)^2} \tag{2.4}$$

Furthermore, it is assumed that the positive and negative current in a three level coil current are of equally opposite polarity. That will give rise to three equations, and three variables of interest. Where Z_{Coil} describes the vertical field produced from the coil.

$$\begin{aligned} F_0 &= \sqrt{H_E^2 + Z_E^2} \\ F_p &= \sqrt{H_E^2 + (Z_E + S\,I_p)^2} = \sqrt{H_E^2 + (Z_E + Z_{Coil})^2} \\ F_m &= \sqrt{H_E^2 + (Z_E + S\,I_m)^2} = \sqrt{H_E^2 + (Z_E - Z_{Coil})^2} \end{aligned} \tag{2.5}$$

Where F_0 is the measured field with no coil current, F_p has a positive coil current (I_P) and F_m has an equal and oppositely polarised coil current i.e. $I_p = -I_m$. These assumption enables a description of F_E, Z_E and H_E with the measured intensities F_0, F_p and F_m. F_E is directly measured when no current is applied in the coil system.

$$Z_{Coil} = \sqrt{\frac{F_p^2 + F_m^2}{2} - F_0^2}$$

$$\begin{aligned} Z_E &= \frac{F_p^2 - F_m^2}{4Z_{Coil}} \\ H_E &= \sqrt{F_0^2 - Z_E^2} \\ F_E &= F_0 \end{aligned} \tag{2.6}$$

The ES method therefore only need one value for F_0, F_p and F_m. The ES method can only be used for dataset using a three level coil current function.

In practise however the assumptions in the ES method can lead to large errors when deriving Z_E and H_E. Especially the assumption of no misalignment. In Figure 2.3 a theoretical estimation of the error due to change in α and δ is plotted. α is in the interval $[0 : 70]mRad$ and δ between $[-\frac{\pi}{2} : \frac{\pi}{2}]Rad$, which

is a plausible interval of misalignment. In practise, it will not be possible to level out the misalignment, to exactly zero, and an error is expected with ES method.

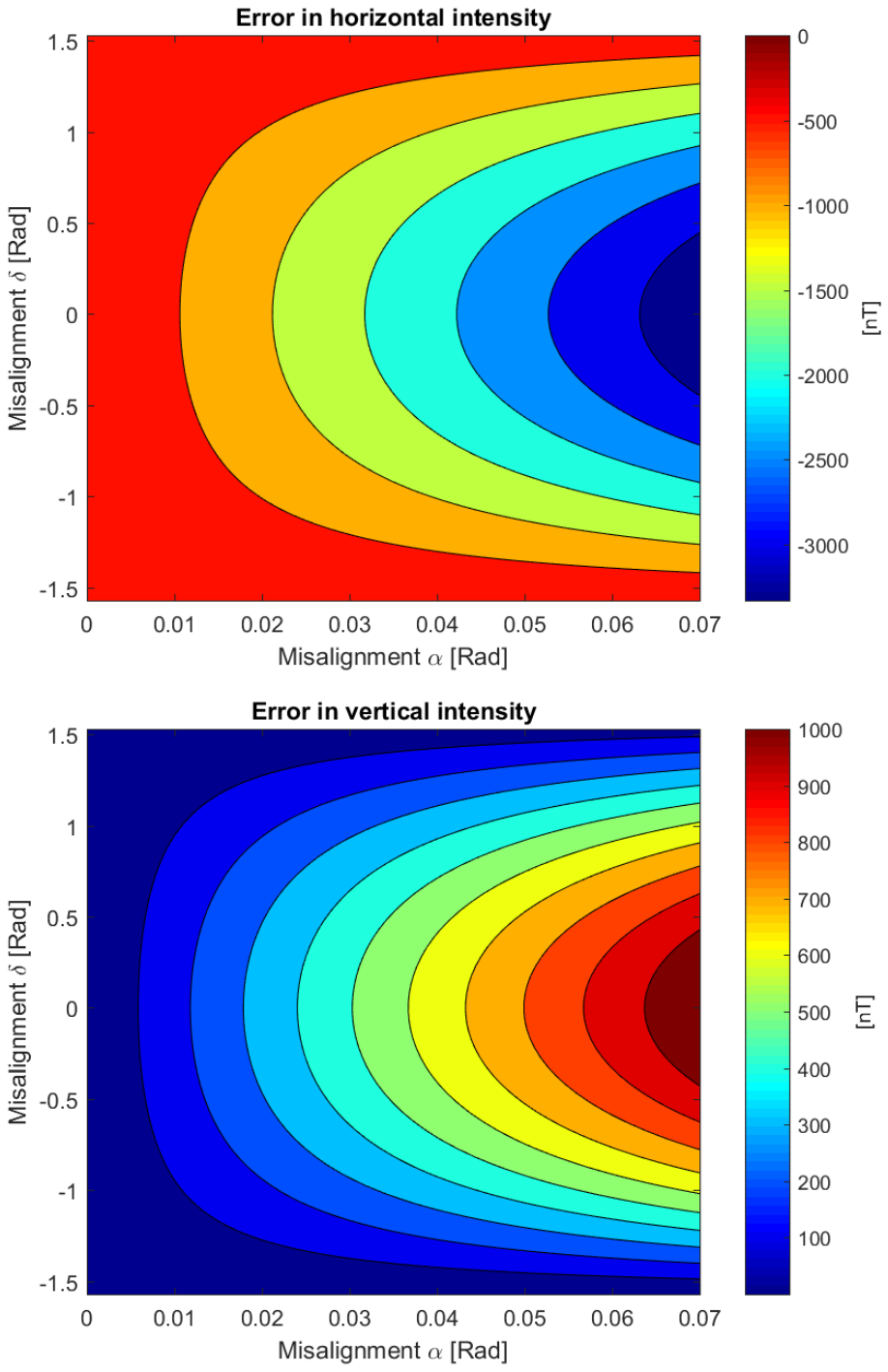


Figure 2.3: *Theoretical estimation of error due to change in α and δ using ES method. Where the true parameters are set to; $H = 17207nT$, $Z = 47000nT$, $S = 137\frac{nT}{mA}$ and $I = \pm 40mA$. α is in the interval $[0 : 70]mRad$, and δ is in the interval $[-\frac{\pi}{2} : \frac{\pi}{2}]Rad$. Calculation can be seen in appendix section D.3.*

2.4 Inverse problem schemes

The simplification in the forward scheme make the problem easy to solve, but the high errors make an interest for an alternative method for estimation of model parameters. By estimating the model parameters in Equation 2.2 as an over-constrained non-linear inverse problem, a various of inverse problem techniques become available without simplifying the model. In this section two approaches are investigated.

Firstly, an approach to solving the more general problem by numerical optimization is presented. secondly, an approach taking a Bayesian (Probabilistic) approach with a focus on deriving solutions using the Markov Chain Monto Carlo (MCMC) method is presented.

At this point, it is important to distinguish between model parameters and variables. Model parameters are all the parameters that are included in the forward model and can be freely varied or estimated from the data. While variables are for example time or position.

2.4.1 Numerical optimization

Numerical optimization is a broad expression, and is here further subdivided into Newton's method and Regularization. However, the goal is to minimize the residuals between the forward model predictions and the observations while obtaining a robust, stable, model parameter estimates.

A non-linear system will have a function of residuals $\mathbf{r}(\mathbf{m})$ where a minimum is found if $\mathbf{r}(\mathbf{m}_*) = \mathbf{0}$. Where $\mathbf{m}$ is defined as the vector of model parameters, and $\mathbf{m}_*$ is the vector of model parameters at a minimum.

For the specific case of Equation 2.2 the implementation can be seen in Equation 3.6.

A sufficient solution is obtained when the residuals are minimized. meaning that $\mathbf{r}(\mathbf{m})$ is equal or close to zero. The non-linear function of residuals can be expressed as an optimization problem.

$$\min_{\mathbf{m}\in\mathbb{R}^n} \mathbf{r}(\mathbf{m}) \tag{2.7}$$

When a residual minimum is found, the solution is said to be optimized. There is no general way to solve a non-linear problem, but iterative strategies have proven to be effective. It has to be mentioned that even through an optimized solution is found, it is not necessary the right solution. It is possible to find local residual minima in non-linear problems, which are not necessarily the global minimum.

Newton's method

The Newton's method is a classical way of solving a non-linear problem. The basic idea is to evaluate the residuals with a set of model parameters, and then locate a direction in model space where a more optimized solution to the model parameters exists.

The minimisation begins by suggesting a starting point $\mathbf{m}_0$. When assuming that a $\Delta\mathbf{m}$ exists such that following is true;

$$\mathbf{r}(\mathbf{m}_0 + \Delta\mathbf{m}) = \mathbf{0} \tag{2.8}$$

$\Delta\mathbf{m}$ represents the distance to a minimum and can therefore be symbolised as $\Delta\mathbf{m} = \mathbf{m}_* - \mathbf{m}_0$. $\Delta\mathbf{m}$ is henceforth called the **parameter step**. If the function $\mathbf{r}(\mathbf{m})$ is continuously differentiable, a Taylor series approximation can be constructed.

$$\mathbf{r}(\mathbf{m}_0 + \Delta\mathbf{m}) \approx \mathbf{r}(\mathbf{m}) + \mathbf{J}(\mathbf{m}_0) \cdot \Delta\mathbf{m} \tag{2.9}$$

Where $\mathbf{J}(\mathbf{m}_0)$ is the Jacobian matrix of the gradients of the residual function with respect to the model parameters. This gives the approximation.

$$\mathbf{0} \approx \mathbf{r}(\mathbf{m}_0) + \mathbf{J}(\mathbf{m}_0) \cdot \Delta\mathbf{m} \tag{2.10}$$

This Taylor series approximation is a linearisation of the non-linear function. An iterative estimation of $\Delta\mathbf{x}$ will improve the estimation of model parameters, and thereby iteratively minimise the data misfit. Because of the linearisation it can be treated as an iterative linear inverse problem. The most common measure of the misfit is the 2-norm, which is called the least squares (LSQ) solution. The 2-norm or LSQ is statistically the most likely solution if the data errors follows a Gaussian distribution. With a LSQ solution $\Delta\mathbf{x}$ can be isolated as Aster et al. (2011).

$$\Delta\mathbf{m} \approx (\mathbf{J}(\mathbf{m}_k)^T\mathbf{J}(\mathbf{m}_k))^{-1}\mathbf{J}(\mathbf{m}_k)^T\mathbf{r}(\mathbf{m}_k) \tag{2.11}$$

Where $\mathbf{m}_k$ are model parameters at iteration k. This approximation of $\Delta\mathbf{m}$ is used to derive new model parameters. i.e. the following takes place as an iterative process:

1. Calculate $\mathbf{J}(\mathbf{m}_k)$ and $\mathbf{r}(\mathbf{m}_k)$.
2. Solve Equation 2.11 with $\mathbf{m}_k$.
3. Define a new set of optimized model parameter $\mathbf{m}_{k+1} = \mathbf{m}_k + \Delta\mathbf{m}$.
4. Evaluate if $\mathbf{m}_{k+1}$ is a sufficient solution, otherwise, start over again.

This iterative solution to the non-linear problem is a simple way of solving a non-linear problem. But problems easily occurs for scenarios with ill-posed or ill-condition problems.

Regularization

There are a numbers of approaches to solving an ill-posed or ill-condition problem. Regularization is one commonly used approach. Within the scope of this thesis the Moore-Penrose pseudoinverse and the Levenberg-Marquardt method. In appendix section D.2 an alternative regularization with prior information is introduced.

The Moore-Penrose pseudoinverse is used to compute a generalized matrix inverse, and by use of singular value decomposition it can deal with rank-deficiency. Matlab has a function called **pinv** which creates the generalized inverse matrix.

The Levenberg-Marquardt is an extension of the Newton's method where a positive parameter λ is adjusted during the iteration (seen in Equation 2.12). The Levenberg-Marquardt method ensures non-singularity and therefore convergence.

$$\Delta\mathbf{m} \approx (\mathbf{J}(\mathbf{m})^T\mathbf{J}(\mathbf{m}) + \lambda I)^{-1}\mathbf{J}(\mathbf{m})^T\mathbf{r}(\mathbf{m}) \tag{2.12}$$

Where I is a identity matrix. For large λ the $\mathbf{J}(\mathbf{m}_0)^T\mathbf{J}(\mathbf{m}_0)$ will be negligible and the method will simulate a steepest decent method. The steepest decent only depends on the slope at the current iteration (Wright and Nocedal, 1999, p. 21-27). With small λ Equation 2.12 is similar to Equation 2.11.

2.4.2 Bayesian approach

The Bayesian approach is named after Thomas Bayes (1702 - 1761) who formulated "Bayes' theorem". When working with a Bayesian approach we distance ourself from the classical assumption that true values for the model parameters can be found, and instead we seek a probability distribution to describe our knowledge of the model parameters. It is therefore said that the model parameters in the Bayesian approach are random variables described by a posterior probability distribution of the model parameters. In the same way the solution is a probability distribution that is called the posterior probability distribution. It may seem a bit confusing to look for a solution that is not an exact solution, but by doing so we can estimate how likely our solution is. In cases where an single estimate of the model parameters is desired, the value with the largest posterior probability is chosen, also called the **Maximum A Posterior** (**MAP**).

When using a Bayesian approach to solve inverse problems, the method called the Bayesian inference, Bayes' theorem is applied to update the knowledge concerning the model.

$$q(\mathbf{m}|\mathbf{d}) = \frac{f(\mathbf{d}|\mathbf{m})\,p(\mathbf{m})}{c} \tag{2.13}$$

Where the posterior probability distribution for the model is denoted $q(\mathbf{m}|\mathbf{d})$. The prior probability distribution for the model is denoted $p(\mathbf{m})$. $f(\mathbf{d}|\mathbf{m})$ is the

conditional probability distribution for the model parameters $\mathbf{m}$ conditioned on the measured data $\mathbf{d}$, i.e. the likelihood for the model $\mathbf{m}$ being true given the corresponding data $\mathbf{d}$. c is sometimes called the **marginal likelihood**. The "c" notation is taken from (Aster et al., 2011, p.256), it is defined as

$$c = \int_{All\ models} f(\mathbf{d}|\mathbf{m})\, p(\mathbf{m})\, d\,\mathbf{m} \tag{2.14}$$

the integral in c is the main difficulty of Equation 2.13, if a general solution should be implemented. Insted of calculating c, two models, can alternatively be compared as a ratio:

$$\frac{q(\mathbf{m_1}|\mathbf{d})}{q(\mathbf{m_2}|\mathbf{d})} = \frac{f(\mathbf{d}|\mathbf{m_1})\, p(\mathbf{m_1})}{f(\mathbf{d}|\mathbf{m_2})\, p(\mathbf{m_2})} \tag{2.15}$$

If the result of this likelihood ratio is a small number, it indicates that $\mathbf{m_2}$ is a more likely solution than $\mathbf{m_1}$. This grants the opportunity for finding a more likely solution without computing c.

The Markov Chain Monte Carlo (MCMC) method

The Markov chain is a numerical sampling scheme, that works with a random choice of model parameters

$$\mathbf{m_0}, \mathbf{m_1}, \mathbf{m_2}, ..., \mathbf{m_n}, ... \tag{2.16}$$

For the sequence of random model parameters the current model parameters only depend on the previous. Each model has a probability distribution that again only depends on the previous distribution.

$$p(\mathbf{m_{n+1}}|\mathbf{m_0}, \mathbf{m_1}, \mathbf{m_2}, ..., \mathbf{m_n}) = p(\mathbf{m_{n+1}}|\mathbf{m_n}) \tag{2.17}$$

The random choice of distribution is used in a factor called the **acceptance ratio** (Aster et al., 2011, p. 271)

$$\boldsymbol{\alpha}(\mathbf{m_1}, \mathbf{m_2}) = min\left[1, \frac{q(\mathbf{m_1}|\mathbf{d})\, r(\mathbf{m}_1, \mathbf{m}_2)}{q(\mathbf{m_2}|\mathbf{d})\, r(\mathbf{m}_2, \mathbf{m}_1)}\right] \tag{2.18}$$

Which by definition of Equation 2.15 is

$$\boldsymbol{\alpha}(\mathbf{m_1}, \mathbf{m_2}) = min\left[1, \frac{f(\mathbf{d}|\mathbf{m_1})\, p(\mathbf{m_1})\, r(\mathbf{m}_1, \mathbf{m}_2)}{f(\mathbf{d}|\mathbf{m_2})\, p(\mathbf{m_2})\, r(\mathbf{m}_2, \mathbf{m}_1)}\right] \tag{2.19}$$

An acceptance ratio of 1 indicate 100% higher likelihood for $\mathbf{m_1}$ that $\mathbf{m_2}$ and is of cause accepted. But if the acceptance ratio is under 1 it indicate how likely it is to be accepted as a new model. This last part is of high importance, since this may prevent us from getting stuck in a local minimum.

2.5 Improving observatory baselines

Improvement of the absolute baseline for observatory time series is one possible application of the ABZ magnetometer. Here we describe the theory of how such an improvement could be achieved, a test of the scheme are presented later in section 4.4.

An article written in 2018 proposes a new approach to calculate a regular baseline for geomagnetic observatory time series (Soloviev et al., 2018). This proposal includes the total intensity F_E as the extra information to baseline estimation, which enables a routinely improved baseline for a relative vector magnetometer. It is tested on both synthetic examples and real data from the INTERMAGNET observatory in Saint Petersburg. An improved data quality compared to the classical approach was achieved.

The ABZ magnetometer directly measures the total intensity, and can be obtained with same accuracy as the measurement used in the article. The same approach could therefore be applied with data from the ABZ magnetometer. Furthermore, the ABZ magnetometer also provides information of the H_E and Z_E, and would therefore further improve the baseline for an relative vector magnetometer.

But before going further, lets walk through the proposed new approach from Soloviev et al. (2018). A slightly different notation will be used here, compared to the article.

Firstly, the known data in the proposal is

- One minute data of the total magnetic field F_a. Which consists of $F_a = F_{scalar} + F_{displacement}$, where F_{scalar} is the absolute magnetometer and $F_{displacement}$ is the difference of total intensity in site of installation between scalar magnetometer and absolute magnetometer. $F_{displacement}$ is a constant value.
- One minute relative magnetic vector field data X_v, Y_v and Z_v.
- One weekly measured baseline value $B = [X_b; Y_b; Z_b]$. Performed by a trained specialist using a theodolite.

The aim is to find the routinely improved baseline between the weekly measured baseline. Even though the baseline does not change significantly within a week, the idea is to reduce misleading fluctuations due to increased geomagnetic activity or weather conditions during an absolute baseline measurement.

The routinely improved baseline gets the symbol $S^{i,j,k} = [X_S^i, Y_S^j, Z_S^k]$, and has a estimated improved baseline value for every minute between the measured baselines. This notation does not differ between values, vectors or matrix, but it can almost directly be compared with the notation of the paper Soloviev et al. (2018). Note that B is a vector of three, and notated as B^p or B^n for

previous or next baseline, respectively. $S^{i,j,k}$ is a matrix of three times the minutes between B^p and B^n. F_a, X_v, Y_v and Z_v is all vector of at same length as $S^{i,j,k}$.
The following objective function $\phi(t, S^{i,j,k})$ consists of all the measured values and by optimising the function over $S^{i,j,k}$, a routinely improved baseline can be found.

$$\phi(t, S^{i,j,k}) = \lambda G + (1 - \lambda)A \tag{2.20}$$

Where λ is a number between $[0, 1]$, and does the weighting between G-matrix and A-matrix. That are defined as

$$G = \sum_{1h} \left[\sqrt{(X_v + X_S^t)^2 + (Y_v + Y_S^t)^2 + (Z_v + Z_S^t)^2} - F_a \right]^2 \tag{2.21}$$

$$A = [(B^p - S^{i,j,k})w_p]^2 + [(B^n - S^{i,j,k})w_n]^2 \tag{2.22}$$

Where

$$w_p = 1 - \frac{\Delta t_p}{T}, \quad w_n = 1 - \frac{\Delta t_n}{T} \tag{2.23}$$

Δt_p is the time since previous baseline measurements to the current moment. Δt_n is the time from current moment to next baseline measurements. T is the total time and is therefore also $T = \Delta t_p + \Delta t_n$.

When implementing this for stationary stations in Greenland, the baseline measurements are measured in horizontal intensity, declination and vertical intensity. The routinely improved baseline therefore has the notation $S^{i,j,k} = [H_S^i; D_S^j; Z_S^k]$, the measured baseline is $B = [H_b; D_b; Z_b]$. The one minute relative magnetic vector field measurement is HN_v, HE_v and Z_v for horizontal intensity in north direction, horizontal intensity in east direction and vertical intensity, respectively. The G-matrix therefore changes form to;

$$G = \sum_{1h} \left[\sqrt{\left(\sqrt{(HN_v + H_S)^2 + HE_v^2} \ \cos(\ \tan^{-1}\left(\frac{HE_v}{HN_v + H_S}\right) + D_S)\right)^2 + \left(\sqrt{(HN_v + H_S)^2 + HE_v^2} \ \sin(\ \tan^{-1}\left(\frac{HE_v}{HN_v + H_S}\right) + D_S)\right)^2 + \overline{(Z_v + Z_S)^2} \ - \ F_a} \right]^2 \tag{2.24}$$

This G-matrix was created by Anna Naemi Willer. An improved baseline estimation with the G-matrix in Equation 2.24, is performed on a 20 hours dataset, seen in the section 4.4.

The above optimization does only contain the total intensity as extra information. If ABZ magnetometer data was available we would have an absolute measure for the Z_E, and it would not be needed in the optimization. The improved

baseline would therefore now only consist of X and Y, i.e. $S^{i,j}_{ABZ} = [X^i_S, Y^j_S]$. Furthermore the G-matrix would be simplified to

$$G_{ABZ} = \sum_{1h} \left[\sqrt{(X_v + X^t_S)^2 + (Y_v + Y^t_S)^2} - H_a \right]^2 \tag{2.25}$$

where $H_a = H_E + H_{displacement}$. $H_{displacement}$ is the difference of horizontal intensity between the ABZ magnetometer location and the absolute magnetometer. At the moment there is not any ABZ magnetometer data available for testing the improving observatory baselines of $S^{i,j}_{ABZ}$. But as seen in Equation 2.25 its an easy implementation, if the proposed approach of Soloviev et al. (2018) is already functional.

3

Method

A big collection of experiments is background for the written material in the this chapter, and it may not show the whole picture of the workload. It is attempted to highlight which pitfalls there will occur, so the method can be repeated without falling into the same problems.

It has to be emphasized that the overall method-idea is to separate the model parameter estimation into two steps. Step 1; estimate S, α and δ as prior information. Step 2; use the estimation from step 1 to make a continuous estimation of H_E, Z_E and F_E. This is primitively illustrated in Figure 3.1, where each steps both have a survey and data processing associated.

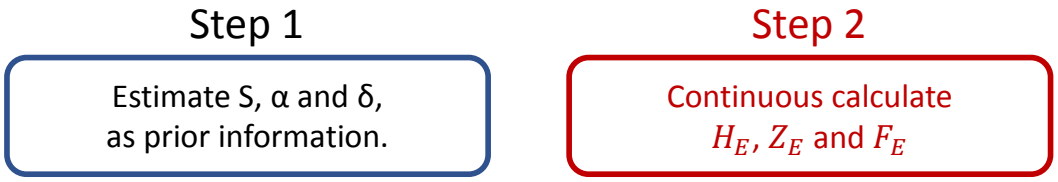


Figure 3.1: *The overall method-idea is to seperate the estimation into two steps, the blue has been achieved and the red has not been achieved in this thesis. Each step is further explaned in section 3.1*

These two steps are separately described in the following section 3.1, where it is also shown how to implement it.

Related to the method multiple surveys are conducted. In order to collect the data at a survey, a computer system was developed. The purpose of the computer system is to control and measure the coil current with high precision. Furthermore, the computer system synchronise the Potassium magnetometer data with coil current, and provide it with a time-stamp. The computer system is evaluated in the following section 3.2. A good computer system provides good data, and this is therefore important.

The following descriptions is targeting the ABZ magnetometer. However, a similar set-up with an Overhauser ppm inside a big Helmholtz coil can be used (See Appendix E). This alternative set-up can be connected to the same computer system, but uses a slightly different method for model parameter estimation.

3.1 Estimation of model parameters

In the theory chapter determination of model parameters was examine, and the three methods were: Equal step method, Numerical optimization and Bayesian approach. With its basis in Equation 2.2 the residual for all model parameters can be written as

$$\begin{aligned}\mathbf{r}(\mathbf{m}) &= \mathbf{F} - \mathbf{f}(\mathbf{m}) \\ &= \mathbf{F} - \mathbf{f}(H_E, Z_E, S, \alpha, \delta) \\ &= \mathbf{F} - \sqrt{(H_E + S\mathbf{I}\sin\alpha \cdot \cos\delta)^2 + (Z_E + S\mathbf{I}\cos\alpha)^2}\end{aligned} \tag{3.1}$$

Where the current (I) and total intensity (F) are the measurements, and will be written as a vector $\mathbf{I}$ and $\mathbf{F}$. H_E, Z_E, S, α and δ are the model parameters, defined by the vector $\mathbf{m}$.

A prior constrain for the misalignment in α is $\alpha < 62mRad$ which is a physical boundary in the construction of the ABZ magnetometer. Furthermore a number of generous lower and upper boundaries were assumed in this optimization problem, see Table 3.1.

Table 3.1: *Lower and upper boundaries for the optimization problem*

	H_c	Z_c	S	α	δ
Lower bound	0	0	0	$-62 \cdot 1e-3$	$-\pi$
Upper bound	70000	70000	300	$62 \cdot 1e-3$	π

Without further ado, this could be applied on ABZ magnetometer measurement. By doing so, a high correlation between S and α are identified, this is shown in section 4.1.

Because of the high correlation a new approach is therefore introduced, where prior distributions of S, α and δ are estimated before H_E, Z_E and F_E are estimated. This also means that we do not include an estimation of slopes (a and b) as described in Equation 2.3. These two slopes are removed completely from the equation.

This introduces a new aspect, determination of distributions of S, α and δ. This is the an initial step before the rest of the the method can continue.

3.1.1 Step 1: Prior estimation of S, α and δ

It is decided to determine S, α and δ as prior information, this is done under controlled circumstances in an observatories. Vector magnetometer measurements can therefore be applied to the estimation as extra information. For this prior estimation, both surveys and data processing are needed, the approach used for the ABZ magnetometer can be seen in Figure 3.2.

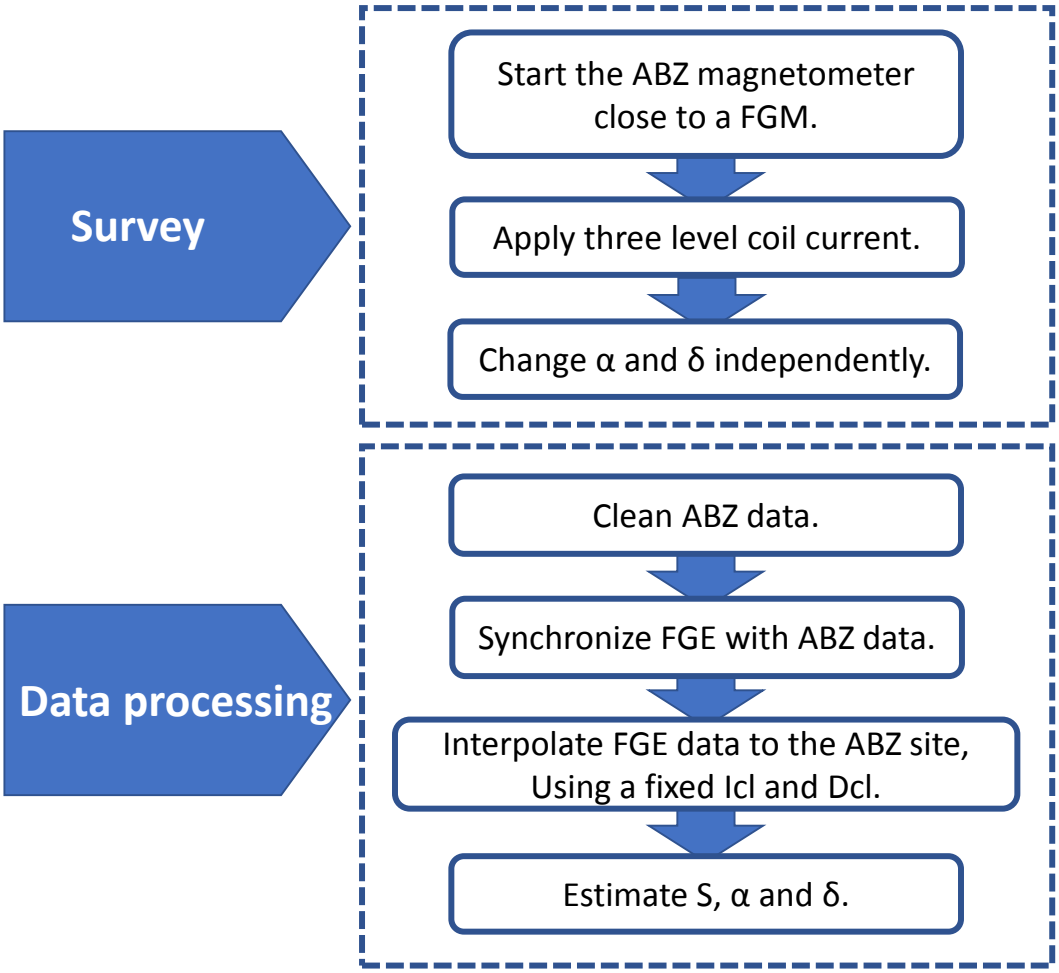


Figure 3.2: *Survey and data processing for step 1*

Survey

For this survey the ABZ magnetometer has to be located close to a vector magnetometer, such as the FGM in Brorfelde. Depending on where the baseline is measured there will occur a difference in the total intensity. This difference can be solved in the data processing by assuming a fixed magnetic inclination and declination. Alternatively the baseline measurements could be conducted at the same site as the ABZ magnetometer measurements. This would remove any error due to magnetic properties like small changes in the magnetic crustal field.

Ideally a ramp coil current could just as well be applied, but the three level coil current is good enough.

An additional independent change of α and δ were made during the survey, by changing them independently it ensured uncorrelated model parameter.

Data processing

Data from the FGM is therefore now available for the data processing, consisting of X_{FGM}, Y_{FGM} and Z_{FGM}, as the geomagnetic component for north, east and nadir, respectively (See Figure 1.1). However, before it can be used the constant offset in the total intensity between FGM and ABZ, has to be accounted for. The value is believed to origin from displacement in sites of measurements. An additional value was therefore applied to each of the components, it was calculated by assuming a constant inclination and declination, i.e.

$$
\begin{aligned}
H_{displacement} &= F_{displacement}\frac{\sin(Icl)}{\tan(Icl)} \\
X_{at\,pole} &= X_{FGM} + H_{displacement}\cos(Dcl) \\
Y_{at\,pole} &= Y_{FGM} + H_{displacement}\sin(Dcl) \\
Z_{at\,pole} &= Z_{FGM} + F_{displacement}\sin(Icl)
\end{aligned}
\tag{3.2}
$$

$F_{displacement}$ is the measured intensity between FGM and ABZ, Icl is the inclination, and Dcl is the declination. Icl and Dcl is measured values for the area. $X_{at\,pole}$, $Y_{at\,pole}$ and $Z_{at\,pole}$ are geomagnetic values at the site of ABZ measurements.
Now this can be used to compose a model for the data,

$$
\begin{aligned}
\mathbf{f}(S,\alpha,\delta,I_{offset}) = &\sqrt{(X_{at\,pole} + S(I+I_{Offset})\sin\alpha\cdot\cos\delta)^2 + } \\
&\overline{(Y_{at\,pole} + S(I+I_{offset})\sin\alpha\cdot\sin\delta)^2 + } \\
&\overline{(Z_{at\,pole} + S(I+I_{offset})\cos\alpha)^2}
\end{aligned}
\tag{3.3}
$$

I_{Offset} is a model parameter, which only is added to positive currents. Now all data from both the FGM and ABZ magnetometer can be employed, which leave S, α, δ and I_{Offset} to be estimated. The theory for determination of model parameters can now be applied for Equation 3.3, but with four model parameters to be estimated.

$$
\begin{aligned}
\mathbf{r}(\mathbf{m}) &= \mathbf{r}(S,\alpha,\delta,I_{Offset}) \\
&= F_{ABZ} - \sqrt{(X_{at\,pole} + S(I+I_{offset})\sin\alpha\cdot\cos\delta)^2 + } \\
&\quad \overline{(Y_{at\,pole} + S(I+I_{offset})\sin\alpha\cdot\sin\delta)^2 + } \\
&\quad \overline{(Z_{at\,pole} + S(I+I_{offset})\cos\alpha)^2}
\end{aligned}
\tag{3.4}
$$

For Equation 3.4 the Bayesian method is primarily used. It enable both a visualization and a distribution.

The above describe the ABZ magnetometer. For the Overhauser ppm inside the Helmholtz coil, it is not possibility to change α or δ independently, since the Helmholtz coil is mounted to the floor (Figure E.1a). Furthermore, there was no need for and current offset as seen in Equation 3.3. I_{offset} is therefore set to zero for the set-up with Overhauser ppm and Helmholtz coil system.

3.1.2 Step 2: Estimation of H_E, Z_E and F_E

When estimating H_E, Z_E and F_E a new survey and data processing has to be made. This is illustrated in Figure 3.3, where red boxes are not tested for the ABZ magnetometer.

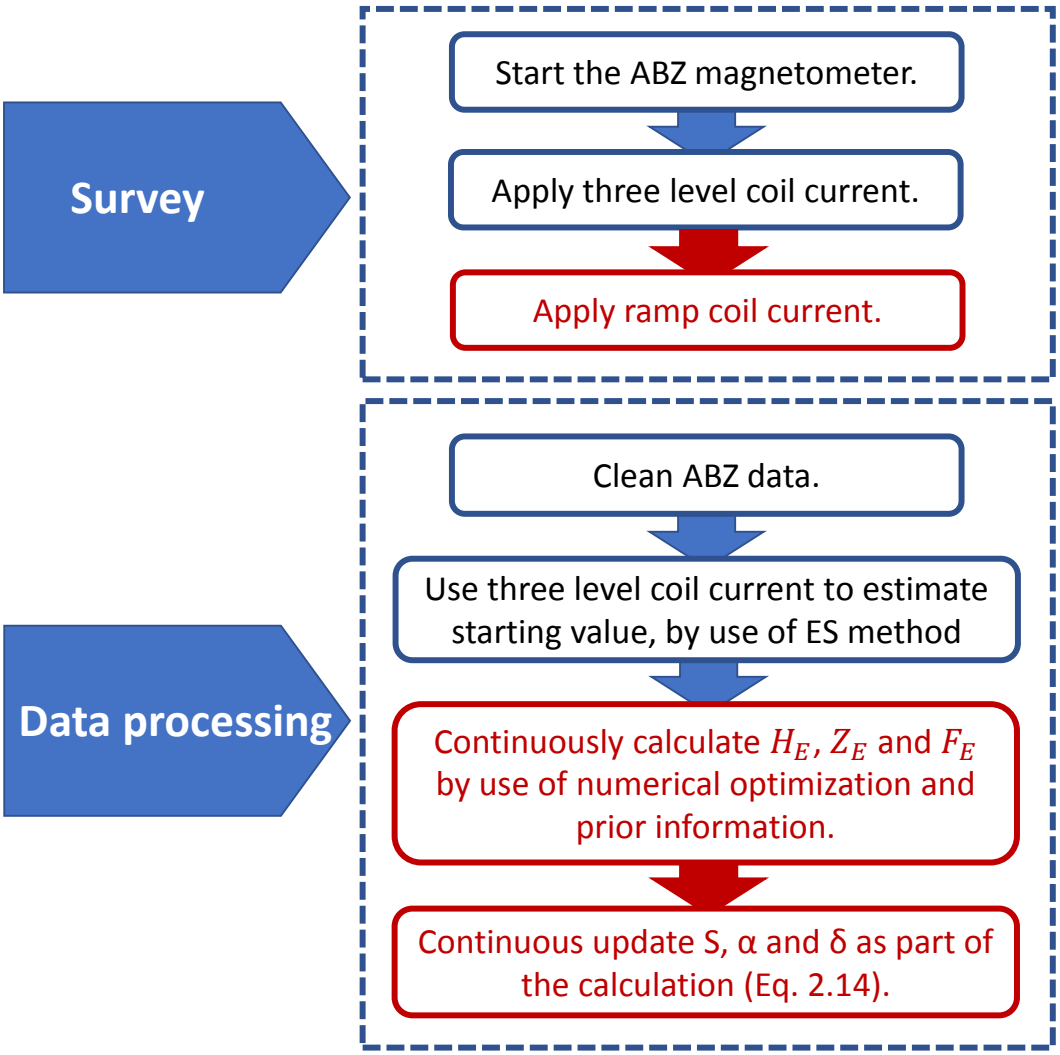


Figure 3.3: *Survey and data processing for step 2. The boxes in red was not implemented with the ABZ magnetometer.*

Survey

During this survey it is not necessary having a FGM close by for calibration. However, when testing the results it can be an advantage to calculate the residuals.

Firstly the three level coil current is applied, which enable a starting estimation of H_E, Z_E and F_E with ES method. Alternatively a guess of starting values could do the job.

Finally, a ramp coil current is applied. The ABZ magnetometer should ideally send the data to an external computer, so the model parameters can be estimation meanwhile.

Data processing

By now S, α and δ is estimated, the forward model is therefore back to the statement in Equation 2.2 but now only with two model parameters

$$\mathbf{f}(H_E, Z_E) = \sqrt{(H_E + SI\sin\alpha \cdot \cos\delta)^2 + (Z_E + SI\cos\alpha)^2} \tag{3.5}$$

Which will create a function of residual

$$\begin{aligned} \mathbf{r}(\mathbf{m}) &= \mathbf{F} - \mathbf{f}(\mathbf{m}) \\ &= \mathbf{F} - \mathbf{f}(H_E, Z_E) \\ &= \mathbf{F} - \sqrt{(H_E + S\mathbf{I}\sin\alpha \cdot \cos\delta)^2 + (Z_E + S\mathbf{I}\cos\alpha)^2} \end{aligned} \tag{3.6}$$

The estimation must mainly be done by numerical optimization, since a Bayesian approach would demand too much computation time even on a good computer.

3.1.3 Implementation of data processing

In the following the implementation for estimating the model parameters are illustrated. These methods are used in both the prior estimation and estimation of H_E, Z_E and F_E.

Equal step method

The Equal step method is a straight forward method, where an average of each step is taken. With one average value for each current step, Equation 2.6 can be used directly. However, this also means that at least three measurements are needed to calculate H_E and Z_E. This method is only applied when there is no prior knowledge for a starting point. This method can only be used for three level coil current.

Numerical optimization

Numerical optimization hastily becomes complex, so a simplification is introduced with the pre-installed Matlab function "`lsqcurvefit();`", it enables a number of reasonable approaches to solve a non-linear inverse problem. Here is how to use it in Matlab:

```
[m,resnorm,residual,exitflag,output,lambda,jacobian]
   = lsqcurvefit(f(x, I_n),x_0,I_n,F_n,lb,ub,options);
```

'$f(x, I_n)$' is the function for which x is the vector of model parameters, 'x_0' is the starting values for the model parameters, 'I_n' and 'F_n' are the current data and total intensity data respectively, 'lb' and 'ub' are the lower and upper bounds respectively. In practice it will be done similar to the the following Matlab code.

```
Window = 300;
for i = 1:(length(I_All)-Window)
    F_n = F_All(i:i+Window);
    I_n = I_All(i:i+Window);
    n = [1:length(I_n)]';

Fun = @(m, [n, I_n]) (sqrt( (m(1) + m(4).*n + ...
        I_t.*m(3).*sin(m(6)).*cos(m(7))).^2 + ...
        (m(2) + m(5).*n + I_t.*m(3).*cos(m(6))).^2    ));

[m(:,i),resnorm,residual,exitflag,output,lambda,jacobian] = ...
    lsqcurvefit(Fun,m0, I_n, F_n, l,u,options);

    m0 = m(:,i);
end
```

I_{All} is all the measurements available in a survey, within I_{All} a window is chosen. The window is the amount of measurements to optimize over. The loop will therefore make the optimization within the window. Save the calculated model parameters and move the window one measurement to do the optimization again. Because the model parameters are not believed to change much the model parameters from last optimization can be used as x_0 in the next.

The advantage of using the pre-installed Matlab function `lsqcurvefit();` is the flexibility within the function, for instance you can easily choose between 'jacobian', 'trust-region-reflective' or 'levenberg-marquardt'.

```
options = optimoptions('lsqcurvefit','Display','iter');
options.ScaleProblem = 'levenberg-marquardt';
```

The 'Display' option can be set to 'off' or 'iter'. For 'iter' Information on each iteration will be displayed in command window of Matlab. The disadvantage for the pre-installed Matlab function, is that it can not include a prior information as Equation D.1. Alternative a simple version of `lsqcurvefit();` is shown in the next box.

```
function [m] = Simple_Linearsation()

while relative_change > 0.0001

    [J] = Jacobion(I_n, m);
    r = F_n - F_field(I_n, m);

    delta_m = pinv(J'*J)*J'*r;
    m = m + delta_m;

    relative_change = 100*norm(delta_m)/norm(m);

end

end
```

Here Moore-Penrose is used as regularization. The extended Matlab code can be seen in appendix B.

Bayesian method

The implementation of the Bayesian method was implemented by C.Finlay 06.04.2017, for the DTU MSc class Inverse problems in Earth and Space Sciences. C.Finlay took basis in the example (Aster et al., 2011, p. 274-278). The example is similar except from the model. The code can be seen in appendix B.

To avoid very small numbers in the numerator the logarithm of the acceptance ratio can be a computational advantage.

$$log\left[\frac{f(\mathbf{d}|\mathbf{m_1})\,p(\mathbf{m_1})\,r(\mathbf{m}_1,\mathbf{m}_2)}{f(\mathbf{d}|\mathbf{m_2})\,p(\mathbf{m_2})\,r(\mathbf{m}_2,\mathbf{m}_1)}\right] =$$
$$log[f(\mathbf{d}|\mathbf{m_1})] + log[p(\mathbf{m_1})] + log[r(\mathbf{m}_1)] - log[f(\mathbf{d}|\mathbf{m_2})] - log[p(\mathbf{m_2})] - log[r(\mathbf{m}_2,\mathbf{m}_1)] \tag{3.7}$$

Additional to the code C.Finlay made, a color was added to the distribution. If the color is random it indicate a proper minimum is found.
It has been descussed have to implement the ES method, Bayesian and two approaches of regularizations of Newton's method. They are all tested in next result chapter.

3.2 Computer system

In this section the conceptual ideas for a computer system to the ABZ magnetometer is examine. The intention is to clarify which choices to take for composing a functional computer system. If it is desired to learn how the specific implementation was made in this project, it is shown in greater detail in Appendix A, where all the components and pictures of the computer system is shown. Furthermore, the an assembling description and Arduino code is included.

The essential purpose of the computer system is to collect data from the potassium magnetometer while controlling a well-measured coil current. All data need to be synchronise and send to a memory storing unit or printed serial to a computer terminal. The synchronization and proper timestamps is very important for a final functional computer system. This is also emphasized in general specification in Table 1.2, with a recommended time-stamp accuracy on 0.01 s.

The Arduino UNO were utilised as microcontroller. Arduino UNO only operate with one serial-channel, and to avoid communicating problems, change between communication units will be kept to a minimum. The constrained from only one serial-channel were the main limitation in the computer system development, and the description reflect this.

The GPS and memory storing is not part of the computer system implemented in appendix A, but they will still be described. It can therefore easily be implemented with an alternative microcontroller with multiple serial-channels.

3.2.1 Workflow

For a proper overview of the element involve in the computer system, the flowchart in Figure 3.4 has been design. The flowchart is closed related to the code for the computer system.

The flowchart start at the "Start" box, this is when all are turn on and ready. The Potassium magnetometer will start producing data, this is represented in the "Potassium data" box. When the Potassium magnetometer send a newline, this indicate an accomplish measurement from the Potassium magnetometer. All the received data is synchronized with a GPS time and a measurement of the coil voltage bit-value. The synchronized data is in fact the end-product and is directly send via a USB connection to a computer terminal. Alternative it is stored on a memory storing unit.

The total intensity is showed on a LCD panel, this can be useful when calibrating the misalignment of the coil suspension.
When synchronization of data is complete, it ask the question; "is cycles smaller that K" ($Cycles < K$). K is a counter of Potassium magnetometer measurements, and then the desired data cycles are execute, is will change the

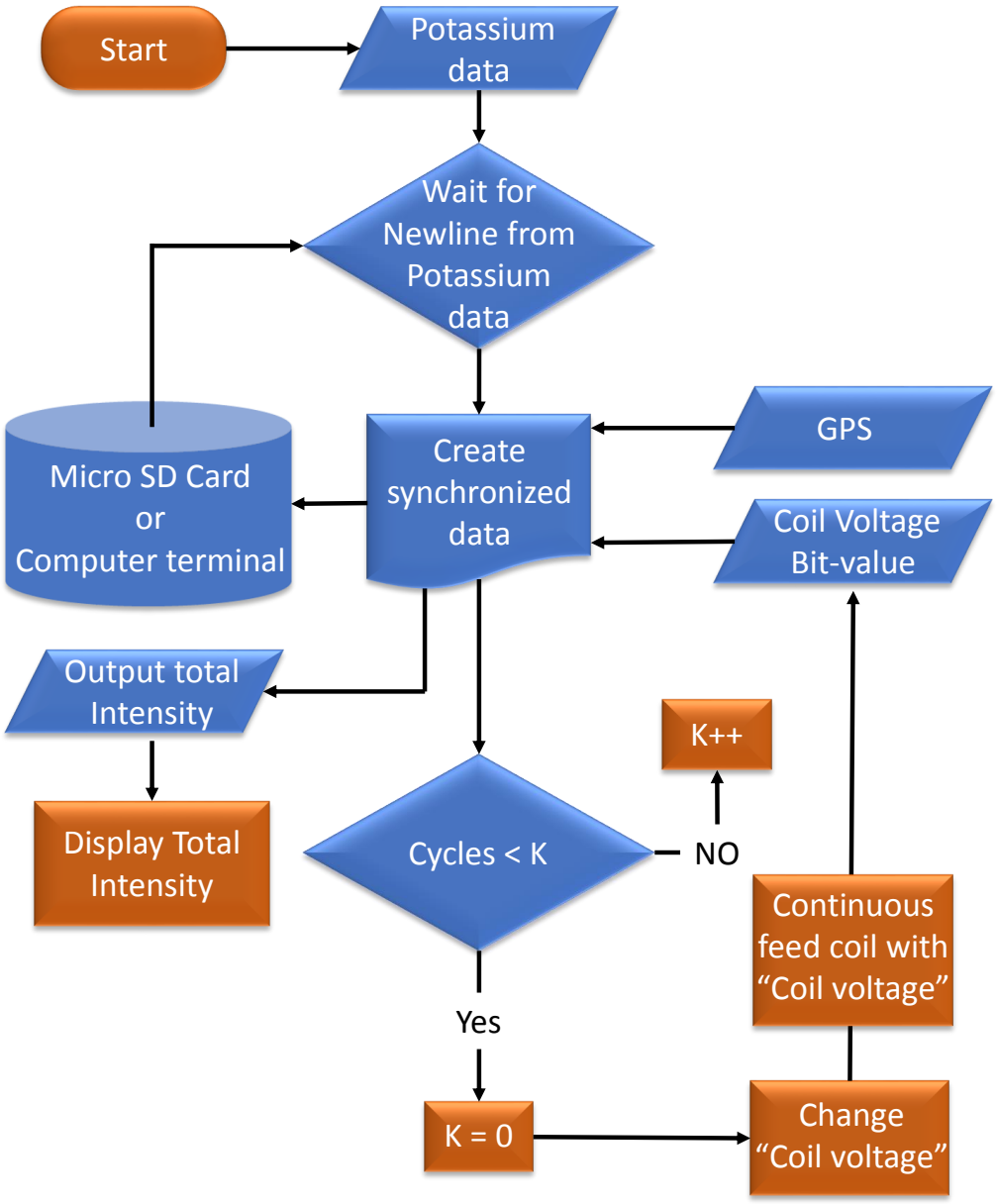


Figure 3.4: *Flowchart of the computer system, the used symbols are the common notation for flowcharts (Lucidchart.com).*

coil voltage. As an example we could simulate a 10 Hz Potassium magnetometer sampling rate and a cycle on 20. Thereby the voltage would change every $\frac{1}{10}\text{s} \cdot 20\text{cycle} = 2\text{s}$. The coil voltage would change due to a predefined sequence depending on the current-function. The current-function refer to either three level current or ramp current as seen in Figure 1.2. When the coil voltage is change the next synchronised dataset will include these new properties.

When only one serial-channel are available, all computations has to be done before a new newline, which also means that computation time have to be consistent with the time available between Potassium magnetometer data.

3.2.2 Electronic design

The main reason for choosing an Arduino microcontroller unit is its easiness and readability. We have to be able to easy implement new features and change the code if new problems or ideas occur, and this is where the Arduino universe is the prize winner. There is a ton of libraries and functions to Arduinos Integrated Development Environment (IDE), and together with the large community sharing knowledge, an Arduino microcontroller unit is perfect for this project.

In Figure 3.5 is how the electronic was implemented in this thesis, where a LCD panel show the current total intensity measured from the Potassium magnetometer. In the process of describing the electronic design, it have been divided into to design-units.

- **Ana-Unit** is the controlling unit, Ana-Unit synchronise all the data and change current via a DAC, it measure the coil current with a ADC. The electronic design of Ana-Unit is illustrated in Figure 3.6

- **Pot-Unit** read the Potassium magnetometer data through a USB Host, the Potassium magnetometer data is send to Ana-microcontroller and the total intensity is display on the LCD panel. The electronic design of Pot-Unit is illustrated in Figure 3.7

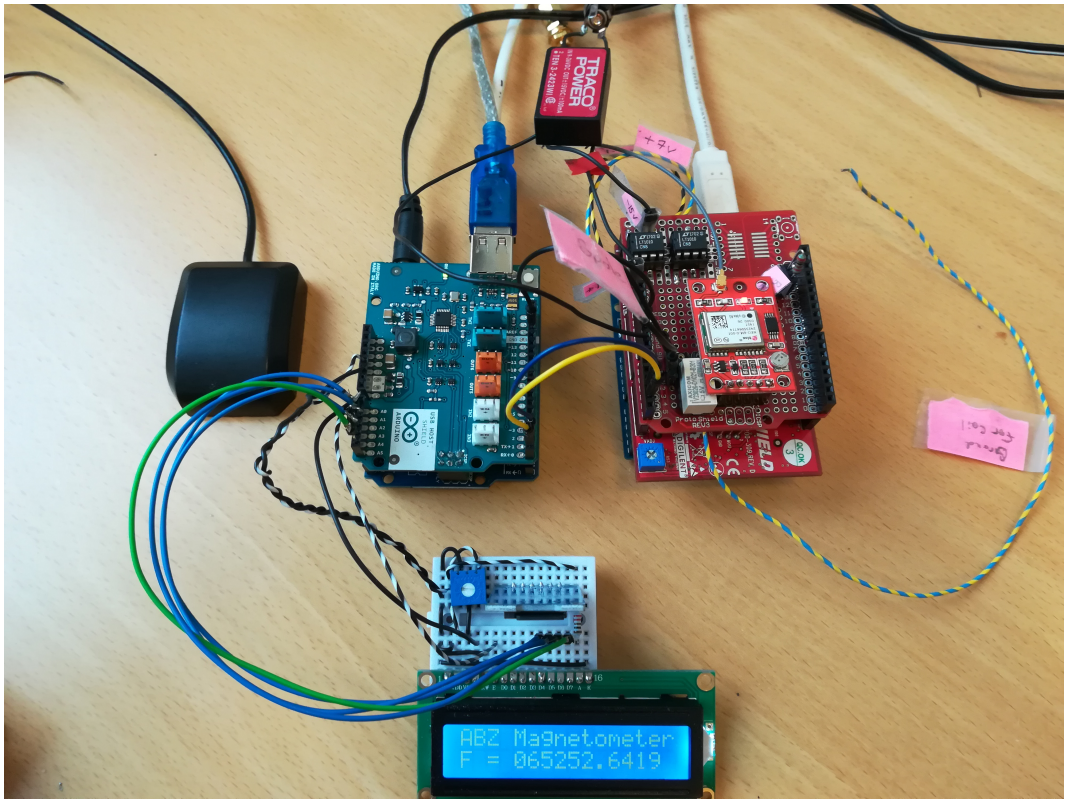


Figure 3.5: *The implementation of the computer system, see appendix A for component information*

The DAC and ADC is connected to the Ana-microcontroller using Serial Peripheral Interface (SPI) (www.byteparadigm.com). The DAC create a coil current corresponding to the chosen current-function. When generating the coil current, the DAC provide a good measure of the current, but due to noise it includes a high uncertainty. The coil current is therefore also measured over a reference resistor with a ADC. The ADC will measure all current in the coil no matter origin, which can provide a more precise measure of the coil field, i.e. a smaller uncertainty. A minimum of 16-bit ADC and DAC with $\pm 5V$ range, is a desired minimum threshold. This create a resolution on

$$\frac{\pm 5V}{16 - Bit} = \frac{10V}{2^{16} Bit} = 0.15259 \frac{mV}{Bit} \tag{3.8}$$

Meaning, that the ADC/DAC have a resolution on $0.15259mV$ when converting analog to digital or the reverse. A coil field range of $\pm 5000\,\mathrm{nT}$, would demand a coil current on approximately $\pm 40\,\mathrm{mA}$. This is estimated with a theoretical coil sensitivity of $137\,\frac{\mathrm{nT}}{\mathrm{mA}}$ (Bjerg, 2017b, p. 16). From Ohm law a reference resistor should accordingly be

$$\frac{U}{I} = R = \frac{5V}{40mA} = 125\Omega \tag{3.9}$$

If this is implemented it would give $\frac{5000nT}{5V} = 1000\frac{nT}{V}$, thus:

$$1000\frac{nT}{V} \cdot 0.15259\frac{mV}{Bit} = 0.15259\frac{nT}{Bit} \tag{3.10}$$

Meaning, a ADC/DAC with above specification, will have a coil field resolution of $0.152\,59\,\mathrm{nT}$. This is a theoretical value, and in reality the resolution would be measured for the computer system.
Higher coil field resolution could be achieved by a higher voltage range, higher reference resistor or a ADC higher that 16-bit. The next constrain for the ADC/DAC is the sampling rate, the implemented system has a sampling rate on $100\,\frac{\mathrm{kS}}{\mathrm{s}}$. Meaning, it can make 10.000 measurement per. second, this is a excellent sampling rate and less may be sufficient. The magnetic field resolution could be improved if the sampling rate of the ADC was used to its full extent of $100\,\frac{\mathrm{kS}}{\mathrm{s}}$.

A relay was used for F_0, to ensure no leakage of coil current. H-bridge is a good alternative to a relay, but a H-bridge also have a small current leakage and it is therefore not recommended. The only threshold for the DC-DC converter, is its range of voltage, i.e. at least ± 40 mA.

On the left side of Figure 3.6, a GPS and memory storing is connected. The data storing is not a critical part, and especially not in the development phase, if the data is send to a computer terminal. The GPS is very important, and is essential for a ABZ magnetometer. If the time is taking only from the GPS it can achieve max $10\,\mathrm{Hz}$ (gpscompared.net), this demand optimal condition and a well written code. The more simple way of achieving time accuacy from the GPS, is by means of pulse per second (pps). An internal clock is running and is continuum calibrated by the ppm from the GPS.

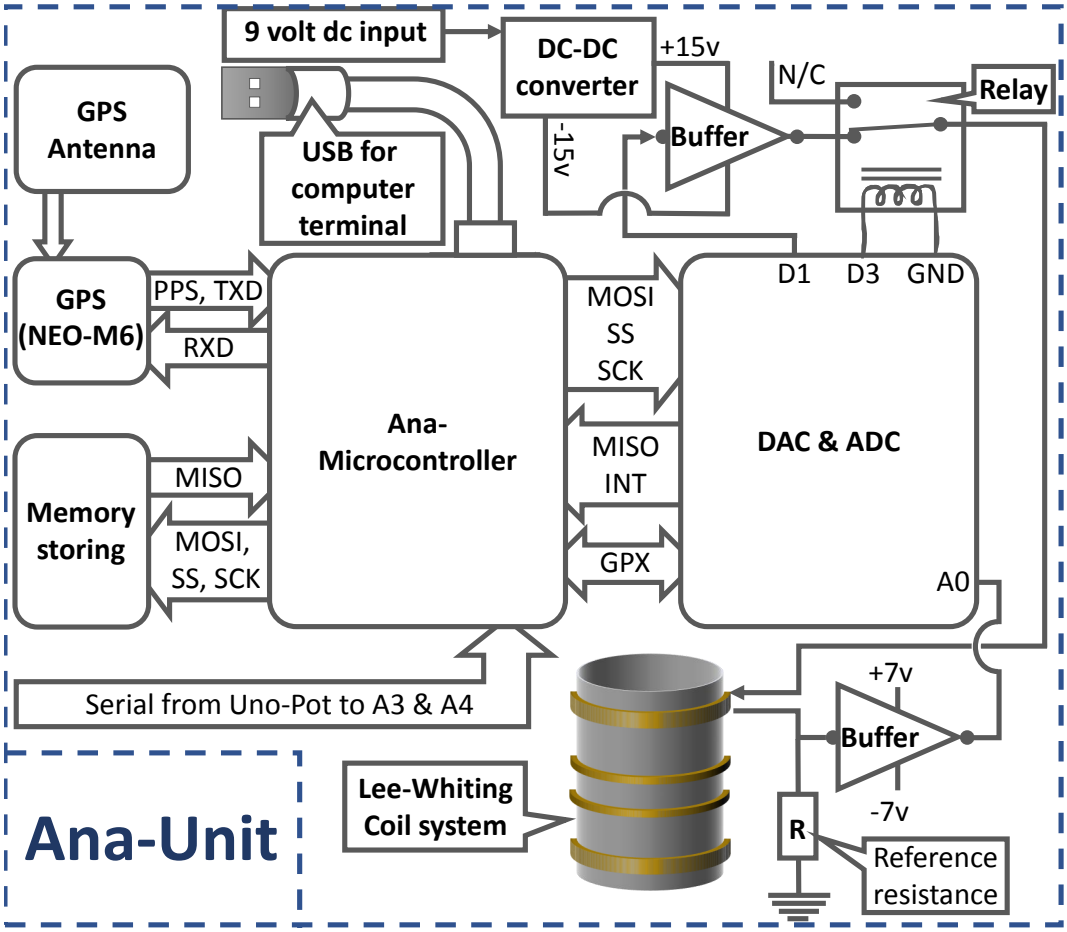


Figure 3.6: *Electronic design of Ana-Unit*

The Potassium magnetometer deliver its measurements serial through a USB, and it can be receive by a USB Host. This is seen on Figure 3.7, where the Pot microcontoller controls the USB Host by SPI. It would seem appealing to mount the USB Host as a top shield on the Ana-Unit, this was not a possibility for the specific components used in this project. If it is desired to mount it on the Ana-unit, make sure that the SPI can communicate. Note especially whether the amplitude and the sampling edge is compatible. The sampling edge can normally be modified in the libraries for the devices, but the consequence can be unclear.

To avoid any communications problems with the SPI, it was decided to isolated the reading of Potassium magnetometer. There are many god solution to reading the serial signal from the Potassium magnetometer. Allocating a microprocessor for the Potassium data has the advantage, that it never missing a measurements, and the reading can be made very robust since it does not take other things into consideration.

The LCD panel with 16 characters times two lines is about the minimum character for this purpose. If more information than just the total intensity is desired to be shown, the $16 \cdot 2$ characters is not enough.

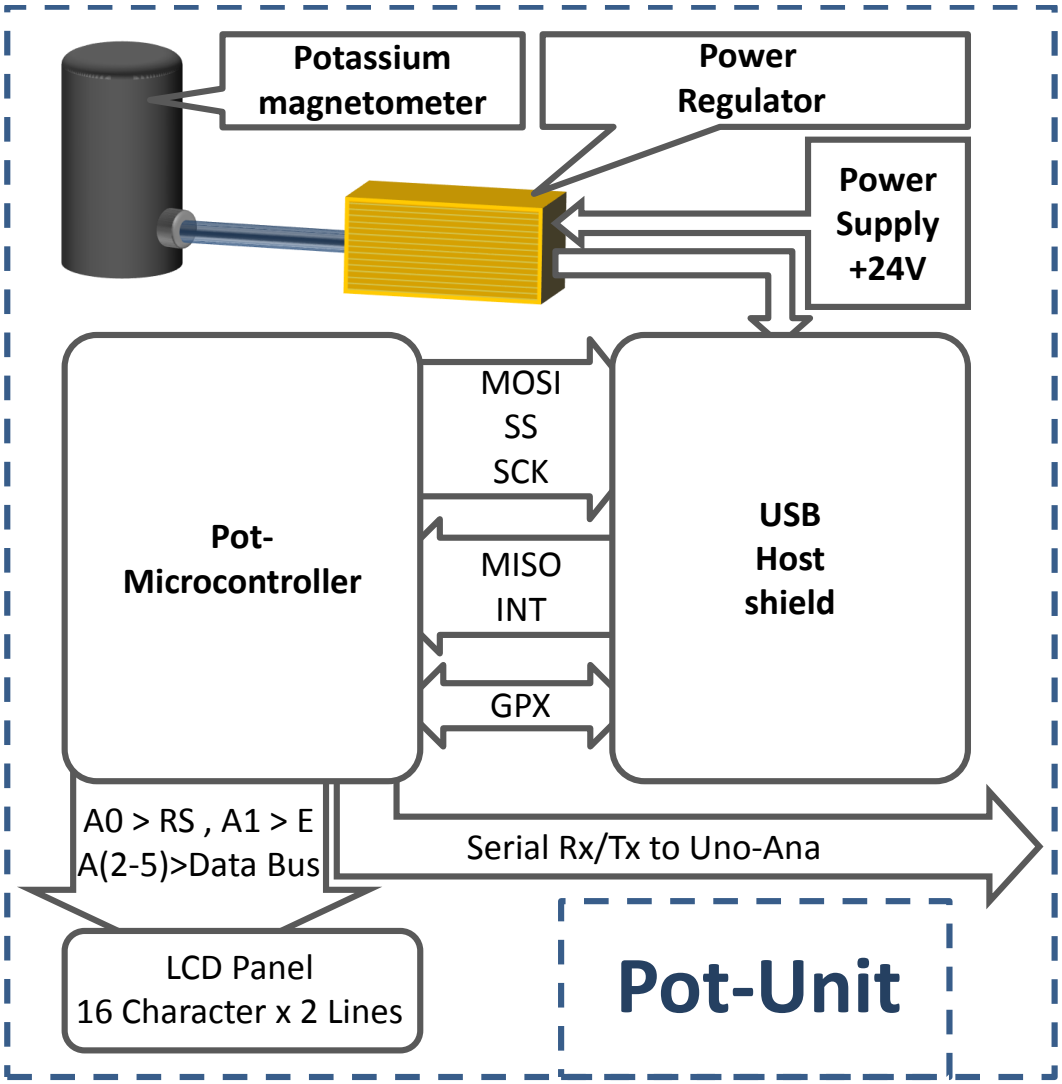


Figure 3.7: *Electronic design of Pot-Unit*

4

Results

While doing this project a vast amount of result have been processed in the strive to optimize and understand the ABZ magnetometer. A smaller amount of result are present in this chapter, this selective part of results, aim to illustrate the forces and weaknesses.

The Bayesian method became an extremely helpful tool to observe the model parameters, even though the method is slow, it visualise the correlation between model parameters. This is seen in Evaluate numbers of model parameters, where it is also learned that we need prior information on S, α and δ to estimate H_E and Z_E.

4.1 Evaluate numbers of model parameters

In this section 1000 measurements with 20 Hz sampling rate is used as the data set. The 20 Hz Potassium magnetometer data with three level coil current is seen in Figure 4.1.

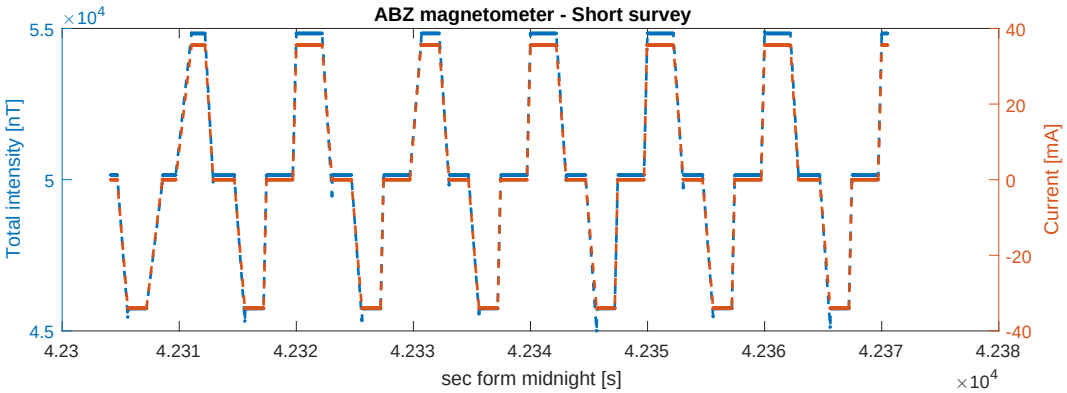


Figure 4.1: *Total intensity and current form a survey of 1000 measurements is illustrated with a two y-axis cordinate system.*

The data seen in Figure 4.1 is all the information required to accomplish the model parameter determination.

In Figure 4.3 the Bayesian method is implemented with five model parameters, and with no prior information. A linear trend is seen between S and α, this imply a correlation between these two model parameters.
In Table 4.1 the estimated model parameters are listed for the differed method, it can be compared with FGM data (FGM with baseline).

Equal step method is far off the reality. Between numerical optimization and MAP a rather high agreement on both H, Z and S, but with a disagreement on both α and δ.

Table 4.1: *Model parameter estimation*

	$F_E\,[nT]$	$H_E\,[nT]$	$Z_E\,[nT]$	$S\,[nT/mA]$	$\alpha\,[mRad]$	$\delta\,[Rad]$
FGM	50221.45	17326.74	47106.77	$\sim$	$\sim$	$\sim$
ES	50150.59	31065.14	39370.53	132437.57	$\sim$	$\sim$
Numerical	50150.59	17053.92	47161.28	138.66	9.65	0.62
MAP	50150.59	17059.49	47159.25	138.68	12.98	-0.91

The measured effect of small change in α can be seen in Figure 4.2, and to assume α or δ is zero, would create big residuals and is therefore not a good solution. Se the theoretical show of residual in Figure 2.3.

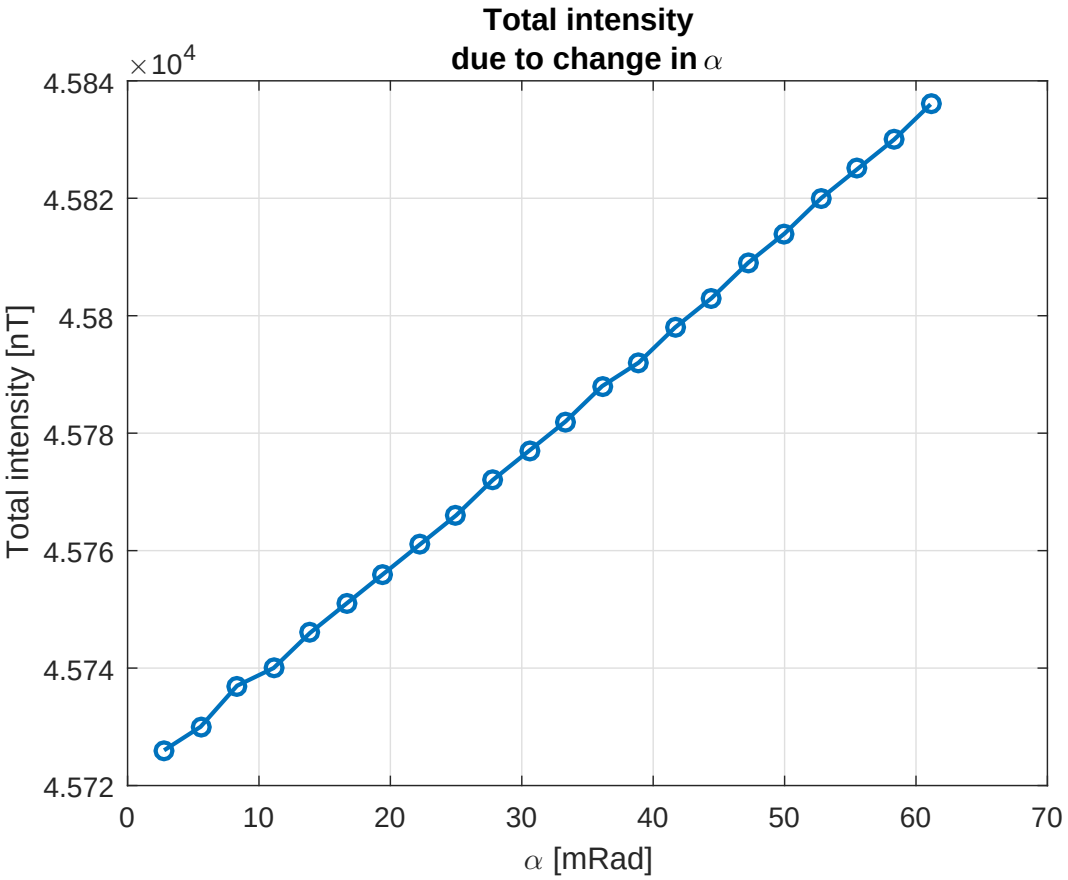


Figure 4.2: *The total intensity is measured, while changing α by 2.7 mRad per measurement.*

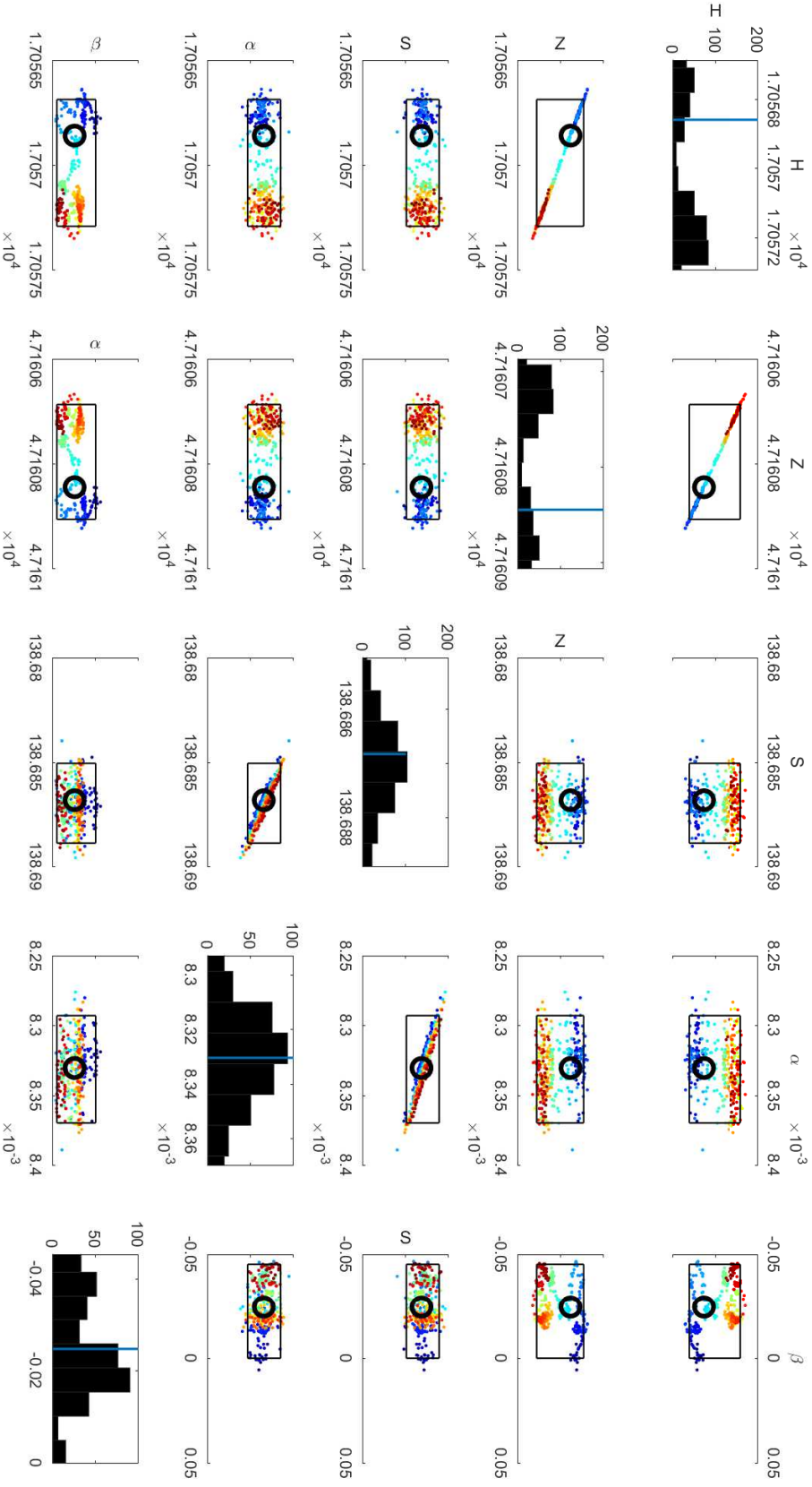


Figure 4.3: *Model distribution of MCMC method, note the high correlation between S and α.*

4.2 Estimation of model parameters

The determination of model parameters was tried for two set-up. Firstly the second is the ABZ magnetometer is considered. Secondly the Overhauser ppm inside a big Helmholtz coil is examined(see Appendix E) .

4.2.1 ABZ magnetometer

Sadly there was not enough time to make a complete estimation of the field with the ABZ magnetometer. However, the estimation of S , α, δ and I_{offset} was estimated.

In Figure 4.4 the estimation of S, α, δ and I_{offset} with used of the Bayesian method are shown, note the more oval-shaped distribution between S and α compared to Figure 4.3. In Figure 4.5 a prediction of the data with the estimated S, α and δ is seen. It is expected to fit the measurements, and in Figure 4.6 it is seen how well this estimation of prior information.

In Table 4.2 the estimation of S, α, δ and I_{offset} is listed.

Table 4.2: *S, α, δ and I_{offset} estimation, for the ABZ magnetometer*

	S $[\frac{nT}{mA}]$	α [°]
MAP	-134.6724	0.0149
95% CI	[-134.6725,-134.6722]	[0.014906,0.014913]

	δ [°]	I_{offset} [mA]
MAP	-3.6968	-1.4288
95% CI	[-3.6968,-3.6967]	[-1.4289,-1.4288]

Table 4.3: *S, α, δ and I_{offset} estimation, with the pre-installed Matlab function and the home-made function. Furthermore the iteration and computation time of is seen for each of the estimation method. The norm of step is the last step.*

	S $[\frac{nT}{mA}]$	α [°]	δ [°]	I_{offset} [mA]
Pre-installed	-134.6724	0.0149	2.5864	-1.4288
Homemade	-134.6724	0.0149	2.5864	-1.4288

	Iterations	Computations time [s]	Norm of step
Pre-installed	8	0.58	3.6e-06
Homemade	9	1.22	2.7e-06

There is a different in δ for MAP and numerical estimation. But since $-3.6968+ 2 \cdot \pi = 2.5864$ its the same. The computation time for the MCMC sript was 374 s for 410.000 posterior distribution samples.

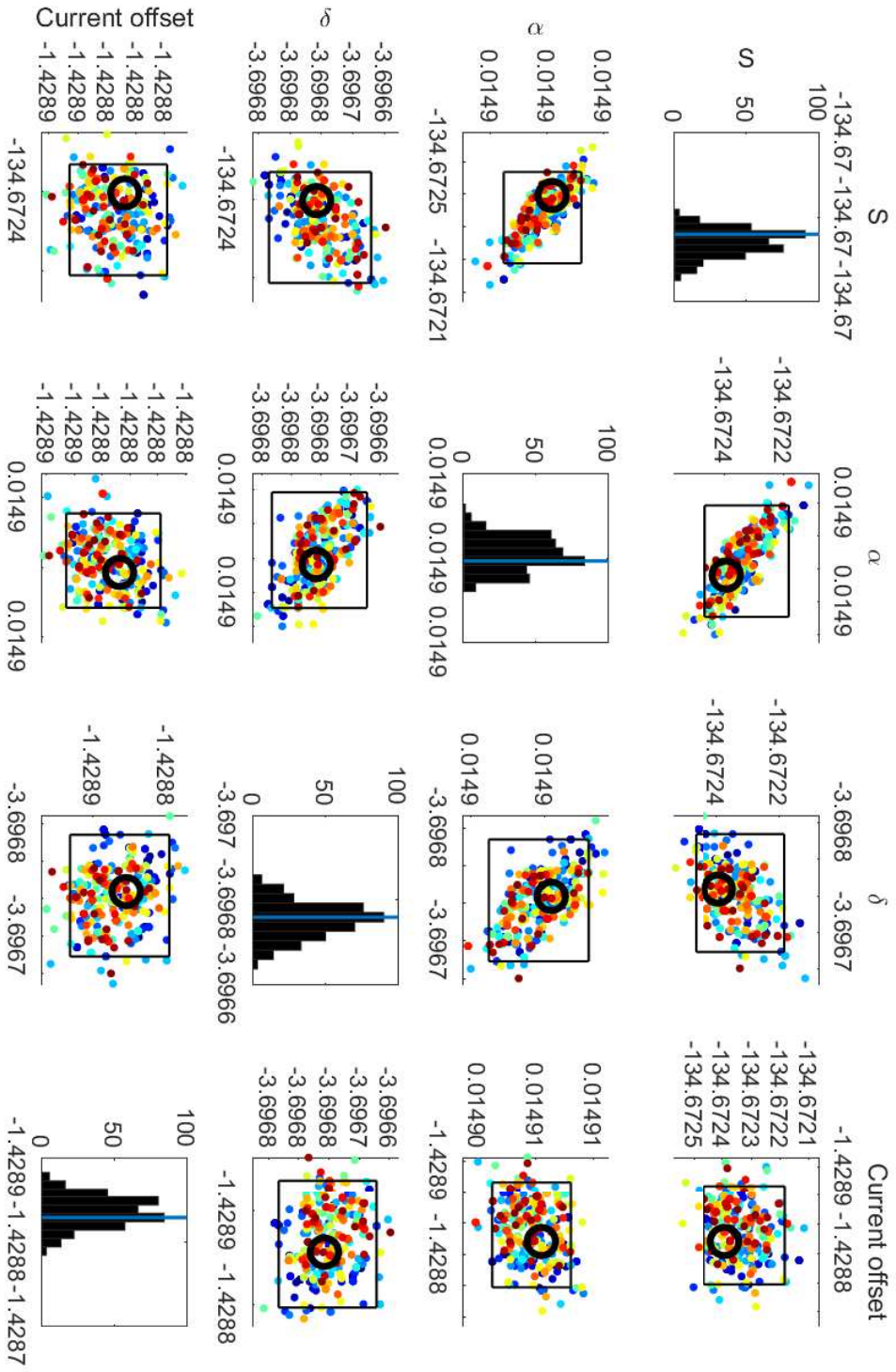


Figure 4.4: *The model distrubution of S, α, δ and current offset (I_{offset})*

.

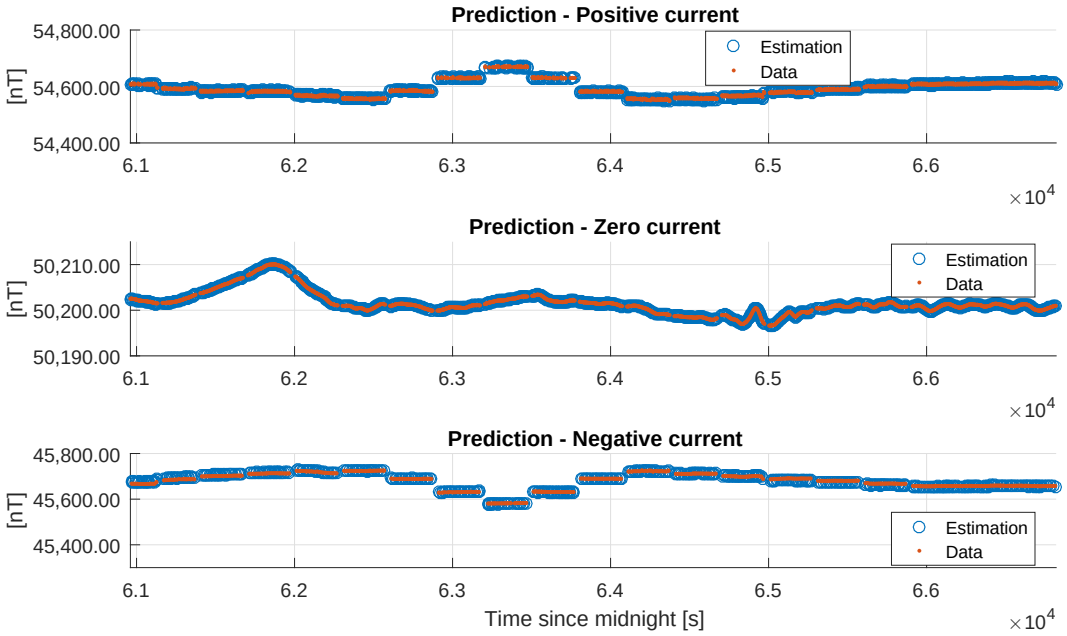


Figure 4.5: *Fitting to ABZ magnetometer data with the estimated S, α and δ*

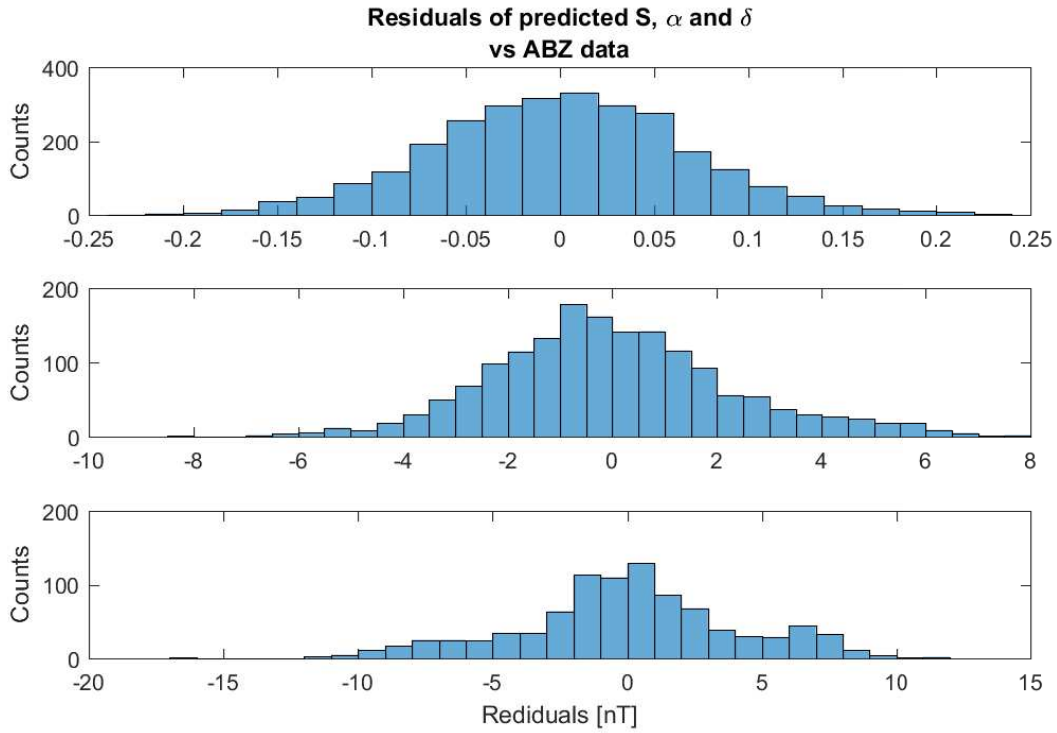


Figure 4.6: *Residuals of fit seen in Figure 4.5*

.

4.2.2 Overhauser ppm in Helmholtz coil

As described in method section a prior estimation of S, α and δ is needed. This include FGM measurements.
The data used in this estimation is 5 Hz measurements with an Overhauser ppm. In Figure 4.7 the total intensity are separated into positive, negative and no current, each of this are compared with the total intensity measured by the FGM.

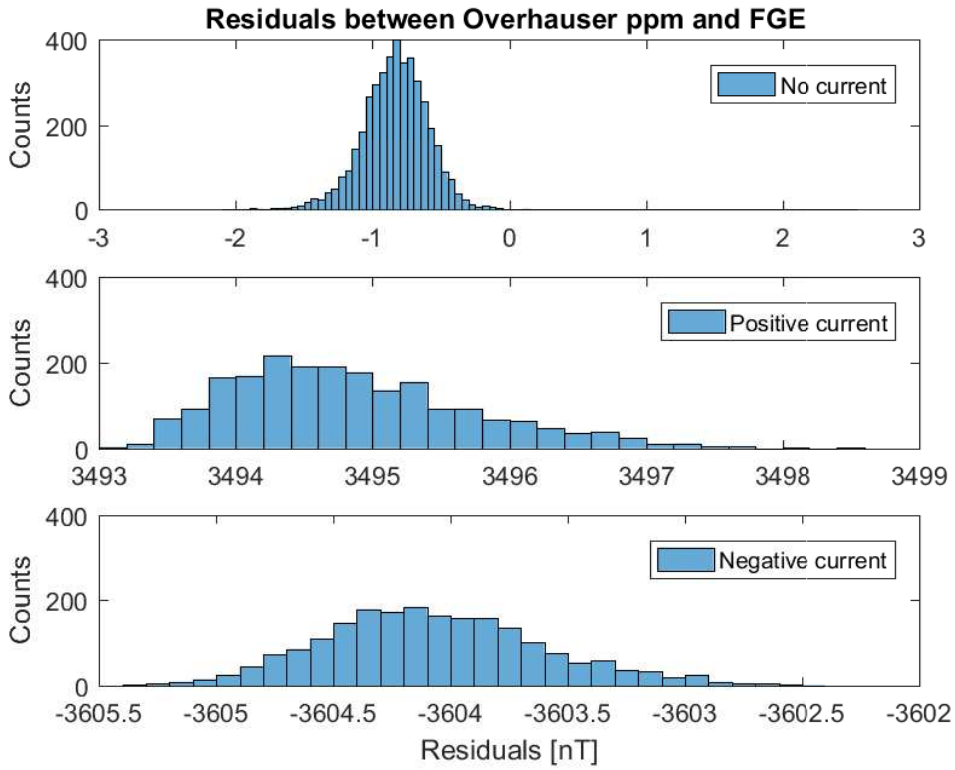


Figure 4.7: *Note the standard deviation of residuals for both the positive and negative current.*

When applying no current a normal distribution arise with a small offset on $\mu = -0.8nT$ as seen in Table 4.4. This may very well come from a offset in sites, and there is no reason for concerns.

Table 4.4: *Evaluation of the distribution of residuals from Figure 4.7*

	No current	Positive current	Negative current
μ	-0.84 nT	3494.85 nT	-3604.07 nT
95% CI	[-0.84, -0.83] nT	[3494.82, 3494.89] nT	[-3604.09, -3604.05] nT
σ	0.25 nT	0.87 nT	0.47 nT
95% CI	[0.24, 0.25] nT	[0.85, 0.90] nT	[0.46, 0.49] nT

A estimated value for S, α and δ is calculated with Bayesian method, this is showed inTable 4.5. The estimation prior estimation of S, α and δ, enable the

estimation of F_E, H_E and Z_E.

Table 4.5: *S, α and δ estimation with Overhauser and Helmholtz set-up*

	S $[\frac{nT}{mA}]$	α [°]	δ [°]
MAP	153.8298	-0.006792	0.0008961
95% CI	[153.829, 153.830]	[-0.00680, -0.00677]	[0.000896, 0.041634]

In Figure 4.8 the root mean square of the residuals is plotted vs window size. From Figure 4.8 a window size of 3 minutes is chosen for the estimation in Figure 4.9. The Residual between FGM and estimated F_E, H_E and Z_E is seen in Figure 4.10.

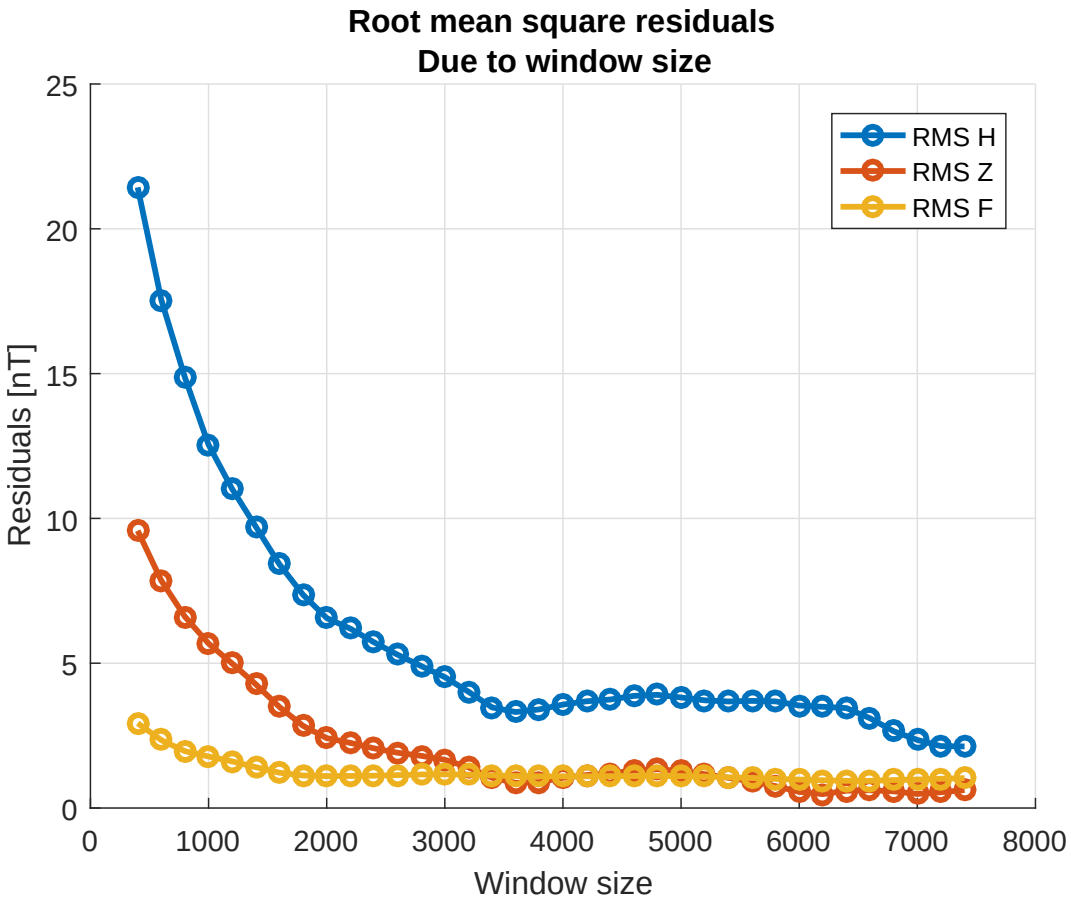


Figure 4.8: *RMS vs. Window*

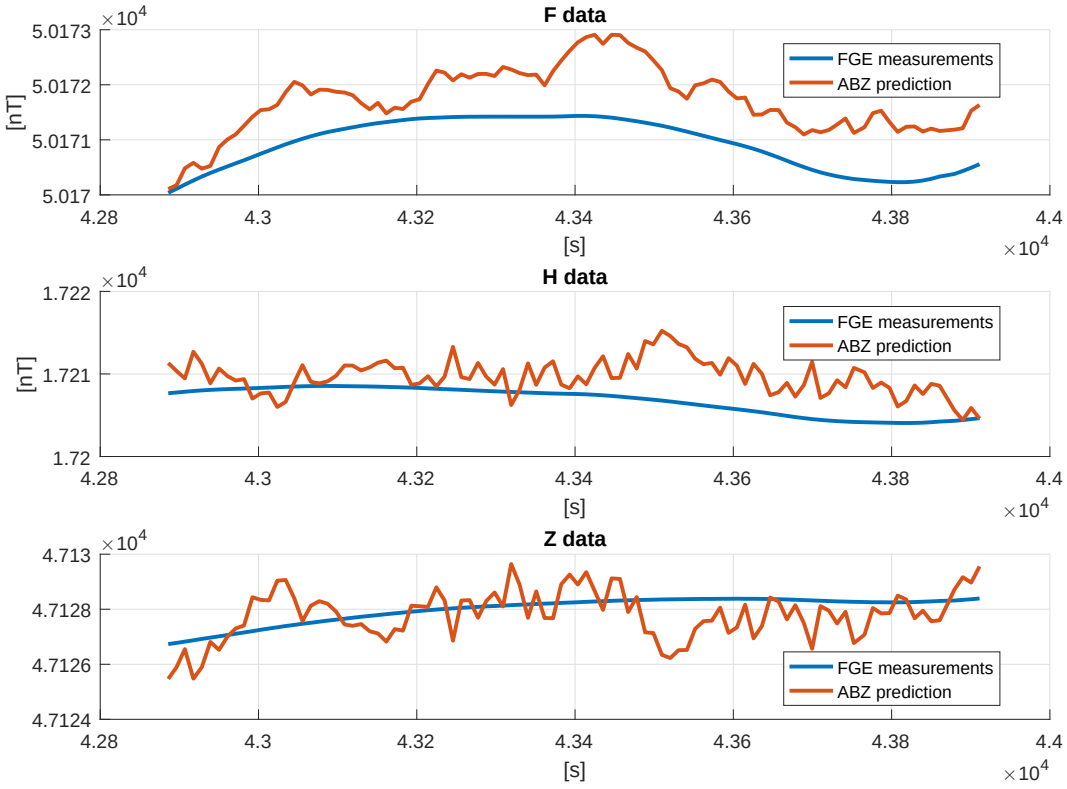


Figure 4.9: *Estimation with a window of 3 minutes*

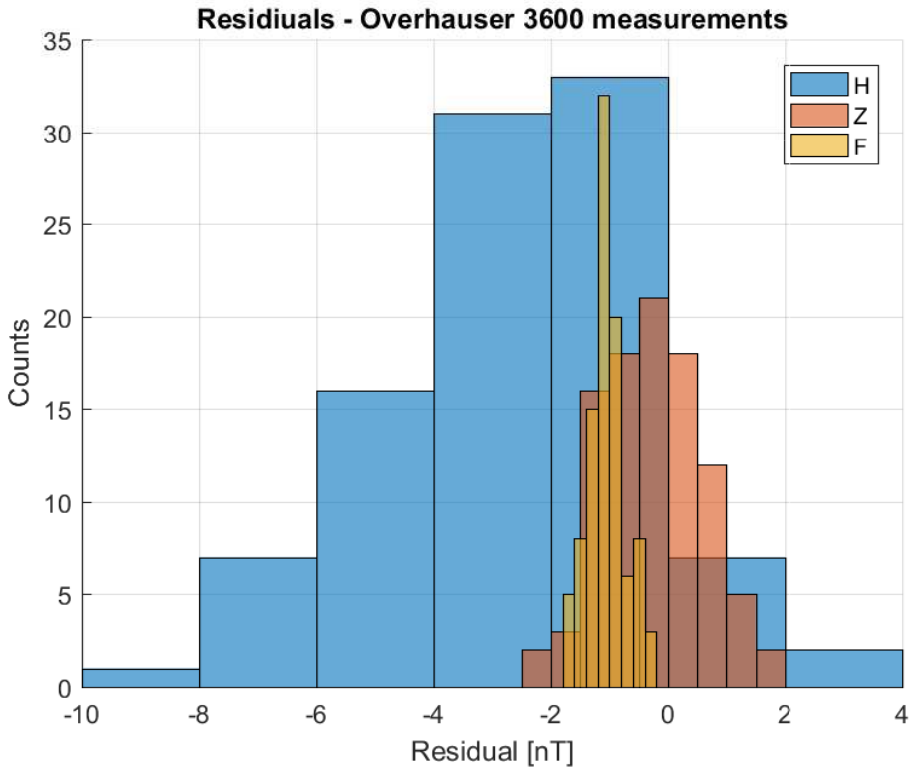


Figure 4.10: *Residuals with a window of 3 min*

.

4.3 Unaccounted-for

In the process of error detection, here are some material for potential further examination. In Figure 4.5 a signal generated by the Potassium magnetometer is showed. The picture is taking of a oscilloscope, and the signal frequency estimation on 32 MHz is seen in the lower right corner.

In Figure 4.12 and Figure 4.13 a illustration of three level current are seen measured with Potassium magnetometer and Overhouser ppm, respectively. a slope that are uncorrelated to the geomagnetic field is seen, the slope could neither be seen in the coil current measurements.

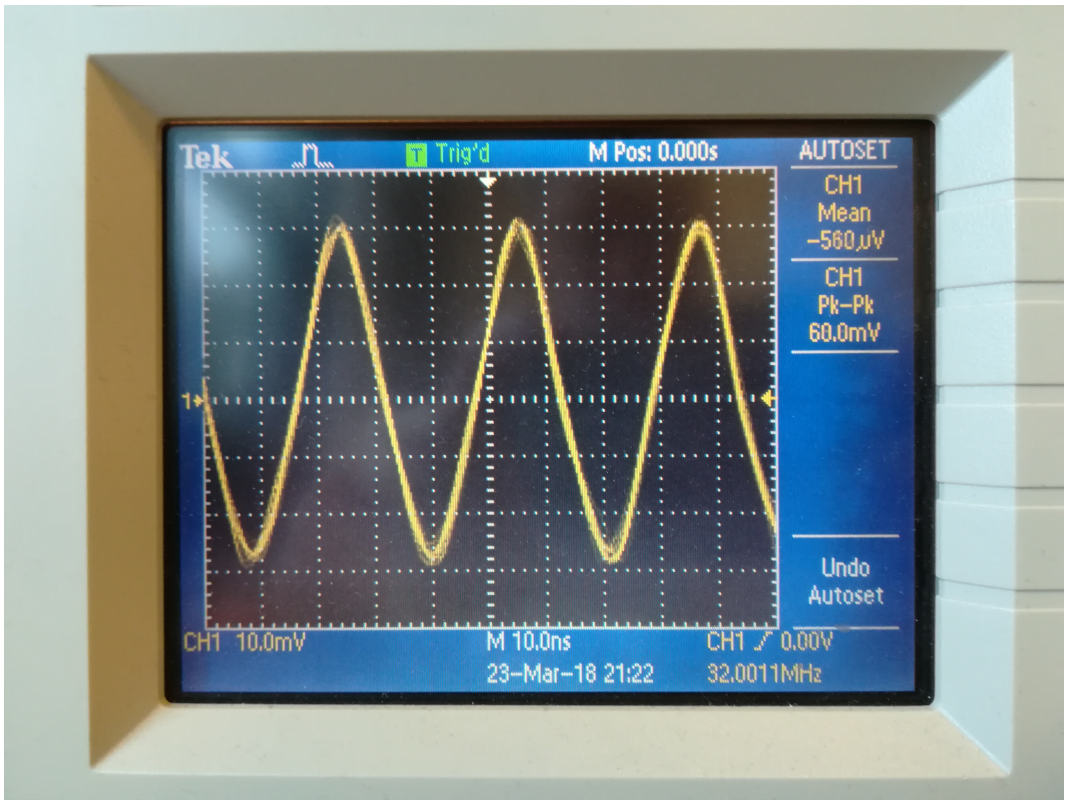


Figure 4.11: *A 32 MHz signal was detected in the coil system, when the potassium magnetometer is turn on. The signal clearly occured as soon as the potassium magnetometer started measureing.*

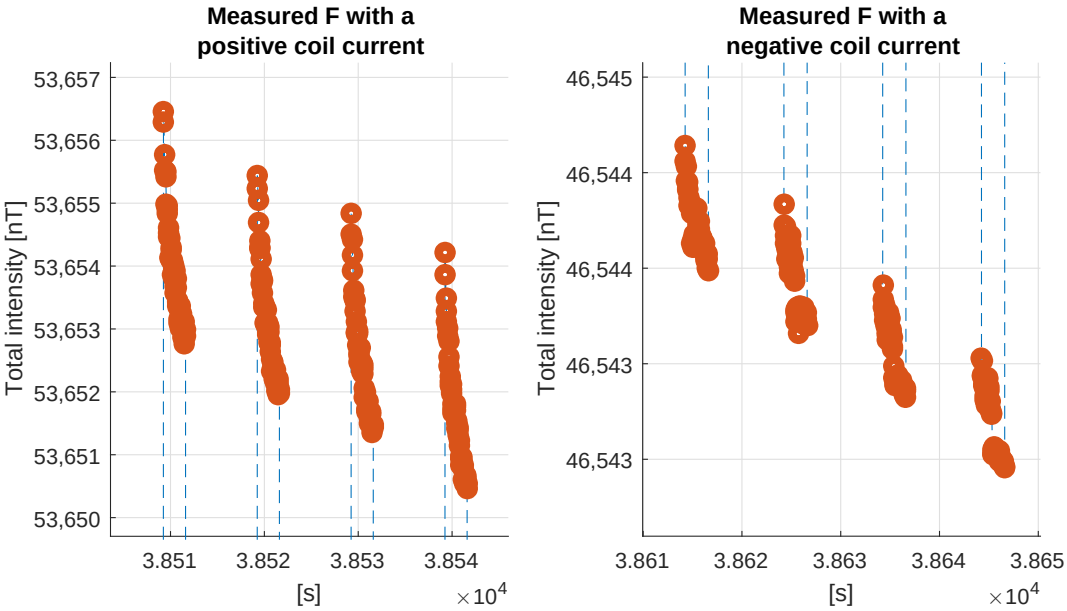


Figure 4.12: *This is total intensity measurements with the Potassium magnetometer inside a Lee-withing coil system. Four steps are seen with a positive and negative coil current.*

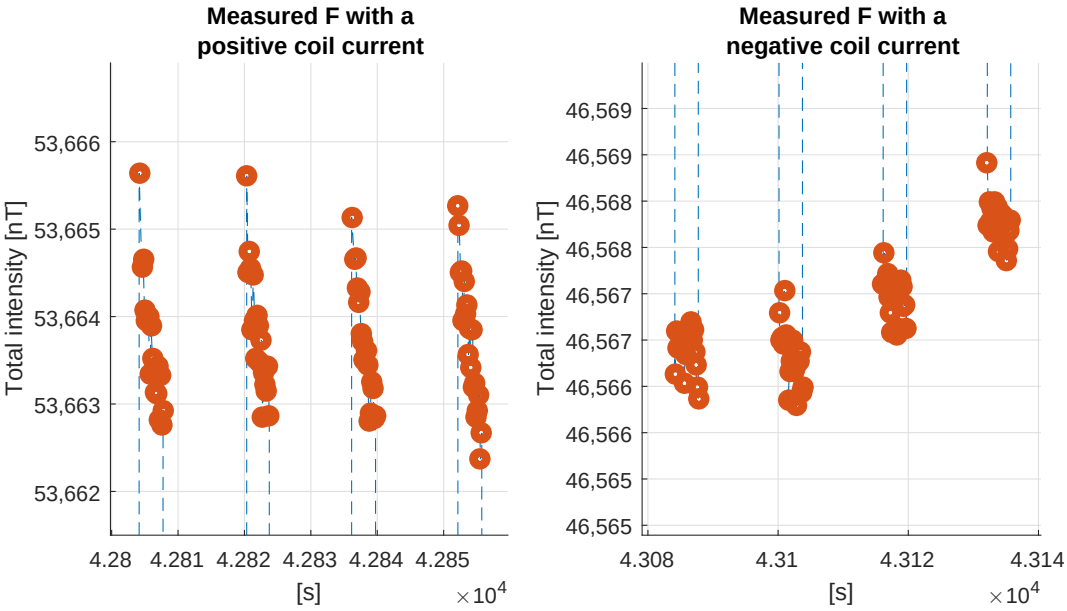


Figure 4.13: *This is total intensity measurements with the Overhauser ppm inside a Helmhotz coil system. Four steps are seen with a positive and negative coil current.*

.

4.4 Baseline improvement

A baseline improvement is implemented on a 20 hours dataset form the geomagnetic station in Thule. In Figure 4.14 the $\phi(t, S^{i,j,k})$ form Equation 2.20 is plotted with the original baseline and estimated improved baseline. Note the 10^{-10} on the y-axis for estimated parameters. In Figure 4.15 the linear baseline estimation, baseline improvement and a hourly mean of the baseline improvement is showed. λ in the optimization of Equation 2.20 is set to 0.5, which was a recommended value from (Soloviev et al., 2018).

The pre-installed Matlab function `lsqcurvefit();`, described in subsection 3.1.3, has been used to minimising the objective function $\phi(t, S^{i,j,k})$.

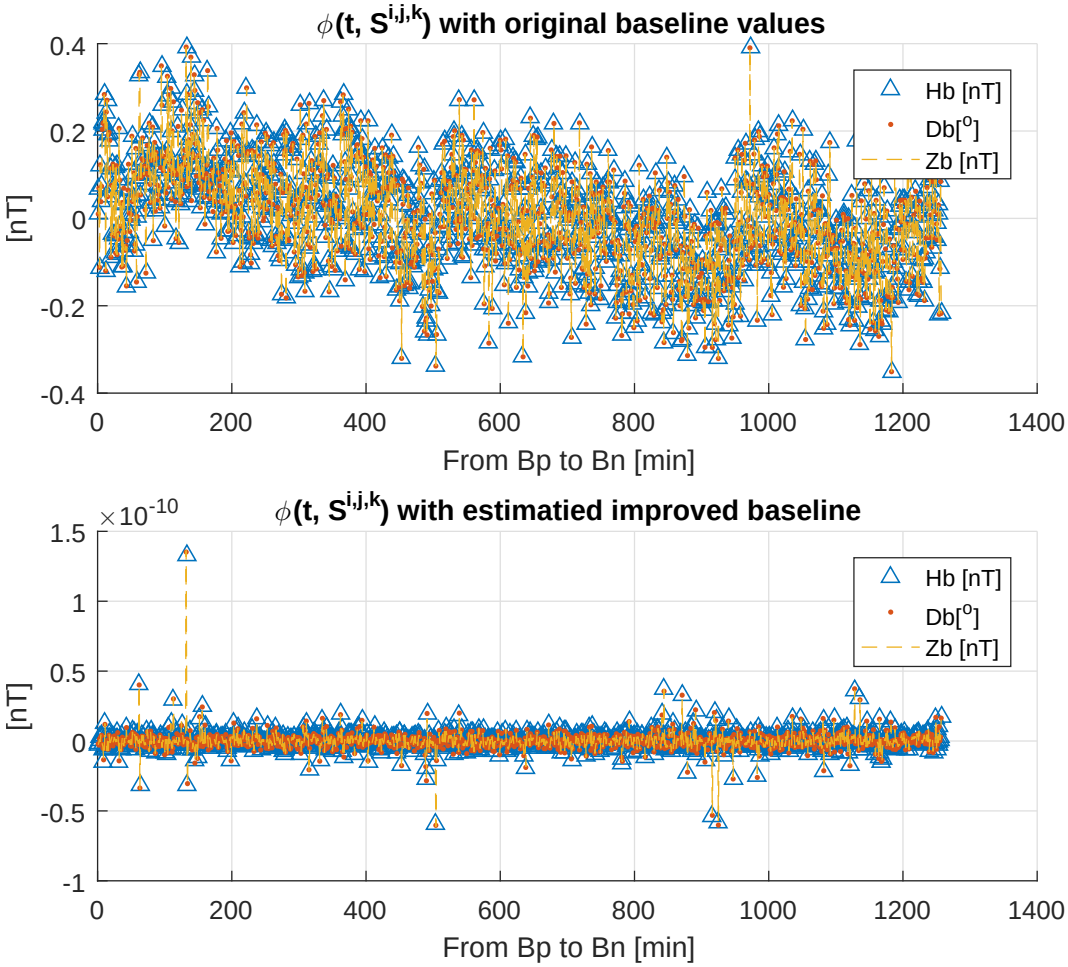


Figure 4.14: *$\phi(t, S^{i,j,k})$ for orginal baseline and estimated baseline value. Note the small value*

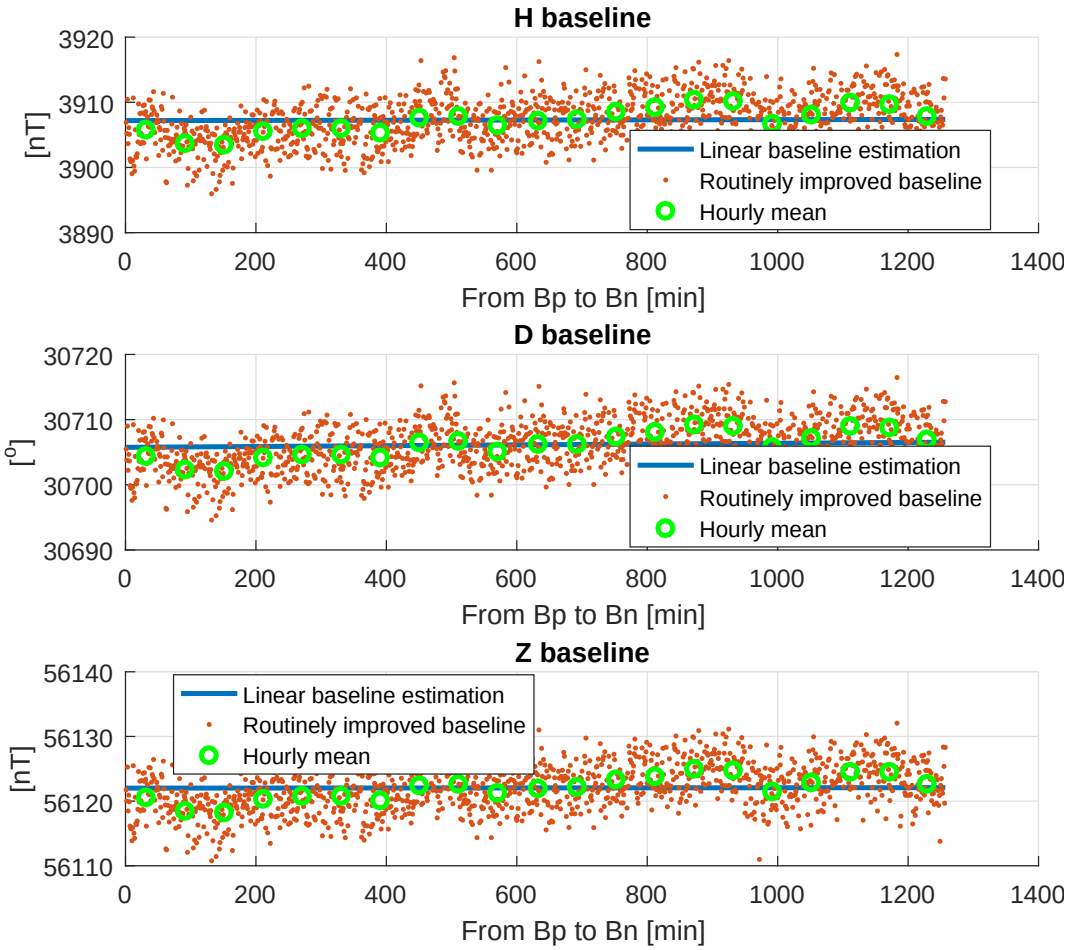


Figure 4.15: *The three components of the baseline is seen for both the linear baseline estimation, improved baseline and its hourly mean.*

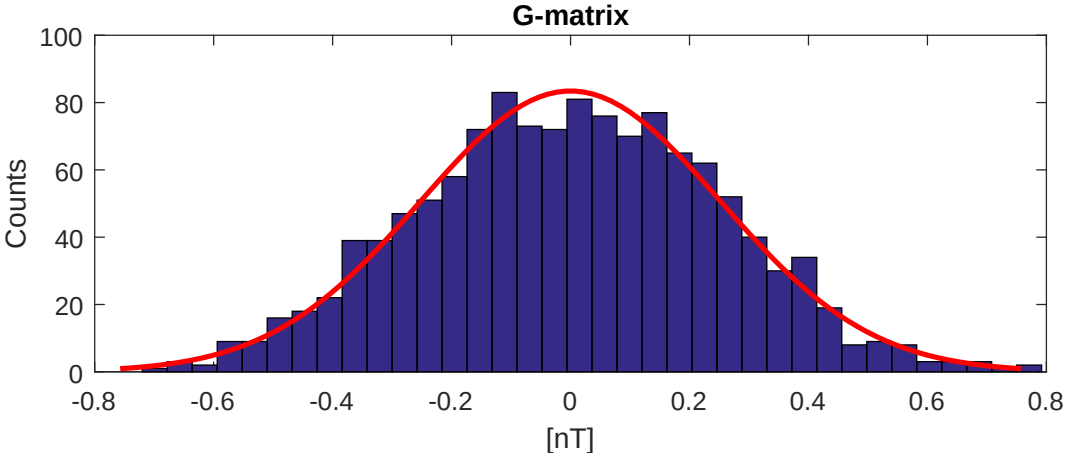


Figure 4.16: *Histogram of the G-matrix.*

5

Discussion

The final objective for this thesis was to find and verify an effective way of deducing H_E, Z_E and F_E with the ABZ magnetometer. While this was not fully accomplished, good insight in the ABZ magnetometer was established, and the method implemented has shown to be promising.

The limiting factor for this project has become the time limitation, which is not surprising with such a far-reaching field of work. Nonetheless, it is hoped that this thesis has brought some light on how to deduce H_E, Z_E and F_E, and can be used in future work.

At the start of the thesis, an initial project description was formulated (appendix G), which included a description of the expectation. In combination with the initial project description a time schedule was conducted. The time schedule has become heavily obsolete, not only because the schedule has changed, but also the activities of the thesis. The old and a new time schedule can be found in appendix F.

Once again a separation is made to create an overview. Each section can be read separately.

5.1 Computer system

The limiting factor for the computer system is the single operating serial-channel, which brought limitations to the functionality of the computer system.

Theoretically a computer system with GPS, SD card, coil system and Potassium magnetometer could work with only one operating serial-channel, but with the amount of computer tasks this becomes risky. The system has to receive serial data from both GPS, ADC and Potassium magnetometer data and has to be ready to receive new data. With the combination of 20 Hz potassium magnetometer data and aperiodical GPS data, the Arduino UNO does not compute fast enough. It could work with a lower frequency data-flow from Potassium magnetometer, which would create time to capture GPS data. But there will always be times where both GPS and Potassium magnetometer would transmit at the same time. This could be resolved by using the Arduino MEGA or Arduino DUE which both have 4 serial channels.

An update to at least a Arduino DUE (Figure A.10) could move the computer system to the next level, this would enable accurate time stamp with a serial-channel allocated only to do so. Furthermore, saving data on an external memory like a SD Card would not create any problem. This simple update would solve many issues for the current system, but also demand time for adjusting the code.

The excellent sampling rate of $100\,\frac{\mathrm{kS}}{\mathrm{s}}$ was not used to its full potential, if the ADC could measure the coil current with $100\,\frac{\mathrm{kS}}{\mathrm{s}}$ it could be used to remove residuals in the coil current measurement. Again the DUE would create this possibility with the extra serial-channels.

If a live display of H_E and Z_E should be implemented a lot of computations are needed. This would probably also be to much for an Arduino DUE. It could either be accomplished by a serious microcontroller update, or by an external computer.

There is no long term test of the system, and therefore no long time robustness of the computer system can be discussed.

There can occur a small delay between Potassium measurements and coil current measurement, this can be due to wire length or computation time. This small time delay create problems for the determination of model parameter. The error is significantly higher for a ramp function than a current step of equal polarity.

However, even though the computer system was very primitive and first prototype, it managed to get useful results for the estimation of model parameters. A new updated computer system with multiple serial-channel, and with focus on GPS, coil current measurements and good synchronization of data, would have bright prospects.

5.2 Estimation of model parameters

Estimation of model parameters has been separated into two steps, where the coil sensitivity, vertical- and horizontal misalignment of the coil suspension were initially estimated to a constant value. There is a big potential in this approach, and a result of this is shown, with an Overhauser ppm and Helmholtz coil system, in Figure 4.9. A window size of 3 minute gave residuals up to a max of 10 nT. However, the residuals follows a Gaussian distribution, with a mean close to zero as seen in Figure 4.10, which is good tendency. This is neither a good window size or residuals, but it is proof of concept.

Applying the method on a set-up with an Overhauser ppm and Helmholtz coil quickly gave promising results. Sadly it was not quite as easy with the ABZ magnetometer, and it was not until very late in the project that a current offset was implemented in the estimation of S, α and δ. The current offset was only implemented for positive coil currents, this is in itself suspicious. It

has also been tested with a negative coil current offset estimation, it was in magnitude of $1\,\mu$A and can therefore be neglected. It has not been verified where the offset in the coil current comes from, but presumably a correlation between this and the 32 MHz signal seen in Figure 4.11 could be justified.

A 32 MHz current signal is generated in the coil system when the Potassium magnetometer is turned on. This fits our expectation for the generated radio frequency of 35 MHz that is created for a 50.000 nT magnetic field, when working with atomic or nuclear magnetic resonance effects. It is therefore very reasonable that this 32 MHz signal comes from the Potassium magnetometer, but it was a bit surprising that this signal created a current in the coil system with this high amplitude (Figure 4.11). The model or computer system does not take this stray signal into account. The stay signal could cause the current offset seen in the ABZ magnetometer.
Together with the theoretical expectation that larger magnetic field create higher frequency, and therefore create different current offsets for F_p, F_m and F_0 (Bjerg, 2017b, p 9-13).

If there were a smart way of incorporating the 32 MHz current signal as part of the problem or measure it, that would solve the issue. It could be tempting to make a lowpass filter in either the computer system or in model parameter estimation, but the signal would still be there, just not measured any more. Alternatively the radius of the coil system could be expanded until the radio frequency would have a negligible impact on the coil current. However this would contradict the concept of the ABZ magnetometer, where a portable ABZ magnetometer unit can measure remote places. This is probably the reason why the offset current is not seen in the setup with Overhauser ppm and Helmholtz coil, which further confirm the suspicion.

In subsection 4.2.1 Bayesian approach and numerical method has been tried to estimated S, α and δ, is gave the same results on all estimated model parameters. With the slow Bayesian method provided a visualisation of the model parameters that is critical for ensuring uncorrelated model parameters. These estimation of model parameter is very satisfactory. The estimation for this data set is included in appendix B.

The ramp current was not implemented. In the sense of finding a good method for the ABZ magnetometer this is not a critical part. Originally the idea of the ramp current function was invented to cope with the fact that the Potassium magnetometer loses its signal if the field is changing to fast. A ramp function would therefore create more useful Potassium magnetometer data, thus a smaller window size in the computation, which results in an improved time resolution of the geomagnetic field.

In Figure 4.12 and Figure 4.13 is a visualisation of an other behaviour that is unaccounted for. It arise no-matter the set-up. It occurs when a coil current is applied.

There are not made any long time measurements of any sort. Over longer periods of measurement, it is expected that the S would change due to temperature change and other material deformation of the coil. If an earthquake or other seismic signals were seen in the area of the ABZ magnetometer, α and δ would oscillate. The assumption of a constant value for S, α and δ is therefore incorrect. In appendix Equation D.1 an method using prior distribution is introduced, this may be a solution to the long periods issues.

5.3 Data specification

The data specification refers to Table 1.2 from Turbitt et al. (2013), shortly introduced in the "Introduction". It states general definitions if one-second measurements are desired, and it is therefore reasonable to use the data specification as a measure for the ABZ magnetometer.

Regarding the data specification for the ABZ magnetometer, there is a long way to go before these are achievable, but nonetheless it is a good indicator where the pitfalls are.

Many of the specifications are related to the computer system, and the most crucial factor here are presumed to be the time-stamp accuracy, phase response and coil current measurements accuracy. The time-stamp accuracy $\leq 0.01\,\mathrm{s}$ should be a possible task since GPS signal have an accuracy down to 10 ns (`www.atomic-clock.com`), but with a sampling rate of max 10 Hz an internal cock is required. The phase response $\leq \pm 0.01\,\mathrm{s}$ is more difficult, and especially the phase difference between the coil current measurement and the Potassium magnetometer data. A scenario where each measurement is out of phase would create problems. An attempted survey with ramp function gave unusual results, and it is believed that the unsynchronized coil current and Potassium magnetometer data was the main problem.

The instrumental amplitude range was $\leq \pm 4000\,\mathrm{nT}$ and $\leq \pm 3000\,\mathrm{nT}$ of high and mid/equatorial latitude, respectively. The scalar magnetometer is the limitation for instrumental amplitude range, especially the dead zones of the scalar magnetometer. The dead zone for the Potassium magnetometer is $\pm 10^\circ$ from the horizontal plane and the vertical line. This means that if the orientation is not physical change, it can not measure close to equator and the poles. A solution to this is to add a constant baseline-current to the coil system, which would move the magnetic field vector out of the dead zone, and enable calculating H_E, Z_E and F_E. The down side to a baseline-current is that F_E has to be calculated, and is not measured at any time, F_E would therefore contain a higher uncertainty.

5.4 Baseline improvement

The baseline improvement was a small side project, that could illustrate one of the possible applications. The baseline improvement did only include F_E as extra information, and if the ABZ magnetometer was implemented for baseline calibration, further extra information could be implemented into the optimization. This appear as a very sensible application.

The data possessing conducted on the 20 hours long data set, from Thule geomagnetic station looks promising, with a small, reasonable deviation from the a linear baseline as seen in Figure 4.15. But even though the residuals seen in Figure 4.14 are reduced to below $1.5 \cdot 10^{-10}$ nT it is not validated that the baseline is improved. We would have to know a true value for numbers estimation, to ensure that the baseline is truly improved.

Nonetheless, it is seen that a simple Matlab code with built-in optimization tools can solve the inverse baseline improvement problem. Even if the ABZ magnetometer is not constructed the baseline improvement could be implemented as a general baseline estimation, instead of the old classical way.

6

Conclusion

Our knowledge concerning how to make measurements with the ABZ magnetometer has been enhanced in this master thesis, and an approach for estimating F_E, H_E and Z_E have been described.
The approach propose a two step method, where first step include a prior estimation of the coil sensitivity, vertical and horizontal coil misalignment. Secondly, use this prior estimation in continuous estimation of the horizontal, vertical and total geomagnetic intensities. This approach removes a high correlation between coil sensitivity and vertical coil misalignment.
A current offset for positive coil currents on $-1.4\,\mathrm{mA}$ was discovered late in the project, and the second step has therefore not been finally demonstrated on the ABZ magnetometer. However, the approach has been conducted on a similar set-up, that included a bigger coil system and an Overhauser ppm, the result of this is seen in Figure 4.9. The residual within $\pm 10\,\mathrm{nT}$ and the 3 minutes estimation window is high, but nonetheless promising within the context of evaluating the approach.

The computer system made to collect the data has to be further upgraded, if the ABZ magnetometer should be improved. The most important factor is multiple serial-channels, GPS time-stamps and better current estimation.

The baseline improvement, which has been conducted parallel to the project, can not be finally concluded to be an improvement to the regular baseline, but it appears promising.
The observatory baseline improvement, with total intensity as extra information, has been conducted on measurements from Thule observatory. The results of this appears promising.

A

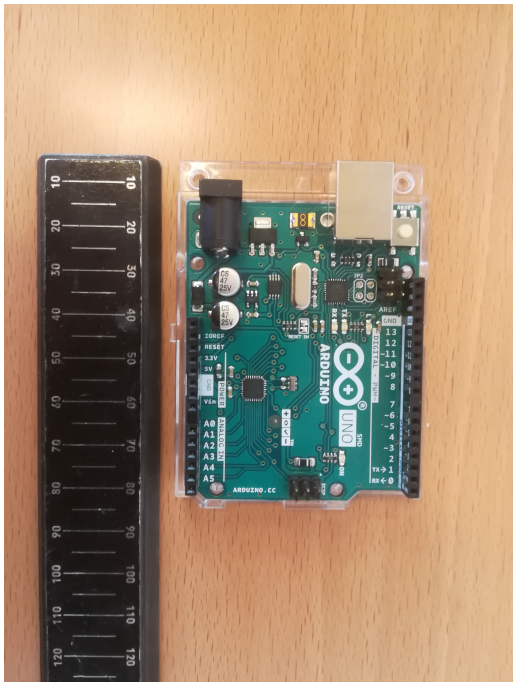
Computer system

A.1 Components

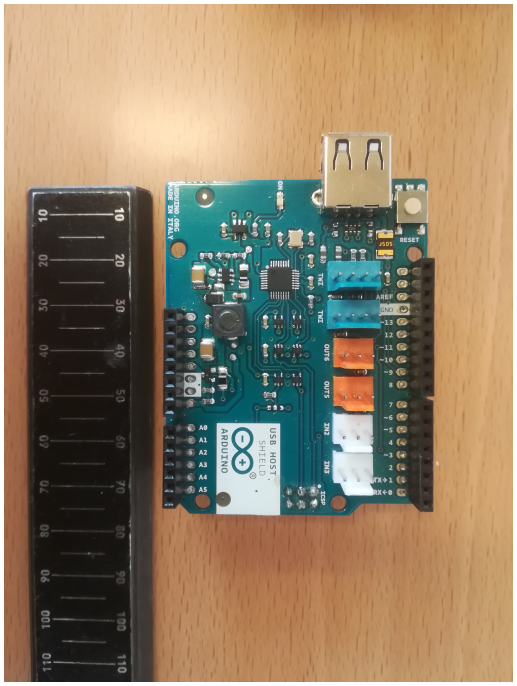
The electrical design was implementation with the electronic components listed down under, a link for information on the specific component is included in the list. The bold word is referring to the electrical design on Figure 3.6 and Figure 3.7. Pictures of all the part listed is showed after the list.

1. **UNO-Ana** & **UNO-Pot**: two "Arduino Uno" a microcontroller board based on the ATmega328P.
   `https://store.arduino.cc/arduino-uno-rev3`
2. **USB Host shield**: Control and unpack data from the Potassium magnetometer.
   `https://store.arduino.cc/arduino-usb-host-shield`
3. **DAC** & **ADC**: "Analog shield" and generate and measure the coil current, with 16 bit 100 ks/s ADC and DAC.
   `https://reference.digilentinc.com/analog_shield`
4. **DC-DC Converter**: ± 15VDC, ± 100 mA, used for generating higher current that a Uno can generate.
   `https://uk.rs-online.com/web/p/isolated-dc-dc-converters/0169235/`
5. **Buffer**: Two LT1010, a Power Buffer used as a op-amp.
   `http://www.analog.com/media/en/technical-documentation/data-sheets/1010fe.pdf`
6. **LCD Panel**: LCD display qapass 1602a, For visual realtime intensity measurement from potassium magnetometer, can be used for vertical calibration.
   `https://www.rhydolabz.com/documents/29/LCD-1602a-yellow.pdf`
7. **GPS**: NEO6MV2, Give time and locations.
   `https://www.xarg.org/2016/06/neo6mv2-gps-module-with-arduino/`
8. **Antenna extension**:
   `https://www.makerfabs.com/GPS-Active-Antenna-3m-with-SMA-Connector.html`
9. **Memory storing**: MicroSD Card Adapter, save data on a SD micro card.
   `https://www.trab.dk/da/breakout/31-micro-sd-board.html`

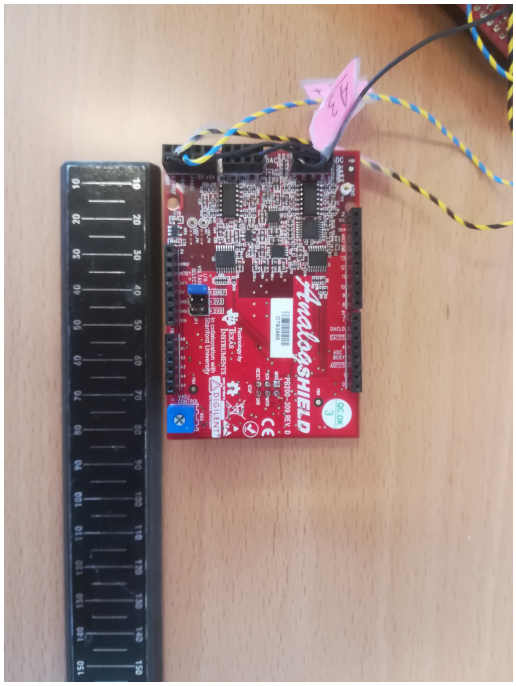
After the picture of the components, pictures of the assembled computer system is showed. A box was 3-D printed for containing the electronic, this simplify the transportations of electronic.



(1): Arduino Uno

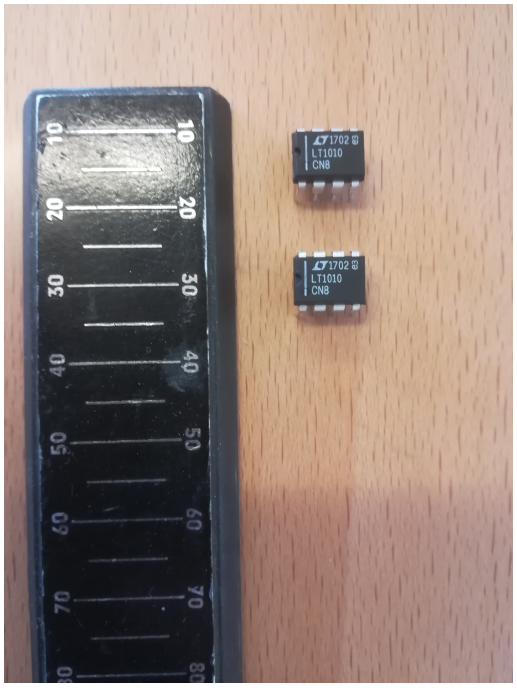


(2): USB Host shield

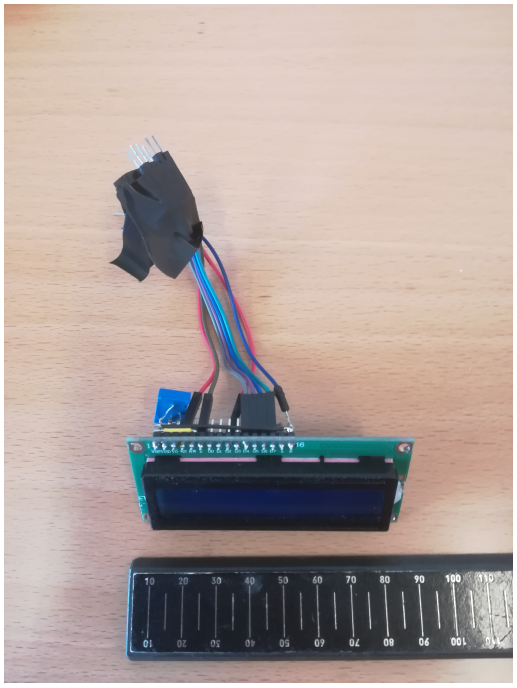


(3): Analog shield

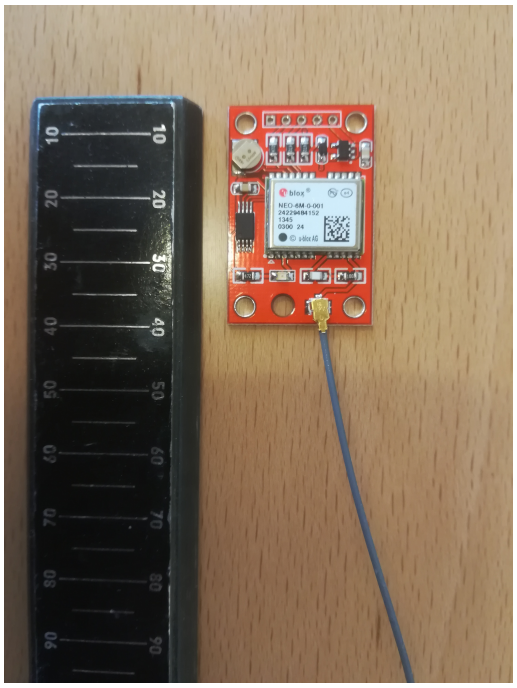
(4): DC-DC Converter



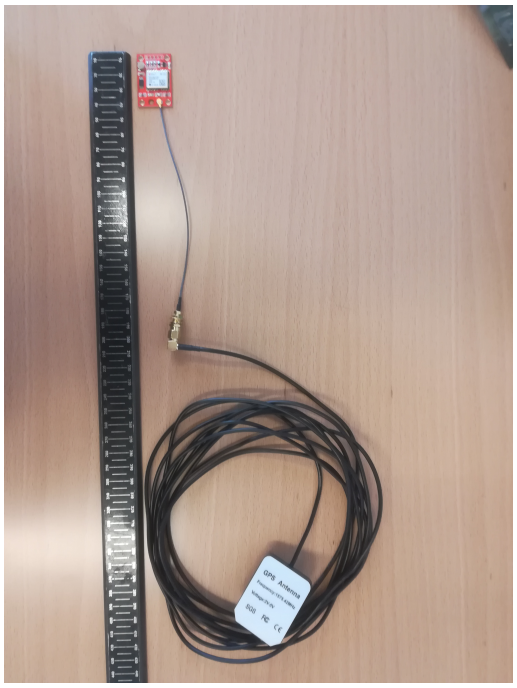
***(5)**: Two LT1010*



***(6)**: LCD display*

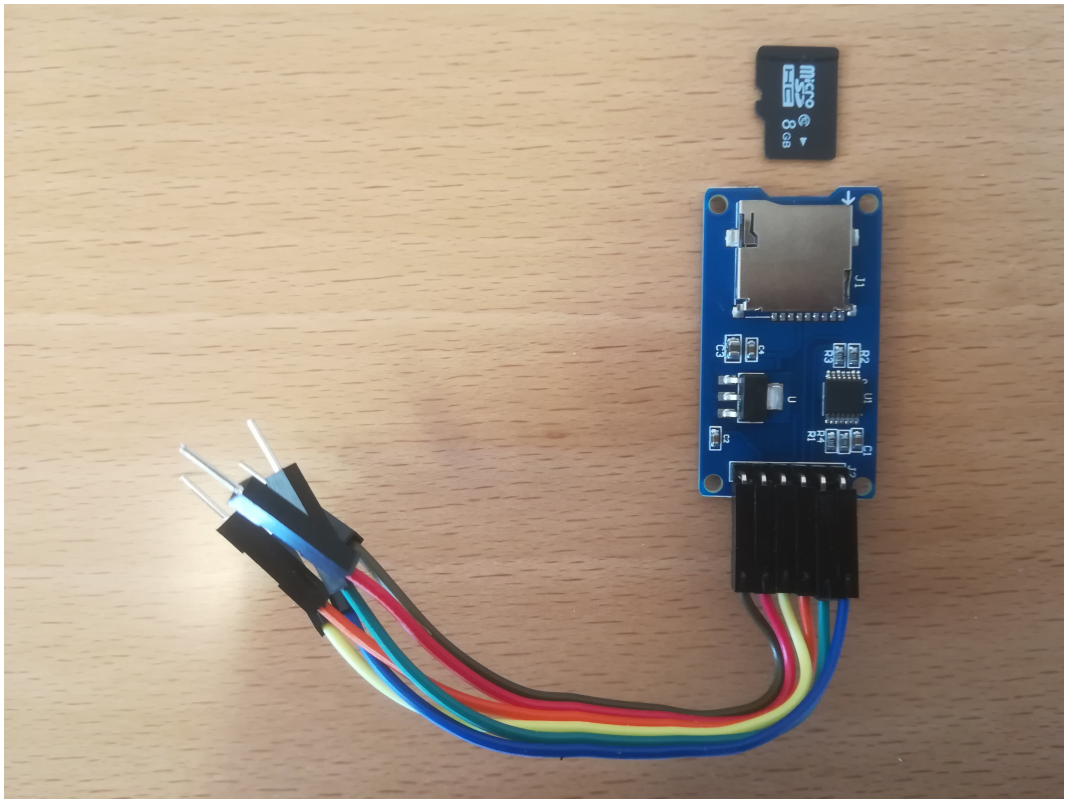


***(7)**: GPS NEO6MV2*



***(8)**: Antenna extension connected to GPS*

.



***(9)**: MicroSD Card Adapter with 8 GB Micro SD Card*

.

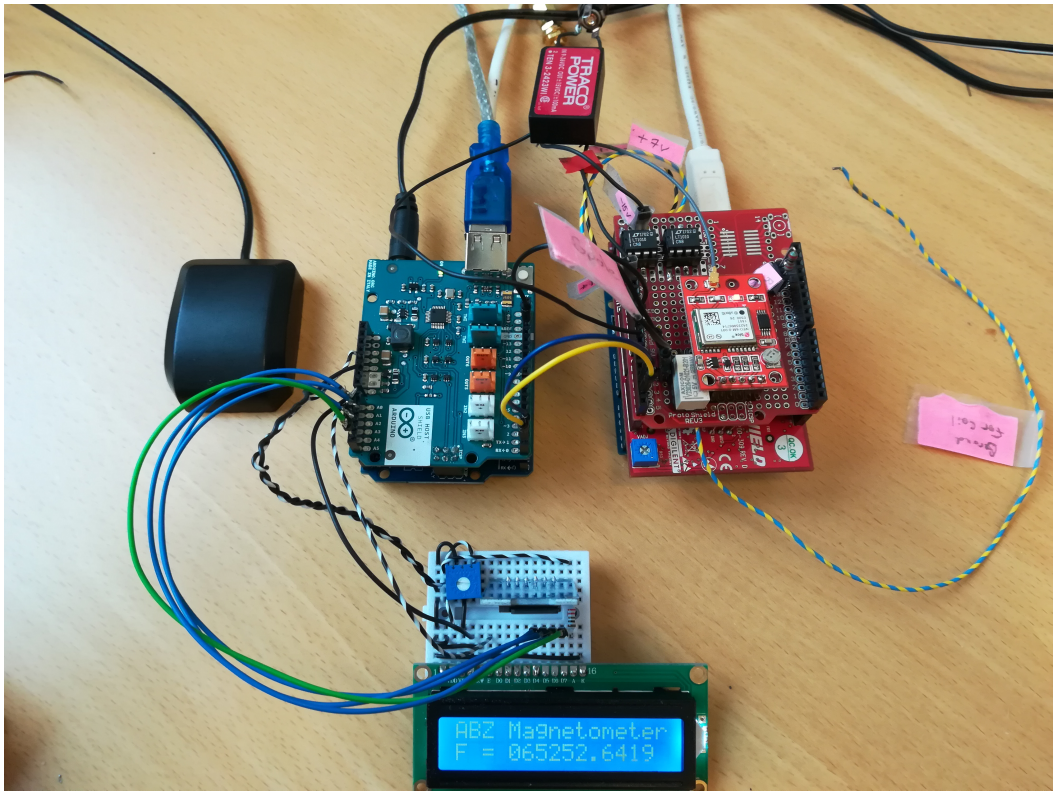


Figure A.6: *The final implementation of the electronik components*

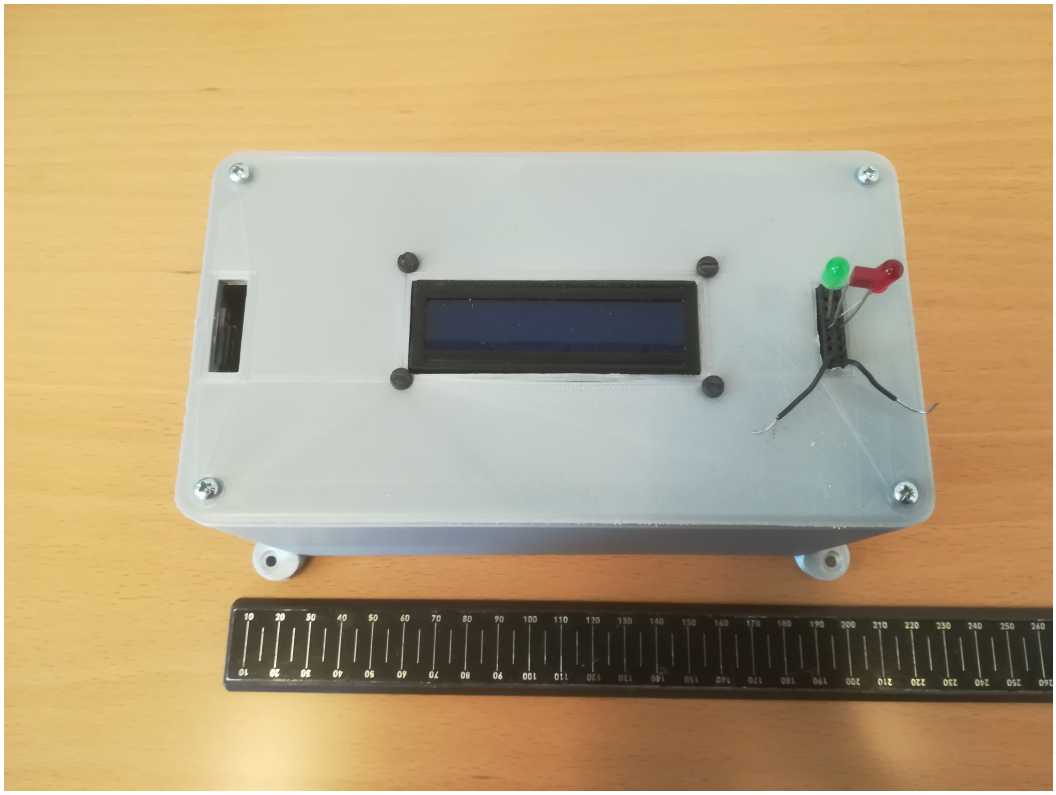


Figure A.7: *A 3D-printet box contains all the electronic from Figure A.6*

.

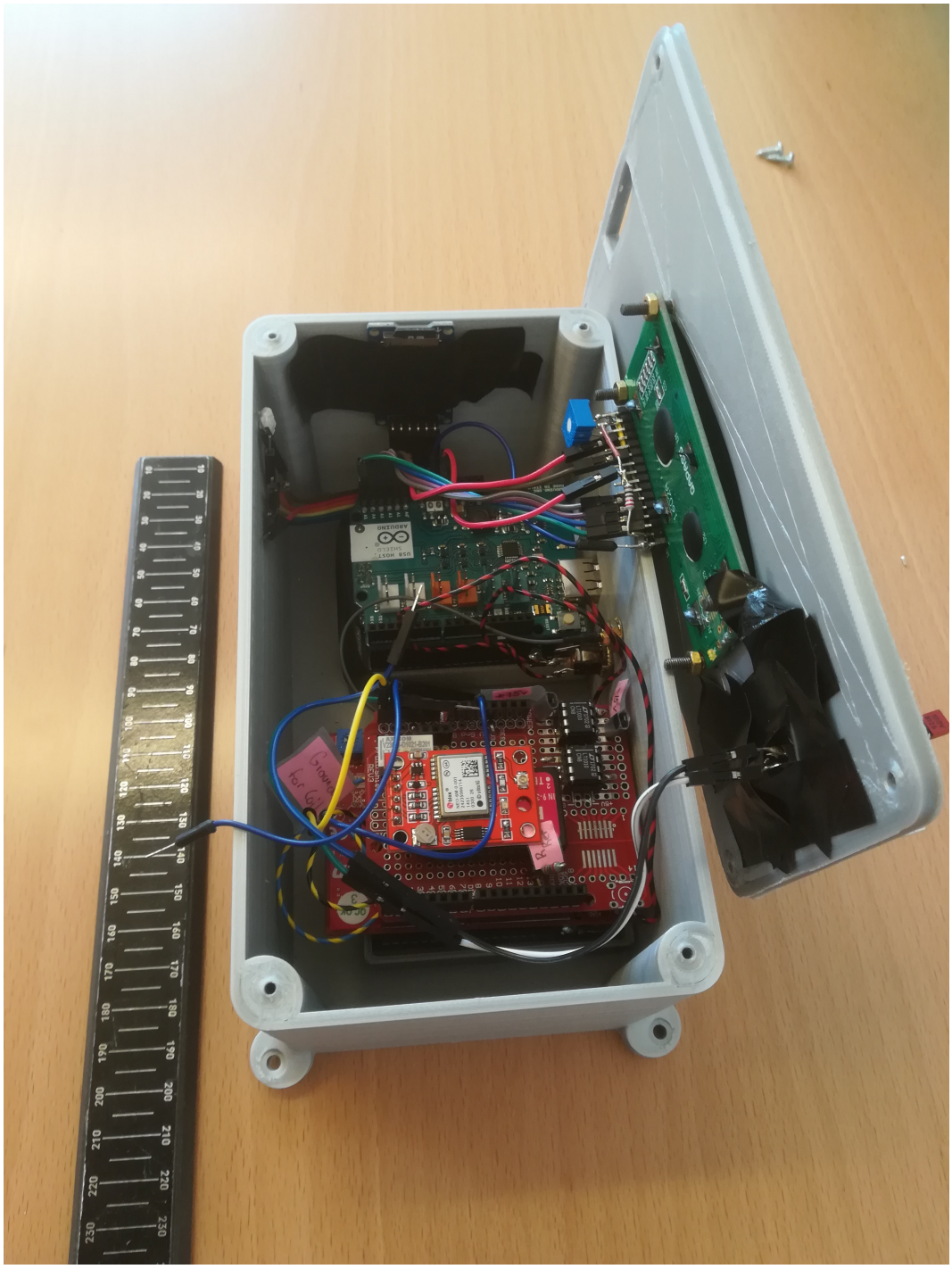


Figure A.8: *Inside the 3D-printet box, are all the elektronik*

.

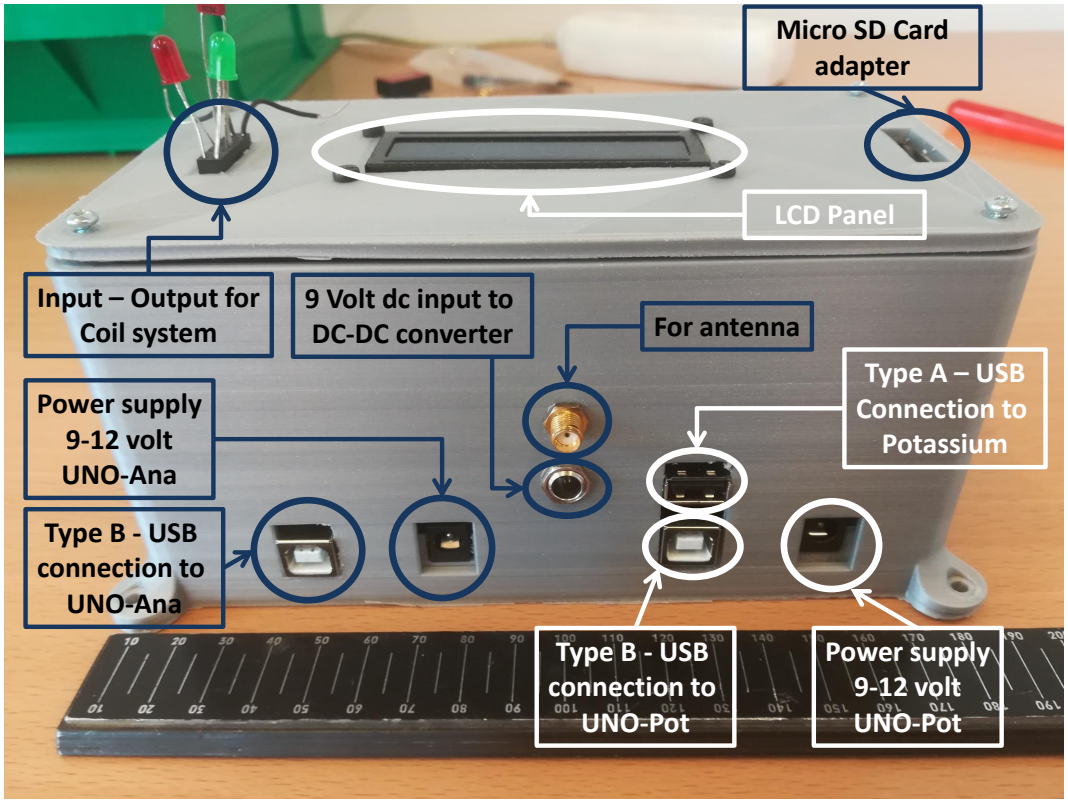


Figure A.9: *From the backside of the 3D-printet box, input and output are available*

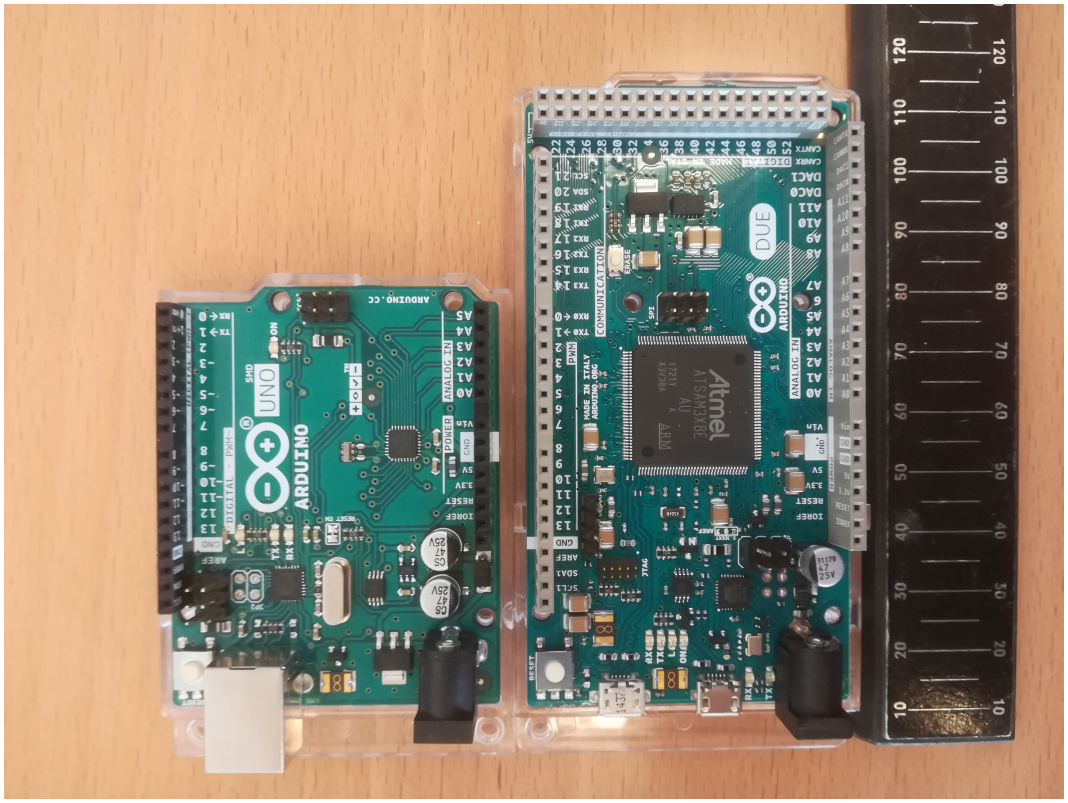


Figure A.10: *Arduino UNO on the left and Arduino DUE to the right*

.

A.2 Assembling

In this thesis the computer system in Figure A.6 was used to collect the data. In this section a description of how to assembly it. If the 3D-printed box is in Figure A.7 is a desired feature to contain the electronic, it can be found on the ftp server:

`ftp://ftp2.space.dtu.dk/pub/ABZ_Magnetometer/A_Computer_system/A.2_Assembling`

Where both the SolidWorks files and the 3D print-frendly STL files are uploaded. The assembling description are divided into Pot-Unit and Ana-Unit.

Pot-Unit:
The Pot-Unit is the simplest configurations of these two, the design is shown in Figure 3.7.

1. Stack the USB Host shield unto Arduino UNO.
2. Compile Arduino code (Listing A.1) into the Arduino UNO.
3. Configured the LCD, use the approach from Arduino Homepage `https://www.arduino.cc/en/Tutorial/HelloWorld`, But with the following pin configuration
   - LCD RS pin to digital pin A0
   - LCD Enable pin to digital pin A1
   - LCD D4 pin to digital pin A2
   - LCD D5 pin to digital pin A3
   - LCD D6 pin to digital pin A4
   - LCD D7 pin to digital pin A5
4. Now its ready to use. Turn on the Potassium magnetometer and plug the USB from Potassium magnetometer into USB Host shield.
5. After the Potassium magnetometer has warmed up the total intensity will be showed on the LCD, and the Potassium magnetometer data will be send serial over Tx/Rx from Arduino UNO.

Ana-Unit
In Figure 3.6 the Ana-Unit is showed. The pin figuration to the Buffer (LT1010), DC-DC and Relay is seen in Figure A.11a, A.11b and A.11c, respectively.

1. The Analog shield is unto Arduino UNO.
2. Compile Arduino code (Listing A.2) into the Arduino UNO.
3. In order to get Potassium magnetometer data, connect:

- Pin Rx from Pot-Micronetroller to A5 on Ana-Micronetroller.
- Pin Tx from Pot-Micronetroller to A4 on Ana-Micronetroller.

4. Connect the two LT1010 bufferer as seen in Figure 3.6:

   - **First LT1010:**
     LT1010 pin 1 (V^+) to DC-DC pin 14 (+15 V).
     LT1010 pin 6 (V^-) to DC-DC pin 11 (-15 V).
     LT1010 pin 8 (Input) to D1 on Analog shield.
     LT1010 pin 3 (Output) to pin 5 on relay.

   - **Second LT1010:**
     LT1010 pin 1 (V^+) to +VADJ (+15 V) on Analog shield.
     LT1010 pin 6 V^- to -VADJ (-15 V) on Analog shield.
     LT1010 pin 8 (Input) from coil system.
     LT1010 pin 3 (Output) to A0 on Analog shield.

5. Connect relay pin 1 to coil system.

6. Connect a 125 Ohm as reference resistance as shown in Figure 3.6.

7. Connect relay pin 8 to GND and relay pin 3 to D3 on Analog shield.

8. Connect GPS as described in
   `https://www.xarg.org/2016/06/neo6mv2-gps-module-with-arduino/`.
   But connect A0 to GPS Rx and A1 to GPS Tx.

9. Connect pin 9 and 16 on DC-DC to GND.

10. Lastly connect the DC-DC to a 9 volt DC input, where pin 2 and 3 form DC-DC is connected to zero, and pin 23 and 23 form DC-DC is connected to plus.

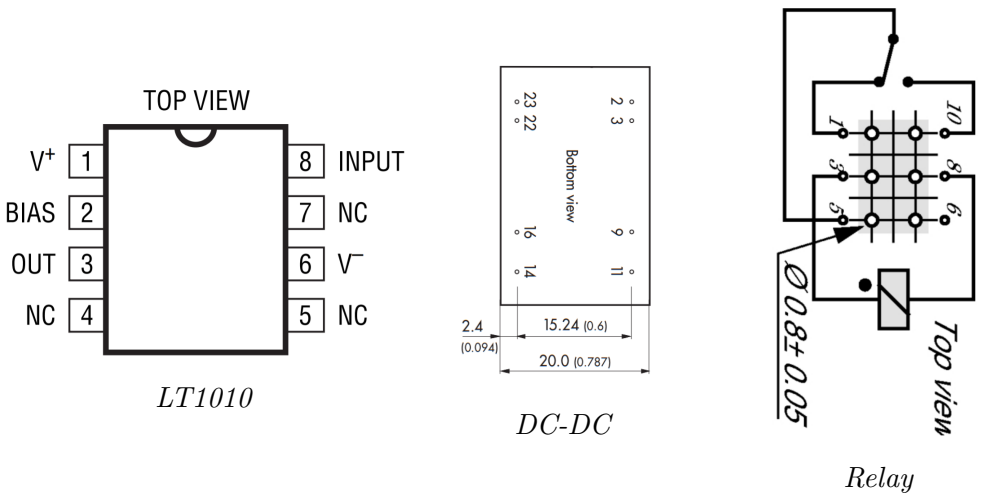


Figure A.11: *Pin figurations*

If the above pin configuration is complete, a similar system as the one showed in Figure A.6 is replicated. Even though the GPS is connected it is not employ in the code Listing A.2. But if Listing A.3 is compiled unto Arduino UNO in the Ana-Unit the GPS will be working. The SD memory are not included in this implementation, the code found in Listing A.4 are working together with the tutorial found here: `http://educ8s.tv/arduino-sd-card-tutorial/`.

A.3 Arduino code

```
////////////////////////////////////////////////////////////////
//              Pot-microcontroller code                      //
// This code is made for the Arduino Uno included in the      //
//               system called UNO-Pot.                       //
////////////////////////////////////////////////////////////////

/* USB support */
#include <usbhub.h>
/* CDC support */
#include <cdcacm.h>
#include <cdcprolific.h>

// Satisfy the IDE.
#ifdef dobogusinclude
#include <spi4teensy3.h>
#endif

#include <SPI.h>

// include the library code:
#include <LiquidCrystal.h>


// initialize the library by associating any needed LCD interface
// pin with the arduino pin number it is connected to
const int rs = A0, en = A1, d4 = A2, d5 = A3, d6 = A4, d7 = A5;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);


/* Setup USB-host shield */
class PLAsyncOper : public CDCAsyncOper {
public:
        uint8_t OnInit(ACM *pacm);
};

/* Read/Print varibels */
const byte numChars = 64;      // How many bit can be read from
                               // Potassium with 36 bit Signal
                               // strength are included
char receivedChars[numChars]; // An array to store the received
    data

uint8_t PLAsyncOper::OnInit(ACM *pacm) {
        uint8_t rcode;

        // Set DTR = 1
```

```
46          rcode = pacm->SetControlLineState(1);
47
48          if(rcode) {
49                  ErrorMessage<uint8_t>(PSTR("SetControlLineState"),
        rcode);
50                  return rcode;
51          }
52
53          LINE_CODING lc;
54          lc.dwDTERate = 115200; //default serial speed of GPS unit
55          lc.bCharFormat = 0;
56          lc.bParityType = 0;
57          lc.bDataBits = 8;
58
59          rcode = pacm->SetLineCoding(&lc);
60
61          if(rcode)
62                  ErrorMessage<uint8_t>(PSTR("SetLineCoding"), rcode
        );
63
64          return rcode;
65  }
66
67  /* More USB setup*/
68  USB Usb;
69  USBHub Hub(&Usb);
70  PLAsyncOper AsyncOper;
71  PL2303 Pl(&Usb, &AsyncOper);
72  uint32_t read_delay;
73  #define READ_DELAY 0
74
75  void setup() {
76    Serial.begin(19200);
77
78    #if !defined(__MIPSEL__)
79      while (!Serial); // Wait for serial port to connect
80    #endif
81
82    if(Usb.Init() == -1)
83       Serial.println("OSCOKIRQ failed to assert");
84
85    // set up the LCD's number of columns and rows:
86    lcd.begin(16, 2);
87    // Print a message to the LCD.
88    lcd.print("ABZ Magnetometer");
89
90
91    delay(200);
92  }
93
94  void loop() {
95
96    char endMarker = '\n';
97    uint8_t rcode;
98    uint8_t buf[64]; //serial buffer equals Max.packet size of bulk-
       IN endpoint
99    uint16_t rcvd = 64;
100       Usb.Task();
```

```
101
102         if(Pl.isReady()) {
103                 /* reading the GPS */
104               if((int32_t)((uint32_t)millis() - read_delay) >= 0L) {
105                     read_delay += READ_DELAY;
106                     rcode = Pl.RcvData(&rcvd, buf);
107                     if(rcode && rcode != hrNAK)
108                           ErrorMessage<uint8_t>(PSTR("Ret"), rcode);
109                     if(rcvd) { //more than zero bytes received
110                           for(uint16_t i = 0; i < rcvd; i++) {
111                                 recvWithEndMarker((char)buf[i]);
112                           }//for( uint16_t i=0; i < rcvd; i++...
113                     }//if( rcvd
114               }//if( read_delay > millis()...
115     }//if( Usb.getUsbTaskState() == USB_STATE_RUNNING..
116 }
117
118
119 void recvWithEndMarker(char rc) {
120  static byte ndx = 0;
121  char endMarker = '\n';
122  char F[17];
123
124  if (rc != endMarker) {
125  receivedChars[ndx] = rc;
126  ndx++;
127  if (ndx >= numChars) {
128  ndx = numChars - 1;
129  }
130  }
131  else {
132  receivedChars[ndx] = '\0'; // terminate the string
133  ndx = 0;
134  Serial.println(receivedChars);
135
136  memcpy(F, receivedChars + 10 , 11 );
137  lcd.setCursor(0, 1);
138  lcd.print("F = "); lcd.print(F);
139  }
140 }
```

Listing A.1: *Arduino code for Pot-Microcontroller*

```
////////////////////////////////////////////////////////////
//              Ana-microcontroller code                  //
// This code is made for the Arduino Uno included in the  //
//               system called UNO-Ana.                   //
//     Note: that GPS and SD Card are comment out.        //
////////////////////////////////////////////////////////////
/* GPS support */
#include <TinyGPS.h>
TinyGPS gps;

// Satisfy the IDE.
#ifdef dobogusinclude
#include <spi4teensy3.h>
#endif

#include <SPI.h>  //required for ChipKIT and Arduino DUE
#include <analogShield.h>   //Include to use analog shield.

/* Serial communications*/
#include <SoftwareSerial.h>
SoftwareSerial po(A4, A3); // software serial #1: RX = digital pin A4, TX = digital pin A3
SoftwareSerial ss(A0, A1); // software serial #1: RX = digital pin A1, TX = digital pin A0

/* GPS varibles */
static void smartdelay(unsigned long ms);
static void print_float(float val, float invalid, int len, int prec);
static void print_int(unsigned long val, unsigned long invalid, int len);
static void print_date(TinyGPS &gps);
byte month, day, hour, minute, second, hundredths;
unsigned long age;
int year;

/* Current control varibels*/
unsigned int cycels = 20;              //
   <-------------------------   Current cycels  !!
int indexSample = 0;
unsigned int coil_output = 0, coil_input = 0, funk = 0, k = 0;
// "-5v" = 0 ,"Zero" = 32768, "5v" = 65535
unsigned int sarray[4] = {32768, 65535, 32768, 0};
//unsigned int sarray[4] = {32768, 49151, 32768, 16384};
// unsigned int sarray[4] = {65535, 60000, 65535, 60000};
   //49151
char Lo[2];


/* Read/Print varibels */
const byte numChars = 22;      // How many bit can be read from Potassium
                               // With 36 bit Signal strength are included
                               // for pot numChars = 36;
                               // for OverH numChars = 22;
char receivedChars[numChars]; // an array to store the received
```

```
    data
boolean newData = false;

/*  Calibrate varible */
float alpha;
unsigned int t1 = 0;
float t;

//static FILE mystdout;

void setup() {
  Serial.begin(115200);
  while (!Serial) {}// wait for serial port to connect.

  ss.begin(4800);

  // Start each software serial port
  po.begin(19200);  // By "try and error" this is the optimal baud
    rate
                    // Fast enough to read 20 Hz potassium,
                    // but slow enough for Uno_Analog to connect.
  analog.write(0,0);
  analog.write(3, 65535);  // Turn on relay - 'D3' - On
  analog.write(1, 32768);  // Make a start signal for ES - 'D1'
  coil_input = sarray[k];      //

  while(t1 < 200){
    t = analog.read(1) - analog.read(0);
    t = 7500/t;
    alpha = alpha + t;
    t1++;
  }

  alpha = alpha/t1;

  delay(200);
}

/*******************         Main loop
    **********************/
void loop() {
  /*   Sequence of "Equal steps"     */
  if(cycels == indexSample){ // for every "cycels", update output;

      /*  Turn on relay when F_earth are measured */
      if(sarray[k] == 32768){
        analog.write(3, 0); } //write current_value out on port
    labeled 'D0'
      else{
        analog.write(3, 65535); } //write current_value out on
    port labeled 'D0'


      analog.write(1, sarray[k]);  // write current_value out on
    port labeled 'D0'
      coil_input = sarray[k];

      indexSample = 0;
```

```
102         k++;
103         if (k == 4) k = 0;
104         //newData == true;
105       }
106
107       /*  Measure input/output to Lee-whiting coil system*/
108       //coil_input = analog.read(2); //read in on port labeled 'A2'
109       coil_output = analog.read(3); //read in on port labeled 'A3'
110
111       /* Read data from Potassium magnetometer */
112       //po.listen();
113       recvWithEndMarker();
114
115       /*  Print all data, when Potassium magnetometer are ready*/
116       showNewData();
117
118   }
119
120   /*******  Read from Potassium - Function *****/
121
122   void recvWithEndMarker() {
123    static byte ndx = 0;
124    char endMarker = '\n';
125    char rc;
126
127    while (po.available() > 0 && newData == false) {
128    rc = po.read();
129
130    if (rc != endMarker) {
131    receivedChars[ndx] = rc;
132    ndx++;
133    if (ndx >= numChars) {
134    ndx = numChars - 1;
135    }
136    }
137    else {
138    receivedChars[ndx] = '\0'; // terminate the string
139    ndx = 0;
140    newData = true;
141    }
142    }
143   }
144
145   /*******  Print Function (And some GPS reading)  ******/
146   void showNewData() {
147
148    if (newData == true) {
149
150    Serial.print(receivedChars); Serial.print(" ");
151    Serial.print(coil_input);    Serial.print(" ");
152    Serial.print(coil_output);   Serial.print(" ");
153
154    Serial.println();
155
156    indexSample++;
157    newData = false;
158    //delay(200);
159    }
```

```
160 }
161
162 void starttime()  {
163   float flat, flon;
164   unsigned long age, date, time, chars = 0;
165   unsigned short sentences = 0, failed = 0;
166
167   gps.f_get_position(&flat, &flon, &age);
168   print_float(flat, TinyGPS::GPS_INVALID_F_ANGLE, 10, 6);
169   print_float(flon, TinyGPS::GPS_INVALID_F_ANGLE, 11, 6);
170   print_date(gps);
171
172   Serial.println();
173   Serial.println();
174
175 }
176
177 /*   Tjeck for Luck   */
178
179 /*void Lock(){
180     recvWithEndMarker();
181     memcpy(Lo, receivedChars + 22, 1);
182
183     while(strcmp(Lo, "0") != 0){  // Check Potassium is locked
184       delay(1000);
185       newData == false;
186       recvWithEndMarker();
187       memcpy(Lo, receivedChars + 22, 1);
188       // Serial.println("  Potassium magnetometer is not Locked
    yet  ");
189     }
190 }*/
191
192
193 static void smartdelay(unsigned long ms)
194 {
195   unsigned long start = millis();
196   do
197   {
198     while (ss.available())
199       gps.encode(ss.read());
200   } while (millis() - start < ms);
201 }
202
203
204 static void print_float(float val, float invalid, int len, int
    prec)
205 {
206   if (val == invalid)
207   {
208     while (len-- > 1)
209       Serial.print('*');
210       Serial.print(' ');
211   }
212   else
213   {
214     Serial.print(val, prec);
215     int vi = abs((int)val);
```

```
216     int flen = prec + (val < 0.0 ? 2 : 1); // . and -
217     flen += vi >= 1000 ? 4 : vi >= 100 ? 3 : vi >= 10 ? 2 : 1;
218     for (int i=flen; i<len; ++i)
219       Serial.print(' ');
220   }
221   smartdelay(0);
222 }
223 
224 static void print_int(unsigned long val, unsigned long invalid,
      int len)
225 {
226   char sz[32];
227   if (val == invalid)
228     strcpy(sz, "*******");
229   else
230     sprintf(sz, "%ld", val);
231   sz[len] = 0;
232   for (int i=strlen(sz); i<len; ++i)
233     sz[i] = ' ';
234   if (len > 0)
235     sz[len-1] = ' ';
236   Serial.print(sz);
237  smartdelay(0);
238 }
239 
240 static void print_date(TinyGPS &gps)
241 {
242   int year;
243   byte month, day, hour, minute, second, hundredths;
244   unsigned long age;
245   gps.crack_datetime(&year, &month, &day, &hour, &minute, &second,
      &hundredths, &age);
246   if (age == TinyGPS::GPS_INVALID_AGE)
247     Serial.print("********** ******** ");
248   else
249   {
250     char sz[32];
251     sprintf(sz, "%02d/%02d/%02d %02d:%02d:%02d ",
252         month, day, year, hour, minute, second);
253     Serial.print(sz);
254   }
255   print_int(age, TinyGPS::GPS_INVALID_AGE, 5);
256   smartdelay(0);
257 }
```

Listing A.2: *Arduino code for Ana-Microcontroller*

```
////////////////////////////////////////////////////////////
//                    GPS test code                       //
//  This script clearly demonstrate how to use the GPS.   //
//  It can run on the arduino UNO in the UNO-Ana system   //
//             without change the setup                   //
//                Cedit to Mikal Hart                     //
////////////////////////////////////////////////////////////

#include <SoftwareSerial.h>

#include <TinyGPS.h>

/* This sample code demonstrates the normal use of a TinyGPS
   object.
   It requires the use of SoftwareSerial, and assumes that you
   have a
   4800-baud serial GPS device hooked up on pins 4(rx) and 3(tx).
*/

TinyGPS gps;
SoftwareSerial ss(A0, A1);

static void smartdelay(unsigned long ms);
static void print_float(float val, float invalid, int len, int
   prec);
static void print_int(unsigned long val, unsigned long invalid,
   int len);
static void print_date(TinyGPS &gps);
static void print_str(const char *str, int len);

void setup()
{
  Serial.begin(9600);

  Serial.print("Testing TinyGPS library v. "); Serial.println(
   TinyGPS::library_version());
  Serial.println("by Mikal Hart");
  Serial.println();
  Serial.println("Sats HDOP Latitude  Longitude  Fix  Date
   Time     Date Alt    Course Speed Card  Distance Course Card
   Chars Sentences Checksum");
  Serial.println("           (deg)     (deg)      Age
          Age  (m)    --- from GPS ----  ---- to London  ----  RX
       RX        Fail");
  Serial.println("--------------------------------------------
   ------------------------------------------
   ------------------------------------  ");

  ss.begin(9600);
}

void loop()
{
  float flat, flon;
  unsigned long age, date, time, chars = 0;
  unsigned short sentences = 0, failed = 0;
  static const double LONDON_LAT = 51.508131, LONDON_LON =
```

```
    -0.128002;
47
48  print_int(gps.satellites(), TinyGPS::GPS_INVALID_SATELLITES, 5);
49  print_int(gps.hdop(), TinyGPS::GPS_INVALID_HDOP, 5);
50  gps.f_get_position(&flat, &flon, &age);
51  print_float(flat, TinyGPS::GPS_INVALID_F_ANGLE, 10, 6);
52  print_float(flon, TinyGPS::GPS_INVALID_F_ANGLE, 11, 6);
53  print_int(age, TinyGPS::GPS_INVALID_AGE, 5);
54  print_date(gps);
55  print_float(gps.f_altitude(), TinyGPS::GPS_INVALID_F_ALTITUDE,
      7, 2);
56  print_float(gps.f_course(), TinyGPS::GPS_INVALID_F_ANGLE, 7, 2);
57  print_float(gps.f_speed_kmph(), TinyGPS::GPS_INVALID_F_SPEED, 6,
      2);
58  print_str(gps.f_course() == TinyGPS::GPS_INVALID_F_ANGLE ? "***
      " : TinyGPS::cardinal(gps.f_course()), 6);
59  print_int(flat == TinyGPS::GPS_INVALID_F_ANGLE ? 0xFFFFFFFF : (
      unsigned long)TinyGPS::distance_between(flat, flon, LONDON_LAT,
      LONDON_LON) / 1000, 0xFFFFFFFF, 9);
60  print_float(flat == TinyGPS::GPS_INVALID_F_ANGLE ? TinyGPS::
      GPS_INVALID_F_ANGLE : TinyGPS::course_to(flat, flon, LONDON_LAT
      , LONDON_LON), TinyGPS::GPS_INVALID_F_ANGLE, 7, 2);
61  print_str(flat == TinyGPS::GPS_INVALID_F_ANGLE ? "*** " :
      TinyGPS::cardinal(TinyGPS::course_to(flat, flon, LONDON_LAT,
      LONDON_LON)), 6);
62
63  gps.stats(&chars, &sentences, &failed);
64  print_int(chars, 0xFFFFFFFF, 6);
65  print_int(sentences, 0xFFFFFFFF, 10);
66  print_int(failed, 0xFFFFFFFF, 9);
67  Serial.println();
68
69  smartdelay(1000);
70}
71
72static void smartdelay(unsigned long ms)
73{
74  unsigned long start = millis();
75  do
76  {
77    while (ss.available())
78      gps.encode(ss.read());
79  } while (millis() - start < ms);
80}
81
82static void print_float(float val, float invalid, int len, int
    prec)
83{
84  if (val == invalid)
85  {
86    while (len-- > 1)
87      Serial.print('*');
88    Serial.print(' ');
89  }
90  else
91  {
92    Serial.print(val, prec);
93    int vi = abs((int)val);
```

```
94      int flen = prec + (val < 0.0 ? 2 : 1); // . and -
95      flen += vi >= 1000 ? 4 : vi >= 100 ? 3 : vi >= 10 ? 2 : 1;
96      for (int i=flen; i<len; ++i)
97        Serial.print(' ');
98    }
99    smartdelay(0);
100 }
101
102 static void print_int(unsigned long val, unsigned long invalid,
      int len)
103 {
104   char sz[32];
105   if (val == invalid)
106     strcpy(sz, "*******");
107   else
108     sprintf(sz, "%ld", val);
109   sz[len] = 0;
110   for (int i=strlen(sz); i<len; ++i)
111     sz[i] = ' ';
112   if (len > 0)
113     sz[len-1] = ' ';
114   Serial.print(sz);
115   smartdelay(0);
116 }
117
118 static void print_date(TinyGPS &gps)
119 {
120   int year;
121   byte month, day, hour, minute, second, hundredths;
122   unsigned long age;
123   gps.crack_datetime(&year, &month, &day, &hour, &minute, &second,
      &hundredths, &age);
124   if (age == TinyGPS::GPS_INVALID_AGE)
125     Serial.print("********** ******** ");
126   else
127   {
128     char sz[32];
129     sprintf(sz, "%02d/%02d/%02d %02d:%02d:%02d ",
130         month, day, year, hour, minute, second);
131     Serial.print(sz);
132   }
133   print_int(age, TinyGPS::GPS_INVALID_AGE, 5);
134   smartdelay(0);
135 }
136
137 static void print_str(const char *str, int len)
138 {
139   int slen = strlen(str);
140   for (int i=0; i<len; ++i)
141     Serial.print(i<slen ? str[i] : ' ');
142   smartdelay(0);
143 }
```

Listing A.3: *GPS test*

```
////////////////////////////////////////////////////////////////
//                    SD memory test code                     //
//  This script demonstrate how to setup and save data        //
//     on the Micro SD Card through the SD Adapter            //
//                                                            //
//   It can run on the arduino UNO in the UNO-Ana system      //
//                    without change the setup                //
////////////////////////////////////////////////////////////////

#include <SD.h>
#include <SPI.h>

const int CS_PIN = 10;

File dataFile;

void setup()
{

 Serial.begin(9600);


  //setup SD card
   Serial.print("Initializing SD card...");

  // see if the SD card is present and can be initialized:
  if (!SD.begin(CS_PIN)) {
    Serial.println("Card failed, or not present");
    // don't do anything more:
    return;
  }
  Serial.println(" Card initialized.");

  // write down the date (year / month / day
  // prints only the start, so if the logger runs
  // for sevenal days you only findt the start back at the begin.
    dataFile = SD.open("datalog.txt", FILE_WRITE);
    dataFile.println("Start logging now: ");
    dataFile.close();
}

void loop()
{

    //open file to log data in.
   dataFile = SD.open("datalog.txt", FILE_WRITE);

  // if the file is available, write to it:
  // log the temperature and time.
  if (dataFile) {
    dataFile.println(millis(),DEC);

    dataFile.close();
    // print to the serial port too:
    Serial.print("data stored    -   ");
    Serial.println(dataFile);
  }
```

```
58    // if the file isn't open, pop up an error:
59    else {
60      Serial.println("error opening datalog.txt   -   ");
61      Serial.println(dataFile);
62      dataFile.close();
63    }
64
65    delay(1000);
66  }
```

Listing A.4: *SD memory*

B

Estimation of model parameters

B.1 Clean ABZ data

```
clear all
close all

% Inetial information
AA = load('Data/Three_Level_Current_20180610.txt');
idx_start = find(AA(:,1) == 806);
AA = AA(idx_start:end, :);
start_time = 16*60*60 + 56*60 + 0;
Hz = 20;

%% Remove bad data points

AA_time = [1:length(AA)]/Hz + start_time;
idx = find(AA(:,3) == 1); % tjek for lock
%         time,          Field,       coil in   , Coil out
%         1              2            3           4
ABZ01 = [AA_time(idx)',AA(idx,2),  AA(idx,8), AA(idx,9)];

indi= find(abs(diff(ABZ01(:,2))) > 0.5);
indi02 = indi+1;

ABZ01([indi; indi02],:) = [];


figure()
hold on
plot(ABZ01(:,1), ABZ01(:,2), '.')
ax = gca;
ax.YAxis.TickLabelFormat = '%,.2f';
ax.YAxis.Exponent = 0;

% figure()
% hold on
% plot(ABZ01(idx_e,1), ABZ01(idx_e,2), '.')
% ax = gca;
% ax.YAxis.TickLabelFormat = '%,.2f';
% ax.YAxis.Exponent = 0;


%% Interpolated ABZ data to FGM data

fileID = fopen('Data/BFE6_20180610_9999.sec');
```

```
frewind(fileID); % Move file position indicator to beginning of
    open file
    FGE_data = textscan(fileID,['%s %f %f %f %f %f']); %f %f %f %f
    %f %f %f');
fclose(fileID);

FGE_data{1,1};
out = datevec(FGE_data{1,1});
out = out(:,4:end);

t = out(:,1)*60*60 + out(:,2)*60 + out(:,3);
t(end) = 24*60*60;

H_baseline = 17149.5; % nT
Z_baseline = 46960.7; % nT
% H_baseline = 17211.2; % nT
% Z_baseline = 47128; % nT


% F_earth
Fe = sqrt((H_baseline + FGE_data{1,2}).^2 + FGE_data{1,3}.^2 + (
    Z_baseline + FGE_data{1,4}).^2 );
He = sqrt((H_baseline + FGE_data{1,2}).^2 + FGE_data{1,3}.^2);
Ze = Z_baseline + FGE_data{1,4};
Xe = H_baseline + FGE_data{1,2} ;
Ye = FGE_data{1,3};

t_int = find(t >= ABZ01(1,1) & t <= ABZ01(end,1));
ABZ = t(t_int);
ABZ(:,2) = interp1(ABZ01(:,1),ABZ01(:,2),ABZ(:,1),'nearest','
    extrap'); % F
ABZ(:,3) = interp1(ABZ01(:,1),ABZ01(:,3),ABZ(:,1),'nearest','
    extrap'); % Coil input
ABZ(:,4) = interp1(ABZ01(:,1),ABZ01(:,4),ABZ(:,1),'nearest','
    extrap');

ABZ(:,5) = Fe(t_int);
ABZ(:,6) = Xe(t_int);
ABZ(:,7) = Ye(t_int);
ABZ(:,8) = Ze(t_int);


%%  alpha and delta have been independently changes
%   Here the data are seperated into parts.

ac(1,1) =  start_time;
ac(1,2) =  (11*60) - (8*60 + 6) + start_time;
ac(2,1) =  (11*60 + 30) - (8*60 + 6) + start_time;

ii = 2;
for i = 15:5:90
    ac(ii,2) =  (i*60) - (8*60 + 6) + start_time;
    ac(ii+1,1) =  (i*60 + 30) - (8*60 + 6) + start_time;
    ii = ii + 1;
end
ac(ii,2) =  ABZ(end,1);

N_turn = 1/180;
```

```
ac(:,3) = [N_turn; 2*N_turn; 3*N_turn; 4*N_turn; repmat(5*N_turn
    ,[9,1]);...
    4*N_turn; 3*N_turn; 2*N_turn; 1*N_turn; 0];

R_turn = pi/4;
ac(:,4) = [zeros(5,1); pi/4; pi/2 ; pi*3/4; pi; pi*3/4; pi/2 ; pi
    /4; zeros(6,1)];

ABZ(:, 9:10) = nan; %zeros(length(ABZ01),1);
for i = 1:length(ac)
    idx_ac = find(ac(i,1) <= ABZ(:,1) & ac(i,2) >= ABZ(:,1));
    ABZ(idx_ac, [9:10]) = ones(length(idx_ac),1)*ac(i,3:4);
end

out = find(isnan(ABZ(:,9)));


figure()
hax=axes;
hold on
plot(ABZ(:,1), ABZ(:,2),'.')
line([ac(:,1) ac(:,1)],get(hax,'YLim'),'Color',[1 0 0])
line([ac(:,2) ac(:,2)],get(hax,'YLim'),'Color',[1 0 0])
plot(ABZ(out,1), ABZ(out,2),'o')

ABZ(out,:) = [];

figure()
hax=axes;
hold on
plot(ABZ(:,1), ABZ(:,2),'.')

%% For free Alpha and Delta

Free_A = zeros(length(ABZ),6);
Free_B = zeros(length(ABZ),5);

Free_A(find(ABZ(:,9) == N_turn),1) = 1;
Free_A(find(ABZ(:,9) == 2*N_turn),2) = 1;
Free_A(find(ABZ(:,9) == 3*N_turn),3) = 1;
Free_A(find(ABZ(:,9) == 4*N_turn),4) = 1;
Free_A(find(ABZ(:,9) == 5*N_turn),5) = 1;
Free_A(find(ABZ(:,9) == 0),6) = 1;

Free_B(find(ABZ(:,10) == 0),1) = 1;
Free_B(find(ABZ(:,10) == pi/4),2) = 1;
Free_B(find(ABZ(:,10) == pi/2),3) = 1;
Free_B(find(ABZ(:,10) == pi*3/4),4) = 1;
Free_B(find(ABZ(:,10) == pi),5) = 1;


ABZ = [ABZ, Free_A, Free_B];
%% Change X, Y, Z due to location.

idx_p = find(ABZ(:,2) > 53000);
idx_e = find(ABZ(:,2) > 50000 & ABZ(:,2) < 51000);
idx_m = find(ABZ(:,2) < 47000);

```

```
figure()
hold on
plot(ABZ(:,1), ABZ(:,5))
plot(ABZ(idx_e,1), ABZ(idx_e,2))
legend('FGE','ABZ')


hist_p = ABZ(idx_p,2) - ABZ(idx_p,5);
hist_e = ABZ(idx_e,2) - ABZ(idx_e,5);
hist_m = ABZ(idx_m,2) - ABZ(idx_m,5);


figure()
histogram(hist_e - mean(hist_e))

% Inc = degtorad(77.7131);
Inc = degtorad(69.9377);
Dec = degtorad(2 + 35/60 + 69/3600);
% Dec = degtorad(3.5382);

F_change = mean(hist_e);


di = 1;
while abs(di) > 0.001
Z_change =  F_change*sin(Inc);
H_change = Z_change/tan(Inc);

X_change = H_change*cos(Dec);
Y_change = H_change*sin(Dec);

X = ABZ(:,6) + X_change;
Y = ABZ(:,7) + Y_change;
Z = ABZ(:,8) + Z_change;

di = mean(ABZ(:,5) + mean(hist_e) - sqrt(X.^2 + Y.^2 + Z.^2));
F_change = mean(hist_e) + di;
end


figure()
hold on
plot(ABZ(:,1), sqrt(X.^2 + Y.^2 + Z.^2))
plot(ABZ(:,1), ABZ(:,5) + F_change,'o')
legend('X, Y, and Z', 'F')
ax = gca;
ax.YAxis.TickLabelFormat = '%,.2f';
ax.YAxis.Exponent = 0;

ABZ(:,5) = ABZ(:,5) + F_change;
ABZ(:,6) = ABZ(:,6) + X_change;
ABZ(:,7) = ABZ(:,7) + Y_change;
ABZ(:,8) = ABZ(:,8) + Z_change;

figure()
hold on
plot(ABZ(:,1), ABZ(:,5))
plot(ABZ(idx_e,1), ABZ(idx_e,2))
```

```
209 legend('FGE','ABZ')
210
211 %% Current
212 gamma =  1.5259e-04;  % 0.000155;  %  = (10 volt)/(2^16 Bit)   %
213 OffSet = mean(ABZ(idx_e,3).*gamma);
214 ABZ(:,3) = (ABZ(:,3).*gamma - OffSet)/(115*1e-3);
215 ABZ(:,4) = (ABZ(:,4).*gamma - OffSet)/(115*1e-3);
216 ABZ(idx_e,[3,4]) = 0;
217
218
219 %% save
220 save('Data/Prior_ABZ_dataset.mat','ABZ')
```

B.2 Equal step method

```
1  function [Zp, Z0, H0, Q] = Equal_pm(Fp, Fm, F0, I)
2
3  Zpm = @(Fp, Fm, F0)(sqrt((Fp.^2 + Fm.^2)./2 - F0.^2) );
4  Z0_pm = @(Fp, Fm, Zpm)(( Fp.^2 - Fm.^2 )./(4.*Zpm) );
5  H0_pm = @(F0, Z0)(sqrt(abs(F0.^2 - Z0.^2))  );
6
7  Zp = Zpm(Fp, Fm, F0);
8  Z0 = Z0_pm(Fp, Fm, Zp);
9  H0 = H0_pm(F0, Z0);
10 Q = Zp./I;
11 end
```

B.3 Numerical optimization - Pre-installed Matlab function

```
1  % Tobias Bjerg - Numerical optimization 2018-06/10.
2
3  clear all
4  close all
5
6  load Data/Prior_ABZ_dataset.mat;
7  Adda = ABZ; clear ABZ;
8
9  idx_e = find(Adda(:,3) == 0);
10 idx_p = find(Adda(:,3) > 0);
11 idx_m = find(Adda(:,3) < 0);
12
13 figure()
14 hold on
15 plot(Adda(idx_e,1), Adda(idx_e,2),'.')
16 plot(Adda(idx_e,1), sqrt(Adda(idx_e,6).^2 + ...
17                          Adda(idx_e,7).^2 + ...
18                          Adda(idx_e,8).^2)  ,'o')
19
20
21 %% %%%%%%%%%%%%%%%%%%  Varing Alpha  %%%%%%%%%%%%%%%%%%%%%%%
22 options = optimoptions('lsqcurvefit','Display','off');
```

```
options.Algorithm = 'levenberg-marquardt'; lb =[]; ub = [];
options.TolX = 1e-20;

Fun = @(m, x) (sqrt( (x(:,1) + (x(:,4) + x(:,7).*m(4)).*m(1).*
    sin(m(2) + x(:,5)).*cos(m(3) + x(:,6))).^2 + ...
                        (x(:,2) + (x(:,4) + x(:,7).*m(4)).*m(1).*
    sin(m(2) + x(:,5)).*sin(m(3) + x(:,6))).^2 + ...
                        (x(:,3) + (x(:,4) + x(:,7).*m(4)).*m(1).*
    cos(m(2) + x(:,5))).^2    ));

Q = zeros(length(Adda),2);
Q(idx_p,1) = 1;
Q(idx_m,2) = 1;

% x = [X, Y, Z, I, ak, dk]
xdata = [Adda(:,[6:8]), Adda(:,[4,9,10]),Q ];
ydata = Adda(:,2);

m0 = [-130; 0; 0; 0];
% load Data/m0;
tic
[x_alpha,resnorm,residual,exitflag,output] = ...
    lsqcurvefit(Fun,m0, xdata , ydata, lb,ub,options);
toc
m0 = x_alpha;
save('Data/m0.mat','m0');


x_alpha
est = Fun(x_alpha, xdata);

figure()
hold on
plot(Adda(:,1), est,'o')
plot(Adda(:,1), Adda(:,2),'.')
legend('Estimation','Data','Location','Best')
ylabel('[nT]')
ax = gca;
ax.YAxis.TickLabelFormat = '%,.2f';
ax.YAxis.Exponent = 0;
xlim([Adda(1,1), Adda(end,1)])
grid on


figure()
subplot(3,1,1)
hold on
plot(Adda(:,1), est,'o')
plot(Adda(:,1), Adda(:,2),'.')
legend('Estimation','Data','Location','Best')
ylabel('[nT]')
ax = gca;
ax.YAxis.TickLabelFormat = '%,.2f';
ax.YAxis.Exponent = 0;
ylim([54400, 54800])
xlim([Adda(1,1), Adda(end,1)])
title('Prediction - Plus current')
grid on
```

```
subplot(3,1,2)
hold on
plot(Adda(:,1), est,'o')
plot(Adda(:,1), Adda(:,2),'.')
legend('Estimation','Data','Location','Best')
title('Prediction - Zero current')
ylabel('[nT]')
ax = gca;
ax.YAxis.TickLabelFormat = '%,.2f';
ax.YAxis.Exponent = 0;
ylim([50190, 50215])
xlim([Adda(1,1), Adda(end,1)])
grid on
subplot(3,1,3)
hold on
plot(Adda(:,1), est,'o')
plot(Adda(:,1), Adda(:,2),'.')
legend('Estimation','Data','Location','Best')
ylabel('[nT]')
ax = gca;
ax.YAxis.TickLabelFormat = '%,.2f';
ax.YAxis.Exponent = 0;
ylim([45300, 45800])
xlim([Adda(1,1), Adda(end,1)])
title('Prediction - Minus current')
xlabel('Time since midnight [s]')
ylabel('[nT]')
grid on
% print('../../LaTeX/fig/Calibration_Prediction02','-depsc')

%%

figure()
subplot(3,1,1)
histogram(est(idx_e) - Adda(idx_e,2))
legend('No current','Location','best')
title('Residuals between Prediction and ABZ data')
subplot(3,1,2)
histogram(est(idx_p) - Adda(idx_p,2) )
legend('Positive current','Location','best')
subplot(3,1,3)
histogram(est(idx_m) - Adda(idx_m,2) )
legend('Negative current','Location','best')


%% %%%%%%%%%%% save S, alpha, delta %%%%%%%%%%%%
% save('Sab','x_alpha');
% save('Ramp/Sab','x_alpha');
```

B.4 Numerical optimization - Homemade

```
clear all
close all

load ../Data/Prior_ABZ_dataset;

```

```
%%

idx_e = find(ABZ(:,3) == 0);
idx_p = find(ABZ(:,3) > 0);
idx_m = find(ABZ(:,3) < 0);

Q = zeros(length(ABZ),2);
Q(idx_p,1) = 1;
Q(idx_m,2) = 1;

% Select data for the function
d = [ABZ(:,[6:8]), ABZ(:,[4,9,10]),Q ];

Fun = @(x) (sqrt( (d(:,1) + (d(:,4) + d(:,7).*x(4)).*x(1).*sin(x(2) + d(:,5)).*cos(x(3) + d(:,6))).^2 + ...
                  (d(:,2) + (d(:,4) + d(:,7).*x(4)).*x(1).*sin(x(2) + d(:,5)).*sin(x(3) + d(:,6))).^2 + ...
                  (d(:,3) + (d(:,4) + d(:,7).*x(4)).*x(1).*cos(x(2) + d(:,5))).^2    ));




%% %%%%%%%%%%   Model Estimation  %%%%%%%%%%%%
x0 = [-130; 0; 0; 0];
tic
[m] = Linearsation(Fun, x0, ABZ(:,2), d);
toc
```

```
function [m, J, r, Wm, W, L] = Linearsation(Fun, m_0, F_n, d)

m = m_0;

% l = l(1:length(m_0));
% u = u(1:length(m_0));

relative_change = 1; i = 1; e = 1e-4; % don't change
lambda = 0;

while relative_change > 0.0001

    [J] = Jacobion(d, m);
    r = F_n - Fun(m);

    delta_m = pinv(J'*J)*J'*r;
    m = m + delta_m;

    relative_change = 100*norm(delta_m)/norm(m)

    i = i + 1;

    if i > 50
        disp(' Fail  !!!  max iterations ')
        m = m_0;
        relative_change = 0.00001;
    end
end
fprintf('Iterations : %i \n',i)
```

```

end
```

```
function [J] = Jacobion(d, m)
% I_n are current
% m are model parameters:

S = m(1);
alpha = m(2);
delta = m(3);
Io = m(4);

% The Jacobian has been calculated in Maple.
for i = 1:length(d(:,1));
    X = d(i,1);
    Y = d(i,2);
    Z = d(i,3);
    In = d(i,4);
    step1 = d(i,5);
    step2 = d(i,6);
    p = d(i,7);


    J(i,1) = ((2*(X+(Io*p+In)*S*sin(alpha+step1)*cos(delta+step2))
    )*(Io*p+In)*sin(alpha+step1)*cos(delta+step2)...
        +(2*(Y+(Io*p+In)*S*sin(alpha+step1)*sin(delta+step2)))*(Io
    *p+In)*sin(alpha+step1)*sin(delta+step2)+...
        (2*(Z+(Io*p+In)*S*cos(alpha+step1)))*(Io*p+In)*cos(alpha+
    step1))/(2*sqrt((X+(Io*p+In)*S*sin(alpha+step1)*cos(delta+step2
    ))^2+...
        (Y+(Io*p+In)*S*sin(alpha+step1)*sin(delta+step2))^2+(Z+(Io
    *p+In)*S*cos(alpha+step1))^2));

    J(i,2) = ((2*(X+(Io*p+In)*S*sin(alpha+step1)*cos(delta+step2))
    )*(Io*p+In)*S*cos(alpha+step1)*cos(delta+step2)+ ...
        (2*(Y+(Io*p+In)*S*sin(alpha+step1)*sin(delta+step2)))*(Io*
    p+In)*S*cos(alpha+step1)*sin(delta+step2)- ...
        (2*(Z+(Io*p+In)*S*cos(alpha+step1)))*(Io*p+In)*S*sin(alpha
    +step1))/(2*sqrt((X+(Io*p+In)*S*sin(alpha+step1)*cos(delta+
    step2))^2+ ...
        (Y+(Io*p+In)*S*sin(alpha+step1)*sin(delta+step2))^2+(Z+(Io
    *p+In)*S*cos(alpha+step1))^2));

    J(i,3) = (-(2*(X+(Io*p+In)*S*sin(alpha+step1)*cos(delta+step2)
    ))*(Io*p+In)*S*sin(alpha+step1)*sin(delta+step2)+...
        (2*(Y+(Io*p+In)*S*sin(alpha+step1)*sin(delta+step2)))*(Io*
    p+In)*S*sin(alpha+step1)*cos(delta+step2))/(2*sqrt((X+...
        (Io*p+In)*S*sin(alpha+step1)*cos(delta+step2))^2+(Y+(Io*p+
    In)*S*sin(alpha+step1)*sin(delta+step2))^2+(Z+(Io*p+In)*S*cos(
    alpha+step1))^2));

    J(i,4) = ((2*(X+(Io*p+In)*S*sin(alpha+step1)*cos(delta+step2))
    )*p*S*sin(alpha+step1)*cos(delta+step2)+...
        (2*(Y+(Io*p+In)*S*sin(alpha+step1)*sin(delta+step2)))*p*S*
    sin(alpha+step1)*sin(delta+step2)+...
        (2*(Z+(Io*p+In)*S*cos(alpha+step1)))*p*S*cos(alpha+step1))
    /(2*sqrt((X+(Io*p+In)*S*sin(alpha+step1)*cos(delta+step2))^2+
    ...
```

```
        (Y+(Io*p+In)*S*sin(alpha+step1)*sin(delta+step2))^2+(Z+(Io
    *p+In)*S*cos(alpha+step1))^2));
end

end
```

B.5 MCMC

Here are the Matlab scripts that show how all the MCMC calculations are made. This specific is made to estimate S, α and δ, but it can be applied any purpose as long as the function and data are changes accordingly. Note also the σ and step size, they have to be adjusted to the problem. It can be run multiply time with different step size if the "load mMAP2; m0 = mMAP;" is in-comment after first run.

Credit to C. Finlay, who made the script.

```
% This is the main Matlab script, it will direct you to all
% the other scrips:

% Example of application of MCMC to simple parameter estimation
% problem Adapted from Parameter Estimation and Inverse Problems,
% 2nd edition, 2011.
% Example 11.4 by R. Aster, B. Borchers, C. Thurber

% By C.Finlay 06.04.2017
% For the DTU MSc class Inverse problems in Earth and
% Space Sciences

% Make sure we have a clean environment
clear all
close all

rand('state',0);
randn('state',0);

% Global variables for use by the mcmc function calls
global x;
global y;
global sigma;
global step;

% Load the data
load ../Data/Prior_ABZ_dataset;
Adda = ABZ; clear ABZ;

idx_e = find(Adda(:,3) == 0);
idx_p = find(Adda(:,3) > 0);
idx_m = find(Adda(:,3) < 0);

Q = zeros(length(Adda),2);
Q(idx_p,1) = 1;
Q(idx_m,2) = 1;

% Select data for the function
```

```
x = [Adda(:,[6:8]), Adda(:,[4,9,10]),Q ];
% Select total intensity measured by ABZ
y = Adda(:,2);

sigma=0.05*ones(length(x),1); % sigma=0.01*ones(size(ytrue));

% Set the MCMC parameters
% number of skips to reduce autocorrelation of models
skip    = 1000;
% burn-in steps
BURNIN = 20000;
% number of posterior distribution samples
N       = 410000;


% MVN step size
step = [0.0001; 0.0001; 0.0001; 0.0001];
m_l = length(step);
% We assume flat (uniform) priors here

%% initialize model at a random point on [-1,1]
load mMAP; m0 = mMAP;
tic
[mout,mMAP,pacc] = ...
    mcmc('logprior','loglikelihood','generate','logproposal',m0,N)
    ;
toc
save('mMAP.mat','mMAP')

disp(['Acceptance Rate: ',num2str(pacc)]);

% Downsample results to reduce correlation
% and exclude burnin period
k=(BURNIN:skip:N);

mskip=mout(:,k);

for i = 1:m_l
      [~, H] = hist(mskip(i,:));
      [~, idx] = max(hist(mskip(i,:)));
      mMAX(i,:) = H(idx);
end
% Histogram results, and find the modes of the subdistributions
% as an estimte of the MAP model
disp(['m_map','  m_max'])
[mMAP, mMAX]

% Estimate the 95% credible intervals
for i=1:m_l
  msort = sort(mskip(i,:));
  m2_5(i) = msort(round(2.5/100*length(mskip)));
  m97_5(i) =  msort(round(97.5/100*length(mskip)));
  disp(['95% confidence interval for m', num2str(i),' is [',
    num2str(m2_5(i)),',', num2str(m97_5(i)),']'])
end

%% plot a scatter plot and histogram of the posterior distribution
Hlims = [m2_5' , m97_5'];
```

```
Hline = 100.*ones(m_l,1);

C_jet = jet(length(mskip));

figure()
for i=1:m_l
  for j=1:m_l
    subplot(m_l,m_l,m_l*(i-1)+j)
    if i==j
      hist(mskip(i,:));
      h = findobj(gca,'Type','patch');
      set(h,'FaceColor','k')
      line([mMAX(i) , mMAX(i) ], [0, Hline(i)],'LineWidth',1.5);
    else
        for kk = 1:length(mskip)
        hold on
        plot(mskip(j,kk),mskip(i,kk),'k.','Markersize',10,'Color',
    C_jet(kk,:));
        end
        hold on
      plot(mMAX(j),mMAX(i),'ko','Markersize',12,'LineWidth',3);
      plot([m2_5(j),m97_5(j)],[m2_5(i),m2_5(i)],'k-','LineWidth'
    ,1);
      plot([m2_5(j),m97_5(j)],[m97_5(i),m97_5(i)],'k-','LineWidth'
    ,1);
      plot([m2_5(j),m2_5(j)],[m2_5(i),m97_5(i)],'k-','LineWidth'
    ,1);
      plot([m97_5(j),m97_5(j)],[m2_5(i),m97_5(i)],'k-','LineWidth'
    ,1);
      hold off
    end
  end
end

subplot(m_l,m_l,1)
ylabel('S')
subplot(m_l,m_l,5)
ylabel('\alpha')
subplot(m_l,m_l,9)
ylabel('\delta')
subplot(m_l,m_l,13)
ylabel('Current offset')

subplot(m_l,m_l,1)
xlabel('S');set(gca, 'XAxisLocation', 'top')
subplot(m_l,m_l,2)
xlabel('\alpha');set(gca, 'XAxisLocation', 'top')
subplot(m_l,m_l,3)
xlabel('\delta');set(gca, 'XAxisLocation', 'top')
subplot(m_l,m_l,4)
xlabel('Current offset');set(gca, 'XAxisLocation', 'top')
% print('MCMC_model_distrubution','-depsc')

%%
subplot(m_l,m_l,15)
ax = gca;
ax.YAxis.TickLabelFormat = '%,.4f';
ax.YAxis.Exponent = 0;
```

```
ax.XAxis.TickLabelFormat = '%,.4f';
ax.XAxis.Exponent = 0;

%% plot parameter sample histories
figure(2)
for i=1:m_l
  subplot(m_l,1,i)
  hold on
  plot(mskip(i,:),'ko')
  hold off
  if i~=m_l
    set(gca,'Xticklabel',[]);
  end
  xlim([1 length(mskip)])
end
xlabel('Sample Number')
subplot(m_l,1,1)
ylabel('m_1')
subplot(m_l,1,2)
ylabel('m_2')
subplot(m_l,1,3)
ylabel('m_3')
% print('MCMC_model_Sample','-depsc')

% plot parameter correlations
figure(3)
laglen=50;
lags=(-laglen:laglen)';
for i=1:m_l
  acorr(:,i)=calc_corr(mskip(i,:)',laglen);
  subplot(m_l,1,i);
  plot([0 laglen],[0 0],'Color',[0.7 0.7 0.7],'LineWidth',3);
  hold on
  plot(lags(laglen+1:laglen*2+1),acorr(laglen+1:laglen*2+1,i),'ko'
    );
  hold off
  ylabel(['A ( m_',num2str(i),')'])
  ylim([-0.5 1])
  if i~=m_l
    set(gca,'Xticklabel',[]);
  end
end
xlabel('Lag')
% print('MCMC_model_Lag','-depsc')


%% test if the model parameter can predict correct.
%  Note this is on the same data which the model
%  parameters are made from.

Fun = @(m, x) (sqrt( (x(:,1) + (x(:,4) + x(:,7).*m(4)).*m(1).*
    sin(m(2) + x(:,5)).*cos(m(3) + x(:,6))).^2 + ...
                        (x(:,2) + (x(:,4) + x(:,7).*m(4)).*m(1).*
    sin(m(2) + x(:,5)).*sin(m(3) + x(:,6))).^2 + ...
                        (x(:,3) + (x(:,4) + x(:,7).*m(4)).*m(1).*
    cos(m(2) + x(:,5))).^2   ));

est = Fun(mMAP, x);
```

```
202
203 figure()
204 subplot(3,1,1)
205 hold on
206 plot(Adda(:,1), est,'o')
207 plot(Adda(:,1), Adda(:,2),'.')
208 legend('Estimation','Data','Location','Best')
209 ylabel('[nT]')
210 ax = gca;
211 ax.YAxis.TickLabelFormat = '%,.2f';
212 ax.YAxis.Exponent = 0;ylim([54400, 54800])
213 xlim([Adda(1,1), Adda(end,1)])
214 title('Prediction - Positive current')
215 grid on
216 subplot(3,1,2)
217 hold on
218 plot(Adda(:,1), est,'o')
219 plot(Adda(:,1), Adda(:,2),'.')
220 legend('Estimation','Data','Location','Best')
221 title('Prediction - Zero current')
222 ylabel('[nT]')
223 ax = gca;
224 ax.YAxis.TickLabelFormat = '%,.2f';
225 ax.YAxis.Exponent = 0;
226 ylim([50190, 50215])
227 xlim([Adda(1,1), Adda(end,1)])
228 grid on
229 subplot(3,1,3)
230 hold on
231 plot(Adda(:,1), est,'o')
232 plot(Adda(:,1), Adda(:,2),'.')
233 legend('Estimation','Data','Location','Best')
234 ylabel('[nT]')
235 ax = gca;
236 ax.YAxis.TickLabelFormat = '%,.2f';
237 ax.YAxis.Exponent = 0;
238 ylim([45300, 45800])
239 xlim([Adda(1,1), Adda(end,1)])
240 title('Prediction - Negative current')
241 xlabel('Time since midnight [s]')
242 ylabel('[nT]')
243 grid on
244 %%% print('Calibration_Prediction','-depsc')
245
246 idx_e = find(Adda(:,2) > 50000 & Adda(:,2) < 51000);
247 hist_e = Adda(idx_e,2) - est(idx_e);
248 idx_p = find(Adda(:,2) > 51000);
249 hist_p = Adda(idx_p,2) - est(idx_p);
250 idx_m = find(Adda(:,2) < 50000);
251 hist_m = Adda(idx_m,2) - est(idx_m);
252
253 %%
254 figure()
255 subplot(3,1,1)
256 histogram(hist_e)
257 title({'Residuals of predicted S, \alpha and \delta','vs ABZ data'
    })
258 ylabel('Counts')
```

```
subplot(3,1,2)
histogram(hist_p)
ylabel('Counts')
subplot(3,1,3)
histogram(hist_m)
ylabel('Counts')
xlabel('Rediduals [nT]')
%%% print('Histogram_Residuals_Prediction','-depsc')
```

```
% Inspired By:
% Example 11.4
% from Parameter Estimation and Inverse Problems,
% 2nd edition, 2011 by R. Aster, B. Borchers, C. Thurber

function y=fun(m,x)

y = (sqrt( (x(:,1) +  (x(:,4) + x(:,7).*m(4)).*m(1).*sin(m(2) + x
    (:,5)).*cos(m(3) + x(:,6))).^2 + ...
           (x(:,2) +  (x(:,4) + x(:,7).*m(4)).*m(1).*sin(m(2) + x
    (:,5)).*sin(m(3) + x(:,6))).^2 + ...
           (x(:,3) +  (x(:,4) + x(:,7).*m(4)).*m(1).*cos(m(2) + x
    (:,5))).^2   ));
end
```

```
% Parameter Estimation and Inverse Problems, 2nd edition, 2011
% by R. Aster, B. Borchers, C. Thurber
%
% mout=mcmc(logprior,loglikelihood,generate,logproposal,m0,niter)
%
%  logprior           Name of a function that computes the log of
%                     the prior distribution.
%  loglikelihood      Name of a function the computes the log of
%                     the likelihood.
%  generate           Name of a function that generates a random
%                     model from the current model using the
%                     proposal distribution.
%  logproposal        Name of a function that computes the log of
%                     the proposal distribution r(x,y).
%  m0                 Initial model.
%  niter              Number of iterations to perform.
%
%  mout               MCMC samples.
%  mMAP               Best model found in the MCMC simulation.
%  accrate            Acceptance rate
%
function [mout,mMAP,accrate]=mcmc(logprior,loglikelihood,generate,
    logproposal,m0,niter)

%  Figure out some size information.

n=length(m0);

% Allocate space for the results.

mout=zeros(n,niter);

% Initialize the starting point.

```

```
mout(:,1)=m0;
current=m0;
lMAP=-Inf;
mMAP=current;
nacc=0;
%
% The main loop.
%
for k=2:niter

% Generate a candidate model from the previous model.

   candidate=feval(generate,current);

% Evalate the logarithm of the acceptance ratio.

   lpcandidate=feval(logprior,candidate);
   llcandidate=feval(loglikelihood,candidate);
   lr1=feval(logproposal,candidate,current);
   lr2=feval(logproposal,current,candidate);
   lpcurrent=feval(logprior,current);
   llcurrent=feval(loglikelihood,current);
   logalpha=lpcandidate+llcandidate+lr1-lpcurrent-llcurrent-lr2;
%
% Take the minimum of the log(alpha) and 0.
%
   if (logalpha>0)
     logalpha=0;
   end
%
% Generate a U(0,1) random number and take its logarithm.
%
   logt=log(rand());
%
% Accept or reject the step.
%
   if (logt < logalpha)
%
% Accept the step.
%
     current=candidate;
     nacc=nacc+1;
%
% Update the MAP solution if this one is better.
%
     if ((lpcandidate+llcandidate) > lMAP)
       lMAP=lpcandidate+llcandidate;
       mMAP=candidate;
     end
   else
%
% Reject the step.
%
   end
%
% Record the result.
%
   mout(:,k)=current;
```

```
  accrate=nacc/niter;
end
```

```
% Example 11.4
% from Parameter Estimation and Inverse Problems,
% 2nd edition, 2011 by R. Aster, B. Borchers, C. Thurber
% y=generate(x)
%
%
% For this problem, we'll use a multivariate normal generator,
% with standard deviations specified by the vector step.

% Note that logproposal.m and generate.m
% are closely tied to each other.

function y=generate(x)
global step
y = x + step.*randn(length(step),1);
end
```

```
% Example 11.4
% from Parameter Estimation and Inverse Problems,
% 2nd edition, 2011 by R. Aster, B. Borchers, C. Thurber
% lp=logprior(m)
%
% lb = [15000; 40000; 0; -1; -1; -60*1e-3; -pi ];
% ub = [20000; 50000; 150;  1; 1; 60*1e-3 ; pi  ];

function lp=logprior(m)
% if  ((m(1)>=-200)       && (m(1)<=200) && ...
%      (m(2)>=-60*1e-3) && (m(2)<=60*1e-3) &&...
%      (m(3)>=0)          && (m(3)<=pi))

  lp=0;
% else
%   lp=-Inf; qq = 1;
% end
end
```

```
% Example 11.4
% from Parameter Estimation and Inverse Problems,
% 2nd edition, 2011 by R. Aster, B. Borchers, C. Thurber
% l=loglikelihood(m)
%
function l=loglikelihood(m)

% global variables.
global x;
global y;
global sigma;

% Compute the standardized residuals.
fvec=(y-fun(m,x))./sigma;

l=(-1/2)*sum(fvec.^2);
end
```

```
% Example 11.4
% from Parameter Estimation and Inverse Problems,
% 2nd edition, 2011 by R. Aster, B. Borchers, C. Thurber

% For this problem, we'll use a multivariate normal generator,
% with standard deviations specified by the vector step.

% Note that logproposal.m and generate.m
% are closely tied to each other.
%
function lr=logproposal(x,y)
global step

lr=(-1/2)*sum((x-y).^2./step.^2);
end
```

```
function c = calc_corr(x,laglen)
%
%c = calc_corr(x,laglen)
%
%returns the first laglen elements of the circular (normalized)
    crosscorrelation of the column vector x
%
c=zeros(laglen,1);
x=x-mean(x);
c(1)=dot(x,x);
for i=2:laglen+1
    c(i)=dot(x,circshift(x,i-1));
end
c=c/c(1);
c=[flipud(c(2:end));c];
```

```
% Example 11.4
% from Parameter Estimation and Inverse Problems,
% 2nd edition, 2011 by R. Aster, B. Borchers, C. Thurber
% acceptance probability function probacc

function z=getlnprobacc(m1,c,x,y,sigma)
%calculate log likelihoods
ll1=(-1/2)*sum((y-fun(c,x)).^2./sigma);
ll2=(-1/2)*sum((y-fun(m1,x)).^2./sigma);
z=min(0,ll1-ll2);
return;
```

C

Baseline improvement

```
clear all
close all


%% Baseline

fileID = fopen('BaselinesFromDI\THL_baselines.txt'); %  fileID = fopen('test_1.txt'); %
frewind(fileID); % Move file position indicator to beginning of open file
C_text = textscan(fileID,'%s',3,'Delimiter','\n');
Base = textscan(fileID,'%f%f%f %f%f%f%f%f%f%f%f%f %s %f%f%f%f%f%f%f%f%f');
fclose(fileID);



temp = num2str([Base{1,2}, Base{1,3}]);
% Year , Month, Day, time [s], Hb, Db, Zb
Baseline = [str2num(temp(:,1:8)), str2num(temp(:,15:16))*60 + str2num(temp(:,17:18)), Base{1,8}, Base{1,9}, Base{1,10}];
% Baseline = [str2num(temp(:,1:4)), str2num(temp(:,5:6)), str2num(temp(:,7:8)), str2num(temp(:,15:16))*60 + str2num(temp(:,17:18)), Base{1,8}, Base{1,9}, Base{1,10}];

clear temp Base C_text fileID;
%% find ppm & FGE data in baseline interval  (Credit: Anna Naemi Willer for Read of .cdf)

%%%%%%%%%%   ppm data    %%%%%%%%%%
temp = struct2cell(dir('ppm'))';
temp = char(temp(:,1));
ppm_name = str2num(temp(:,6:13));

clear temp;


% find interval
A = find(ppm_name <= Baseline(end,1) & ppm_name >= Baseline(1,1));
Files_ppm = dir('ppm');


ppm_data = [];
for i = A'
FileNames = Files_ppm(i+2).name;
ppm = importdata(['ppm\',num2str(FileNames)], ' ');
```

```
40
41 temp = char(ppm.textdata(:,1));
42 ppm_data = [ppm_data; ppm_name(i).*ones(length(temp),1), ...
43               str2num(temp(:,1:2))*60 + str2num(temp(:,4:5)) +
       str2num(temp(:,7:8))/60,...
44               ppm.data];
45 end
46
47 clear temp A ppm ppm_name Files_ppm FileNames;
48 %%%%%%%%%%   variometer FGE   %%%%%%%%%%
49
50
51 temp = struct2cell(dir('variometerData'))';
52 temp = char(temp(:,1));
53 var_name = str2num(temp(:,6:13));
54
55 clear temp;
56
57 % find interval
58 A = find(var_name <= Baseline(end,1) & var_name >= Baseline(1,1));
59 Files_var = dir('variometerData');
60
61
62 var_data = [];
63 for i = A'
64     FileNames = Files_var(i+2).name;
65
66     cdf_file = ['variometerData\',num2str(FileNames)];
67
68     % Assign a file number to the cdf file
69     cdfID = cdflib.open(cdf_file);
70     % Store information about the CDF file
71     info = cdfinfo(cdf_file);
72     % Reading: read the variables
73     alldata = cdfread(cdf_file, 'CombineRecords', 1);
74     for j=1:size(info.Variables,1)
75         data.(char(info.Variables(j))) = alldata{j};
76     end
77     % Close cdf file
78     cdflib.close(cdfID);
79     t_var = (data.time(:) - round(data.time(1)))*24*60;
80
81     var_data = [var_data;  var_name(i).*ones(length(data.time),1),
       ...
82         t_var, data.HNvar , data.HEvar, data.Zvar];
83 end
84
85 clear temp A var var_name Files_var FileNames;
86
87 %% Find one minute ppm & FGE data between baseline values
88
89 % Make time between base line
90
91 clear All_sp;
92 for i = 1:length(Baseline)-1
93     clear td t_1m temp1 index;
94 B1 = find(Baseline(i,1) <= ppm_data(:,1) & Baseline(i,2) <=
       ppm_data(:,2),1);
```

```
B2 = find(Baseline(i+1,1) <= ppm_data(:,1) & Baseline(i+1,2) <=
    ppm_data(:,2),1);

temp_ppm = ppm_data(B1:B2,:);

for k = 1:length(temp_ppm)
temp1 = num2str(temp_ppm(k,1));
td(k,:) = (sum(eomday(str2num(temp1(:,1:4)),[1:str2num(temp1
    (:,5:6))])) + str2num(temp1(:,7:8)) )*24*60 + temp_ppm(k,2);
end

t_1m = [round(td(1,1)):round(td(end,1))]';
All_sp{1,i} = [Baseline(i,:), t_1m(1); Baseline(i+1,:), t_1m(end)
    ];
[td, index] = unique(td);
All_sp{2,i} = [t_1m, interp1(td, temp_ppm(index,end), t_1m,'linear
    ','extrap')];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     clear td t_1m temp1 index;
B3 = find(Baseline(i,1) <= var_data(:,1) & Baseline(i,2) <=
    var_data(:,2),1);
B4 = find(Baseline(i+1,1) <= var_data(:,1) & Baseline(i+1,2) <=
    var_data(:,2),1);

temp_var = var_data(B3:B4,:);

for k = 1:length(temp_var)
temp1 = num2str(temp_var(k,1));
td(k,:) = (sum(eomday(str2num(temp1(:,1:4)),[1:str2num(temp1
    (:,5:6))])) + str2num(temp1(:,7:8)) )*24*60 + temp_var(k,2);
end

t_1m = [round(td(1,1)):round(td(end,1))]';
[td, index] = unique(td);
All_sp{3,i} = [t_1m, ...
    interp1(td, temp_var(index,3), t_1m,'linear','extrap'), ...
    interp1(td, temp_var(index,4), t_1m,'linear','extrap'), ...
    interp1(td, temp_var(index,5), t_1m,'linear','extrap')];

end


save('All_sp.mat','All_sp')
```

```
clear all
close all

load All_sp.mat


for base_num = 2


    A = [];
for i =1:length(All_sp{2,base_num}(:,1))
    if find(All_sp{2,base_num}(i,1) == All_sp{3,base_num}(:,1))
    A = [A; i];
```

```
    end
end

sp1{1,1} = All_sp{1,base_num};
sp1{2,1} = All_sp{2,base_num}(A,:);
sp1{3,1} = All_sp{3,base_num}(A,:);



    %% Inetial S-grid

Bp = [sp1{1,1}(1,3:5)];
Bn = [sp1{1,1}(2,3:5)];

Hb = interp1(sp1{2,1}([1, end],1), [Bp(1) , Bn(1)], sp1{2,1}(:,1))
    ;
Db = interp1(sp1{2,1}([1, end],1), [Bp(2) , Bn(2)], sp1{2,1}(:,1))
    ;
Zb = interp1(sp1{2,1}([1, end],1), [Bp(3) , Bn(3)], sp1{2,1}(:,1))
    ;

S = [Hb , Db , Zb];
%% optimize using x = lsqnonlin(fun,x0,lb,ub,options)
lambda = 0.5;
options = optimoptions('lsqnonlin','Display','iter');

sp1{1,1} = All_sp{1,base_num};
sp1{2,1} = All_sp{2,base_num}(A,:);
sp1{3,1} = All_sp{3,base_num}(A,:);

fun = @(S) phi(S, sp1, lambda);
x = lsqnonlin(fun,S,[],[],options);

p_orginal{1,base_num} =  phi(S, sp1, lambda);
p_est{1,base_num} =  phi(x, sp1, lambda);

x_save{1,base_num} = x;
S_save{1,base_num} = S;
end

% save('p_orginal.mat','p_orginal')
% save('p_est.mat','p_est')
%
% save('Baseline_est.mat','x_save')
% save('Baseline_start.mat','S_save')


%%

% for QQ = 1
QQ = base_num;

figure()
subplot(2,1,1)
hold on
plot(p_orginal{1,QQ}(:,1),'^')
plot(p_orginal{1,QQ}(:,2),'.')
plot(p_orginal{1,QQ}(:,3),'--')
```

```
legend('Hb [nT]', 'Db[^o]', 'Zb [nT]')
title('\phi(t, S^{i,j,k}) with original baseline values')
xlabel('From Bp to Bn [min]')
ylabel('[nT]')
grid on
subplot(2,1,2)
hold on
plot(p_est{1,QQ}(:,1),'^')
plot(p_est{1,QQ}(:,2),'.')
plot(p_est{1,QQ}(:,3),'--')
legend('Hb [nT]', 'Db[^o]', 'Zb [nT]')
title('\phi(t, S^{i,j,k}) with estimatied improved baseline ')
xlabel('From Bp to Bn [min]')
ylabel('[nT]')
grid on
% print('../../LaTeX/fig/B_im01','-depsc')
% end

ii = 1;
for i = 1:60:length(x)
    if(i+60 < length(x))
        int = [i:i+60];
    else
        int = [i:length(x)];
    end
    Hour_mean(ii,1) = mean(x(int,1));
    Hour_mean(ii,2) = mean(x(int,2));
    Hour_mean(ii,3) = mean(x(int,3));
    Hour_mean(ii,4) = int(round(length(int)/2));
    ii = ii+1;
end


figure()
subplot(3,1,1)
hold on
plot(S(:,1),'Linewidth',2)
plot(x(:,1),'.')
plot(Hour_mean(:,4),Hour_mean(:,1),'o', 'Linewidth',2, 'color', 'g
    ')
legend('Linear baseline estimation', 'Routinely improved baseline'
    ,'Hourly mean', 'Location', 'best')
title('H baseline')
xlabel('From Bp to Bn [min]')
ylabel('[nT]')
ax = gca;
ax.YAxis.TickLabelFormat = '%.0f';
ax.YAxis.Exponent = 0;
grid on
subplot(3,1,2)
hold on
plot(S(:,2),'Linewidth',2)
plot(x(:,2),'.')
plot(Hour_mean(:,4), Hour_mean(:,2),'o', 'Linewidth',2, 'color', '
    g')
legend('Linear baseline estimation', 'Routinely improved baseline'
    ,'Hourly mean', 'Location', 'best')
title('D baseline')
```

```
xlabel('From Bp to Bn [min]')
ylabel('[^o]')
ax = gca;
ax.YAxis.TickLabelFormat = '%.0f';
ax.YAxis.Exponent = 0;
grid on
subplot(3,1,3)
hold on
plot(S(:,3),'Linewidth',2)
plot(x(:,3),'.')
plot(Hour_mean(:,4),Hour_mean(:,3),'o', 'Linewidth',2, 'color', 'g')
legend('Linear baseline estimation', 'Routinely improved baseline','Hourly mean', 'Location', 'best')
title('Z baseline')
xlabel('From Bp to Bn [min]')
ylabel('[nT]')
ax = gca;
ax.YAxis.TickLabelFormat = '%.0f';
ax.YAxis.Exponent = 0;
grid on


% print('../../LaTeX/fig/B_im02','-depsc')


%%

[~, G, ~, ~, ~] =  phi(S, sp1, lambda);

figure()
histfit(G)
title('G-matrix')
ylabel('Counts')
xlabel('[nT]')
% print('../../LaTeX/fig/G_hist','-depsc')
```

```

% Inspired by On the feasibility of routine baseline improvment,
% by Anatoly Sloview and CO.

function [p, G, A, G1, Fppm] = phi(S, sp1, lambda)

Hb = S(:,1);
Db = S(:,2);
Zb = S(:,3);

%% S-grid

Bp = [sp1{1,1}(1,3:5)];
Bn = [sp1{1,1}(2,3:5)];


%%  G-matrix

H = sqrt((sp1{3,1}(:,2) + Hb).^2  + sp1{3,1}(:,3).^2  );
D = atan(sp1{3,1}(:,3)./(sp1{3,1}(:,2) + Hb)) + Db;
Z =  sp1{3,1}(:,4) + Zb;
```

```
Fppm = sp1{2,1}(:,2);

G1 = sqrt((H.*cos(D)).^2 + (H.*sin(D)).^2 + Z.^2);
G = (sqrt((H.*cos(D)).^2 + (H.*sin(D)).^2 + Z.^2) - Fppm); % The
    LSQ is

%% A-matrix

delta_tp = [1:length(sp1{2,1}(:,1))]';
delta_tn = [length(sp1{2,1}(:,1))':(-1):1]';
T = delta_tn + delta_tp;

wp = 1 - delta_tp ./ T;
wn = 1 - delta_tn ./ T;

A = ((repmat(Bp,[length(S),1]) - S).*repmat(wp,[1,3]) ) + ...
     ((repmat(Bn,[length(S),1]) - S).*repmat(wn,[1,3]) ); % The LSQ
     is part of the lsqnonlin();


%% phi

G = G - mean(G);
p = repmat(lambda.*G,[1,3]) + (1-lambda).*A;

end
```

D

Additional material

D.1 Extra results

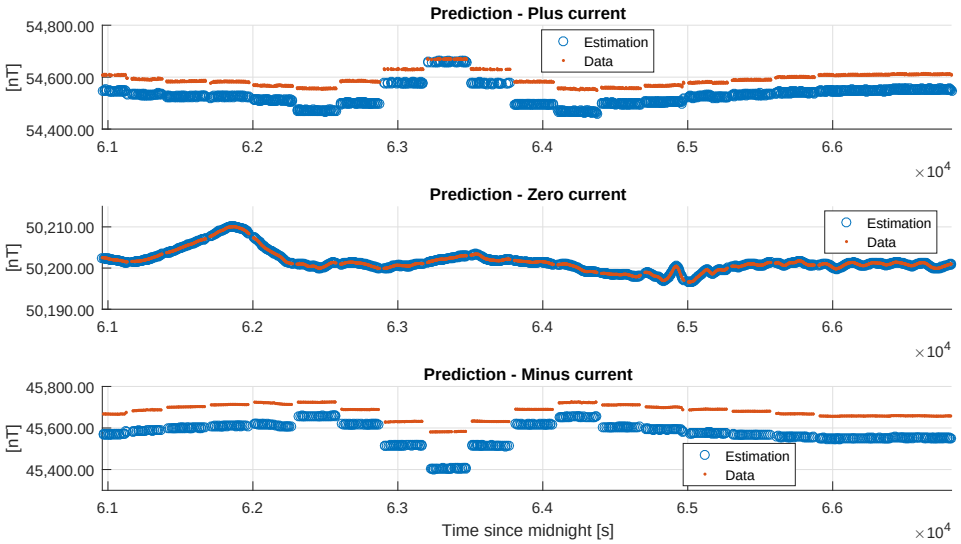


Figure D.1: *How the prediction looks without the current offset. By comparing it with Figure 4.5 is clear that a current offset is needed for a genuine fit.*

D.2 Prior information as regularization

Another way to regulate instability occurring in non-linear problem, is by use of prior information. A way of using prior information is examined in Tarantola (2005)[p.64-68], here both knowledge of expected data and model parameters is taken into considerations. The syntax is a bit different from Tarantola (2005)[p.64-68], in the following equation, but the idea of including prior information is the same.

$$\boldsymbol{x}_* \approx \mathbf{m}_p + (\mathbf{J}(\mathbf{m}_k)^T\mathbf{J}(\mathbf{m}_k) + \mathbf{C}_\mathbf{m}^{-1})^{-1}\mathbf{J}(\mathbf{m}_k)^T(\mathbf{r}(\mathbf{m}_k) + \mathbf{J}(\mathbf{m}_k)\Delta\mathbf{m}_p) \quad \text{(D.1)}$$

Where $\Delta\mathbf{m}_p$ is the prior experted parameter step. Again $\Delta\mathbf{m}_p$ can be separated into $\Delta\mathbf{m}_p = \mathbf{m}_p - \mathbf{m}_k$ where $\mathbf{m}_p$ is our expected model parameters. Furthermore, $\mathbf{C}_d$ is the prior covariance matrix associated with the model parameters.$\mathbf{C}_m$ is the prior covariance matrix associated with the expected

data error.

Now, it can be seen that our model parameters consist of our expected prior model parameters, and a term that evaluates the residual. This is an extremely convenient regularization method, especially if you have a expected prior distribution of your model parameters.

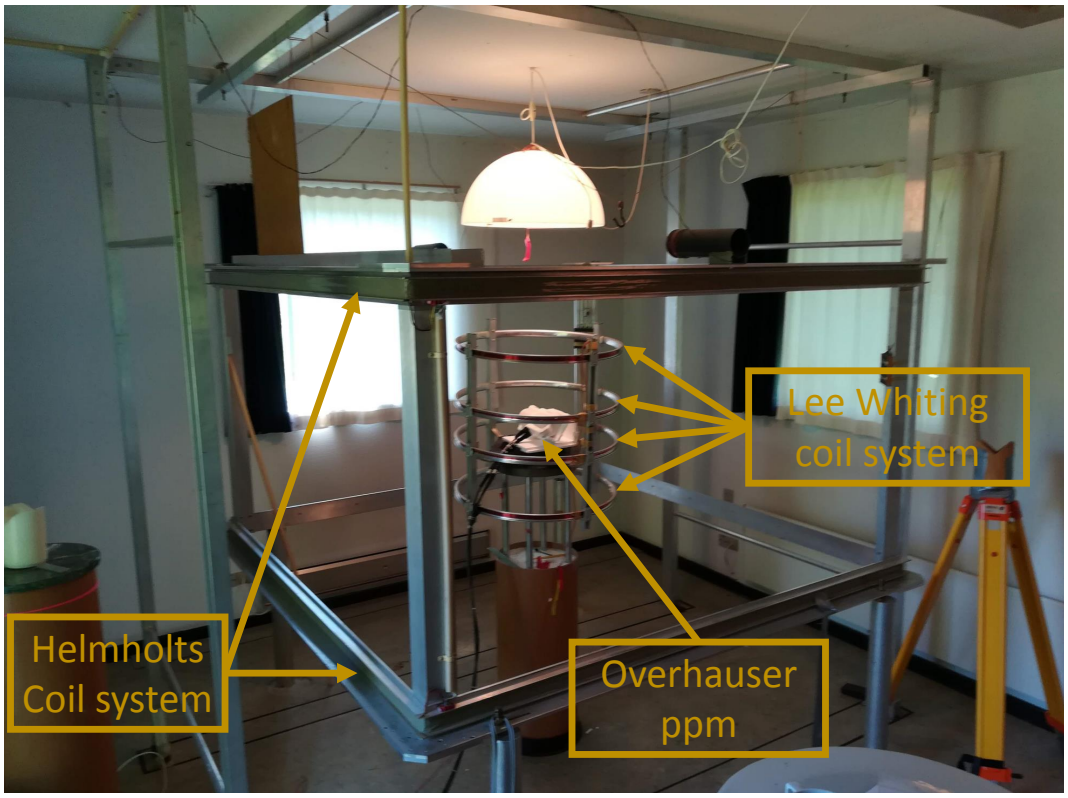
D.3 Calculations of error in ES method

```
%%  Plot error due to change in alpha and delta, for H, Z and F

clear all
close all

syms F(I,S,H,Z,alpha,delta);
F(I,S,H,Z,alpha, delta) = sqrt((H + I.*S.*sin(alpha).*cos(delta) )
    ^2 + (Z + I.*S.*cos(alpha))^2);


% syms H(S,I,alpha,delta,Z,F);
% H(S,I,alpha, delta,F,Z) = sqrt((H + I.*S.*sin(alpha).*cos(delta)
    )^2 + (Z + I.*S.*cos(alpha))^2);
%
% syms Z(S,I,alpha,delta,H,F);
% Z(S,I,alpha, delta,H,F) = sqrt((H + I.*S.*sin(alpha).*cos(delta)
    )^2 + (Z + I.*S.*cos(alpha))^2);


%% Plot  F

a = [0:1*1e-3:70*1e-3];  % alpha
d = [-pi/2:0.05:pi/2]';   % delta

[X,Y] = meshgrid(a, d);

S = 137;
I = 40;
H_E = 17207;
Z_E = 47000;

F_p = double(F(S,40,H_E,Z_E,X,Y));
F_m = double(F(S,-40,H_E,Z_E,X,Y));
F_0 = double(F(S,0,H_E,Z_E,X,Y));

Z_coil = sqrt((F_p.^2 + F_m.^2)/2 - F_0.^2);
Z = (F_p.^2 - F_m.^2)./(4.*Z_coil);
H = sqrt(F_0.^2 - Z.^2);

figure()
contourf(a, d, Z - Z_E)
c = colorbar;
c.Label.String = '[nT]';
colormap('jet')
title('Error in vertical intensity')
xlabel('Misalignment \alpha [Rad]')
```

```
ylabel('Misalignment \delta [Rad]')
% saveas(gcf,['../../LaTeX/fig/Z_error','.png'])

figure()
contourf(a, d, H - H_E)
c = colorbar;
c.Label.String = '[nT]';
colormap('jet')
title('Error in horizontal intensity')
xlabel('Misalignment \alpha [Rad]')
ylabel('Misalignment \delta [Rad]')
% saveas(gcf,['../../LaTeX/fig/H_error','.png'])
```

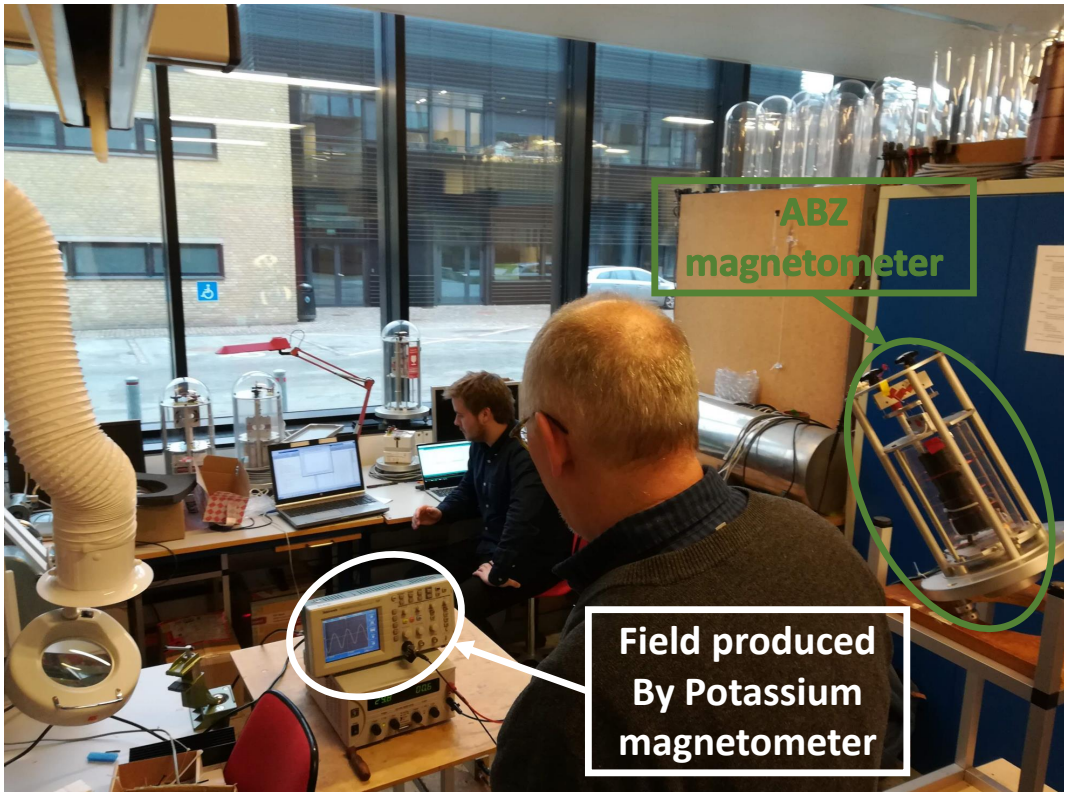
E

Pictures from measuremetns



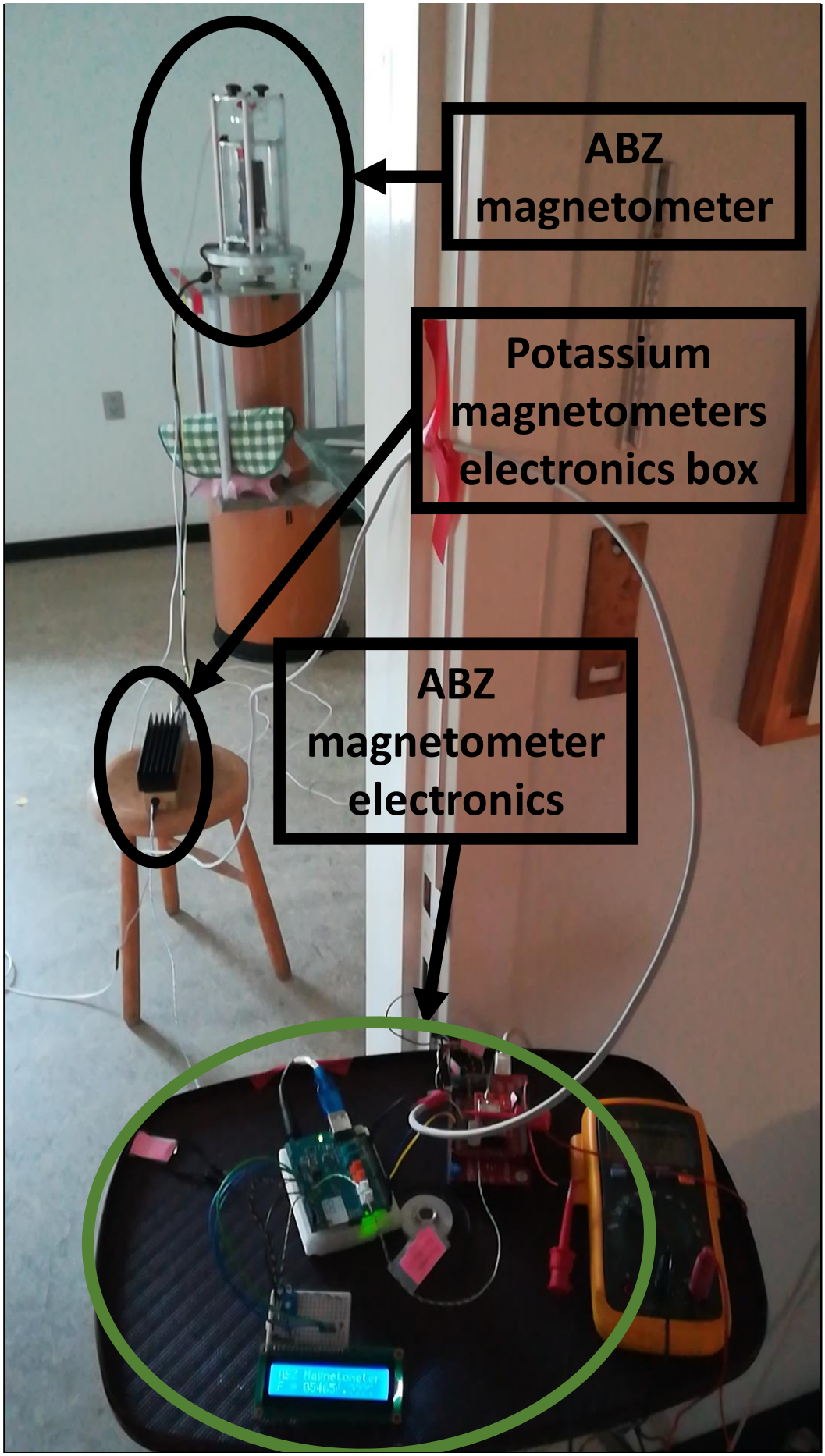
Three elements are enhance in this picture; a Helmholtz coil, an Overhauser ppm and a big Lee-Whiting coil system.
The big Lee-Whiting coil system was only tried a few times for minor test.

If the Lee-Whiting coil system was removed, a set up with a Overhauser ppm inside a Helmholtz coil system appear. This set up was used to produce the results seen in Figure 4.9.



This picture was taking at the labetoria a DTU. The oscilloscope in the picture is measureing the field produced in the coil, is was clearly produced by the Potassium magnetometer. It is also seen in close op in Figure 4.11. Lars William is closet to the camera, and Niels Skødt longest away from the camera.

.



A picture of the ABZ magnetometer while doing a survey.

F

Time schedule

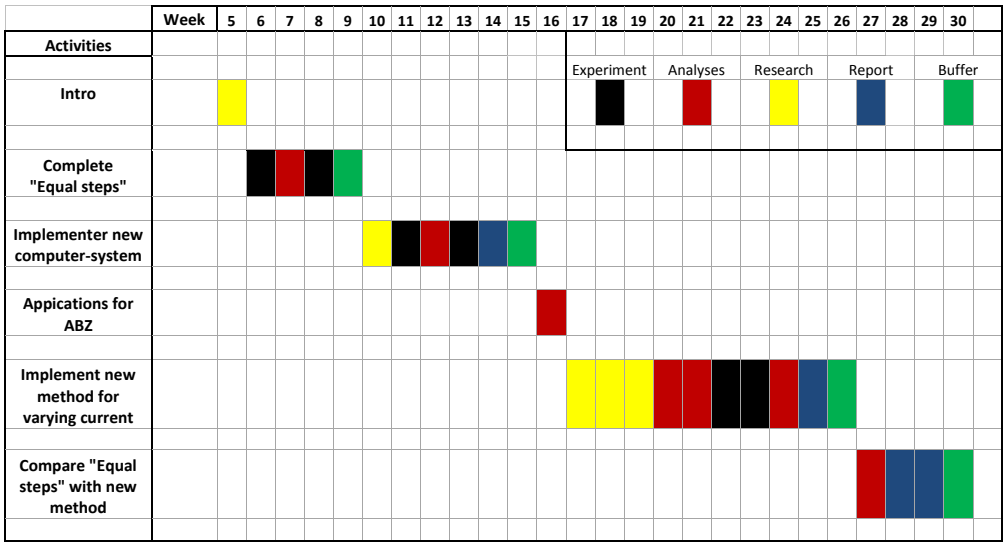


Figure F.1: *Initial time schedule*

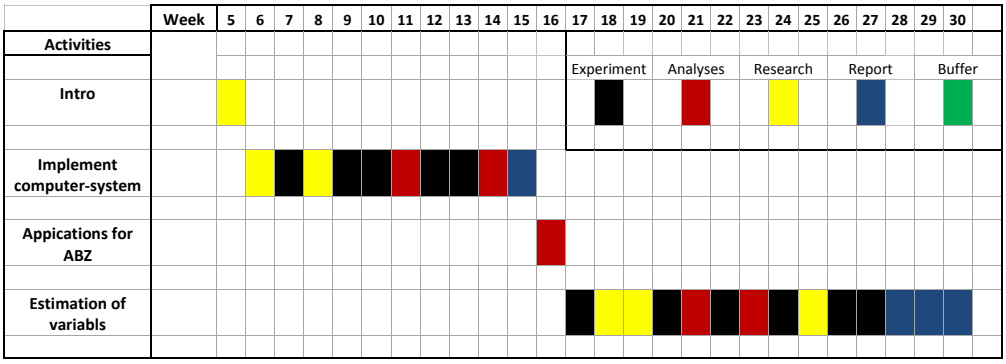
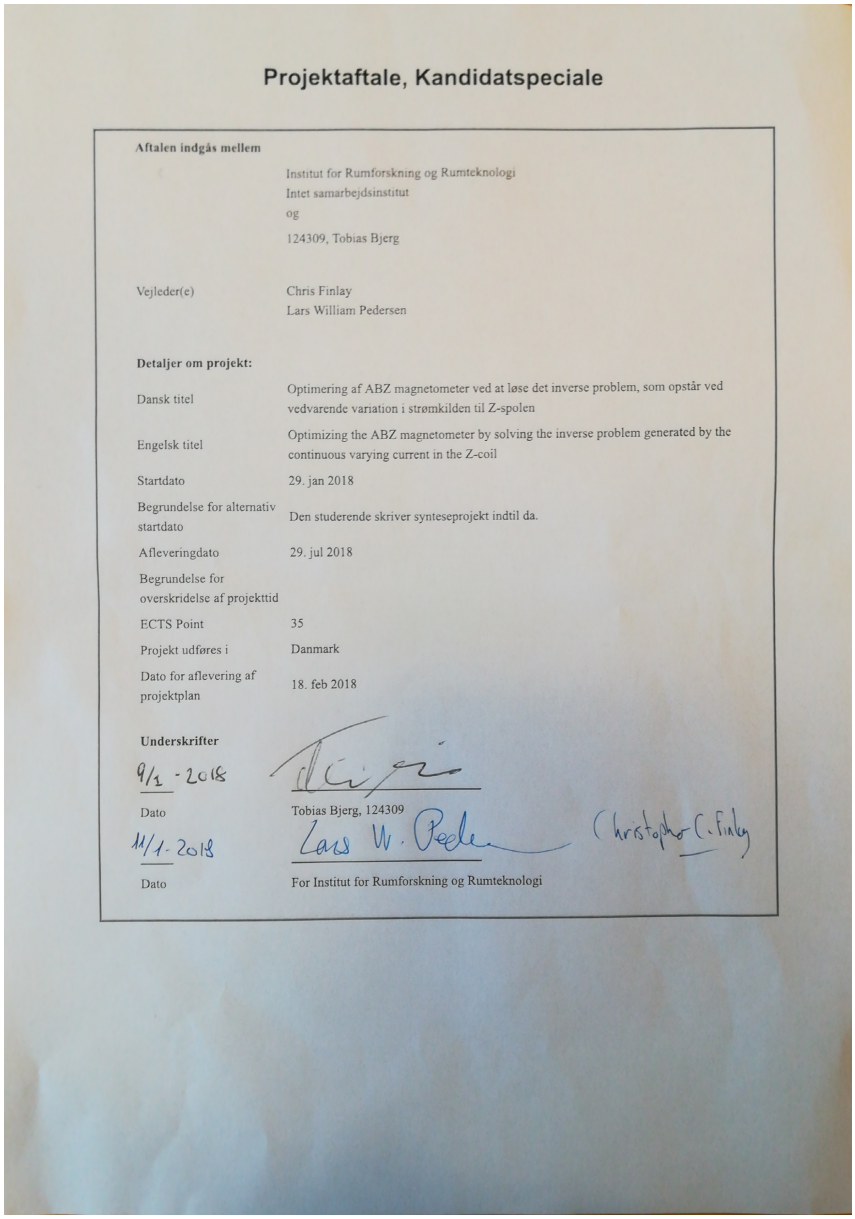


Figure F.2: *An estimated time schedule of the reallity*

G

Initial project agreement and description

Before the master project started a initial project description was formulated.



Projektaftale, Kandidatspeciale

Aftalen indgås mellem

Institut for Rumforskning og Rumteknologi
Intet samarbejdsinstitut
og
124309, Tobias Bjerg

Vejleder(e): Chris Finlay, Lars William Pedersen

Detaljer om projekt:

Dansk titel	Optimering af ABZ magnetometer ved at løse det inverse problem, som opstår ved vedvarende variation i strømkilden til Z-spolen
Engelsk titel	Optimizing the ABZ magnetometer by solving the inverse problem generated by the continuous varying current in the Z-coil
Startdato	29. jan 2018
Begrundelse for alternativ startdato	Den studerende skriver synteseprojekt indtil da.
Afleveringdato	29. jul 2018
Begrundelse for overskridelse af projekttid	
ECTS Point	35
Projekt udføres i	Danmark
Dato for aflevering af projektplan	18. feb 2018

Underskrifter

9/1 - 2018 — Dato — Tobias Bjerg, 124309

11/1-2018 — Dato — Lars W. Pedersen (Christopher C. Finlay) — For Institut for Rumforskning og Rumteknologi

Figure G.1: *Project agreement*

Master thesis:

Title: Optimizing the ABZ magnetometer by solving the inverse problem generated by the continuous varying current in the Z-coil.

Title: Optimering af ABZ magnetometer ved at løse det inverse problem, som opstår ved vedvarende variation i strømkilden til Z-spolen.

Supervisors: Lars William Pedersen, DTU Space & Chris Finlay, DTU Space.

ECTS Point: 35

The goal of this master thesis is to optimize the measurements method for the ABZ magnetometer (Automatic aBsolute Z-direction magnetometer). The fundamental construction and testing of the ABZ magnetometer is evaluated in the synthesis project "Fundamental Construction and Testing of the ABZ magnetometer", which will be handed in the 19th of January. The synthesis project will give rise to the study of optimization of the measurements method. An illustration of the construction is attached in Appendix, the construction of ABZ magnetometer will not be discussed further here or in the master thesis. The measurement method addresses various aspect for the ABZ magnetometer.

- Control the signature of current input to the coil system.
- Analyses of the information which will enable the calculation of the magnetic field components.

The traditional method, which is used for the synthesis project, generates equal current in both polarities for the coil system. This method simplifies the analyses, and the magnetic field components can easily be calculated. But due to many effects, the equal current is imprecise, and the calculations of the magnetic field components will be accordingly.

The optimize method being develop in this thesis, will measure the current at all time and include this in the calculation. This will change the current input and the following calculations become an underdetermined inverse problem.

The general headlines for the work procedure for the thesis will be:

- Make a stable and robust current source, which will vary due to the optimal conditions for the scalar magnetometer.
- Develop a robust method to calculate the generated inverse problem, in the most efficient and optimal way.
- Implement the optimized method in a microcontroller, that automatically can operate the system and display the calculated magnetic field components.
- Test, verify and undo.
- Compare the optimized method with the traditional method.
- Document in a final report.

Project deadlines: The project will start the 29th of January. The thesis hand in will be 6 months after, on the 29th of July. The master defense will take place within 10 days after the hand in, as the DTU regulations demand.

Appendix – Illustration of the ABZ magnetometer

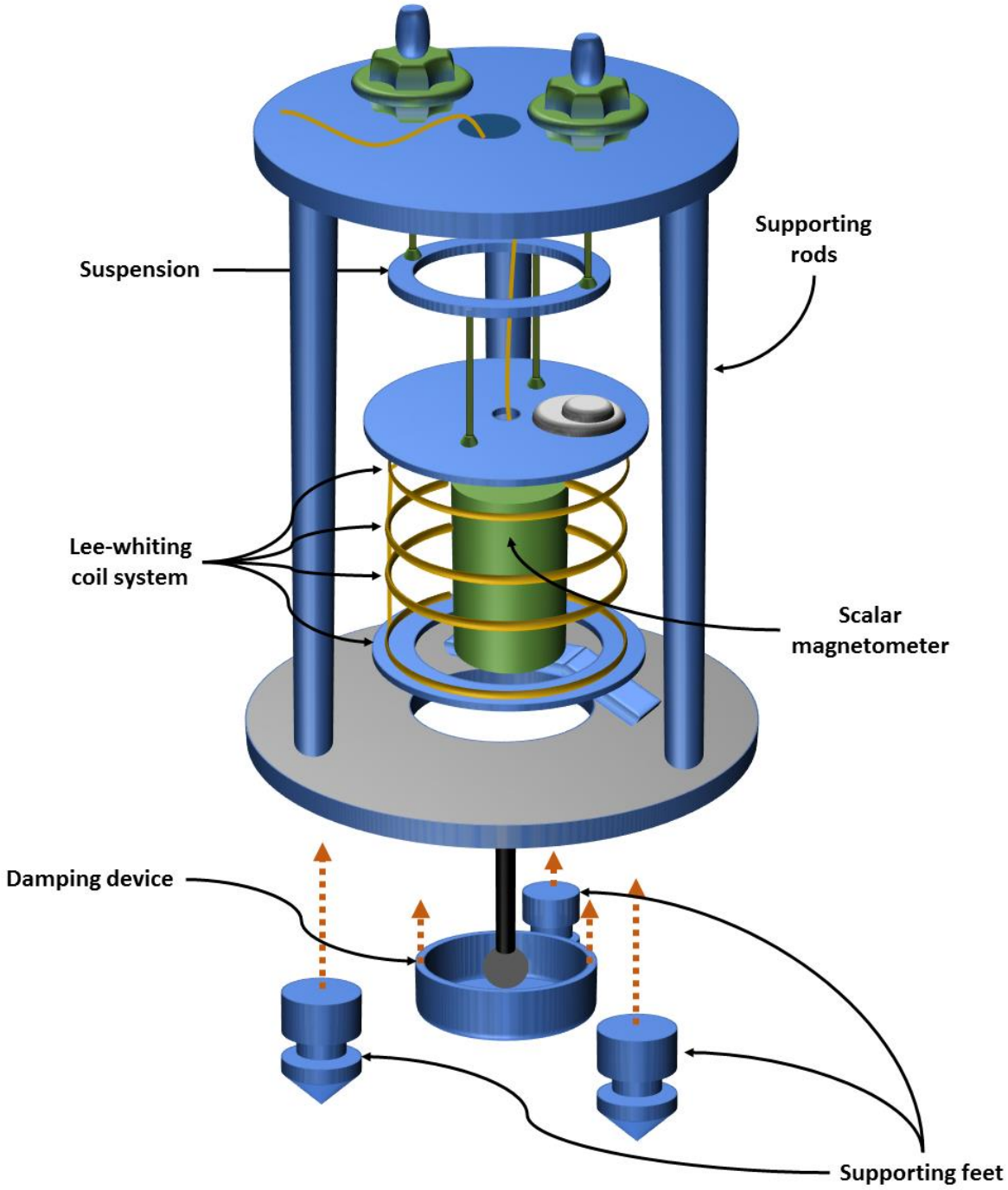


Figure 1: Construction of the ABZ magnetometer.

.

Abbreviations

Automatic aBsolute Z-coil	**ABZ**
Analog-to-digital converter	**ADC**
Confidence Intervals	**CI**
Digital-to-analog converter	**DAC**
Declination and Inclination fluxgate	**D.I. fluxgate**
Equal Steps	**ES**
3-axis FluxGate Magnetometer	**FGE/FGM**
Global Positioning System	**GPS**
Liquid-crystal display	**LCD**
Maximum a posterior	**MAP**
Markov chain Monte Carlo	**MCMC**
Master Input Slave Output	**MISO**
Master Output Slave Input	**MOSI**
Multivariate normal distribution	**MVN**
Proton precision magnetometer	**ppm**
Pulse per second	**pps**
Receive	**Rx**
Clock signal	**SCK**
Secure Digital	**SD**
Serial Peripheral Interface	**SPI**
Slave selection	**SS**
Transmit	**Tx**
Universal Serial Bus	**USB**
International Real-time Magnetic Observatory Network	**INTERMAGNET**

Magnetic total intensity	**F**
Magnetic horizontal intensity	**H**
Magnetic intensity towards geographic North	**X**
Magnetic intensity towards geographic East	**Y**
Magnetic vertical intensity	**Z**
Coil sensitivity	**S**
Coil current	**I**
Horizontal intensity slope	**a**
Vertical intensity slope	**b**
Magnetic inclination	**Icl**
Magnetic Declination	**Dcl**
Vertical misalignment of suspension	$\boldsymbol{\alpha}$
Horizontal misalignment of suspension	$\boldsymbol{\delta}$
Standard deviation	$\boldsymbol{\sigma}$
Mean	$\boldsymbol{\mu}$

Reference

R. C. Aster, B. Borchers, and C. H. Thurber. *Parameter estimation and inverse problems*, volume 90. Academic Press, 2011.

T. Bjerg. Investigation of the theory behind the abz magnetometer., 2017a. Special course in close collaboration with Lars William Pedersen and Chris Finlay, DTU Space.

T. Bjerg. Fundamental construction and testing of the abz magnetometer, 2017b. Synthesis project in close collaboration with Lars William Pedersen and Chris Finlay, DTU Space.

C. Finlay and N. Olsen. Lecture notes in geomagnetism, msc course 30740. *DTU Space - Technical University of Denmark*, May 4, 2017.

gpscompared.net. `https://www.gpscompared.net/garmin-glo-portable-gps-and-glonass-receiver`. A comparison of GSP, maximum GPS sampling rate is set to 10 Hz. Accessed: 2018-11-07.

Lucidchart.com. `https://www.lucidchart.com/pages/flowchart-symbols-meaning-explained`. Flowcharts Description, Accessed: 2018-04-04.

L. W. Pedersen and L. Merenyi. The fge magnetometer and the intermagnet 1 second standard. *J. Ind. Geophys. Union*, 5, 2016.

A. Soloviev, V. Lesur, and D. Kudin. On the feasibility of routine baseline improvement in processing of geomagnetic observatory data. *Earth, Planets and Space*, 70(1):16, 2018.

A. Tarantola. *Inverse problem theory and methods for model parameter estimation*, volume 89. siam, 2005.

H. Toh and Y. Hamano. The two seafloor geomagnetic observatories operating in the western pacific. In *SEAFLOOR OBSERVATORIES*, pages 307–323. Springer, 2015.

H. Toh, K. Satake, Y. Hamano, Y. Fujii, and T. Goto. Tsunami signals from the 2006 and 2007 kuril earthquakes detected at a seafloor geomagnetic observatory. *Journal of Geophysical Research: Solid Earth*, 116(B2), 2011.

C. Turbitt, J. Matzka, J. Rasson, B. St-Louis, and D. Stewart. An instrument performance and data quality standard for intermagnet one-second data exchange. (03/13), 2013.

`Intermagnet.org`. `http://www.intermagnet.org`. Intermagnet home page, Accessed: 2018-05-01.

`www.atomic-clock.com`. `http://www.atomic-clock.galleon.eu.com/support/gps-time-accuracy.html`. The GPS timing signal is typically accurate to 10 ns. Accessed: 2018-20-07.

S. Wright and J. Nocedal. Numerical optimization. *Springer Science*, 35(67-68):7, 1999.

www.byteparadigm.com. `https://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/?/article/AA-00255/22/Introduction-to-SPI-and-IC-protocols.html`. A protocal on SPI and I2C. Accessed: 2018-11-07.

A. Zikmund. Magnetic calibration by using non-linear optimization method. 2014.

DTU Space
National Space Institute
Technical University of Denmark

Elektrovej, building 327
DK - 2800 Kgs. Lyngby
Tel. (+45) 4525 9500
Fax (+45) 4525 9575

www.space.dtu.dk