# Algorithms and Software for Large-Scale Geophysical Reconstructions

Christian Eske Bruun & Trine Brandt Nielsen

Kongens Lyngby 2007

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

# Summary

The main focus of the thesis is algorithms and software for large-scale geophysical reconstructions. Throughout the project we deal with a special Toeplitz structure of the coefficient matrix that enables a significant loss-less compression.

The geophysical surveying problems dealt with in the thesis are by nature ill-posed. For this reason we use regularization methods to perform the reconstuctions. The methods used in the paper are TSVD, Tikhonov, and CGLS. We will describe the constraints in the surveying problems that need to be met in order to achieve the Toeplitz structures. Aside from enabling compression the Toeplitz structures makes it possible to use a FFT matrix-vector multiplication to achieve fast multiplications in the regularization methods. Multi-level observation sets are used to meet the constraints and a Fourier based upward continuation method can be used to achieve the multi-levels, however, for forward problems we are able to compute levels at different altitudes. The performance of the upward continuation method is investigated.

The result of this thesis is a Matlab object-oriented package implemented within the frameworks of M$\mathcal{OO}$ReTools and expands the existing `GravMagTools` package. The package is tested and reconstructions are performed.

*Keywords: Toeplitz structures, Large-scale inversions, Ill-posed problems, Compression, FFT matrix-vector multiplication, Regularization methods, Upward continuation, Object-oriented implementation, MOOReTools*

# Sammenfatning

I dette speciale vil det primære fokus være på algoritmer og software til geofysiske stor-skala rekonstruktioner. Vi beskæftiger os med en speciel Toeplitz struktur i koefficient matricen, der muliggøre en kompression uden tab af data.

De geofysiske problemer i afhandlingen er ill-posed. Derfor bruges regulariserings metoder til at rekonstruere problemerne. I denne afhandling benyttes TSVD, Tikhonov og CGLS. Vi vil beskrive de begrænsninger, vi er underlagt i måle problemerne for at kunne opnå Toeplitz strukturer. Toeplitz strukturerne gør os ikke alene i stand til at komprimere, men også i stand til at implementere en FFT matrix-vektor multiplication, der skal gøre multiplikationerne i regulariserings metoderne hurtigere. Observations sæt i flere planer bruges til at imødegå begrænsningerne, og en Fourier baseret upward continuation kan bruges i denne forbindelse. For forward problemer kan vi beregne observationslag for flere højder. Upward continuation undersøges og testes.

Specialet udmunder i sidste ende i en objekt-orienteret Matlab pakke, der implementeres indenfor rammerne af M$\mathcal{OO}$ReTools og udvider den allerede eksisterende `GravMagTools` pakke. Slutteligt testes pakken og rekonstruktioner foretages.

# Preface

This master thesis is prepared at the Technical University of Denmark (DTU). The study was carried out between September 2006 and April 2007 at the institute of Informatics Mathematical Modelling (IMM) under the supervision of Per Christian Hansen.

We would like to thank the many people who have influenced this project in different ways. First, we would like to thank our supervisor Per Christian Hansen for support and guidance. Likewise we would like to thank Valeria Paoletti from Dipartimento di Scienze della Terra at Università di Napoli Federico II for providing us with insight into the geophysical aspects of this thesis. At IMM we would furthermore like to thank all the people in the scientific computing group. Jesper Pedersen and Toke Koldborg Jensen both former students at IMM provided us with a good insight into the existing implementation of the package and for this we are most grateful. We also want to thank Jesper and Valeria for helping us in the final stages by proofreading the thesis.

Finally we would like to thank our families and friends for support and good non-scientific company.

Lyngby, April 2007

Christian Eske Bruun & Trine Brandt Nielsen

# List of Symbols

Here, we provide some general remarks about the symbols as well as a list of commonly used notation.

| Symbol | Description |
|---|---|
| $\mathbf{A}$ | Coefficient matrix (discretized problem) |
| $a_{ij}$ | Element in the coefficient matrix |
| $B_i$ | BTTB block in $T^3$ matrix |
| $\mathbf{b}$ | Right-hand side (discretized problem) |
| $\mathbf{C}$ | Circulant matrix |
| $c_{ij}$ | Element in the circulant matrix |
| $d$ | Depth of the unknown mass distribution with density $f$ |
| $\hat{\mathbf{d}}$ | Unit vector from the dipole source towards the observation point |
| $\Delta_x$ | Grid spacing in the $x$ directions |
| $\Delta_y$ | Grid spacing in the $y$ directions |
| $\Delta_z$ | Grid spacing in the $z$ directions |
| $e$ | error/noise |
| $f(\mathbf{r})$ | Solution function = unknown distribution of mass density (or magnetization) |
| $\mathbf{F}$ | Fourier matrix |
| $F$ | The field at the source plane |
| $\hat{F}$ | Fourier transformed version of $F$ |
| $\gamma$ | Gravitational constant |

| | |
|---|---|
| $g(\mathbf{r}')$ | Right-hand side function = potential field (magnetic or gravitational) due to source |
| $h_x, h_y, h_z$ | Length of each of the segments in the solution grid |
| $h_{x'}, h_{y'}, h_{z'}$ | Length of each of the segments in the observation grid |
| $H$ | Unknown measured field |
| $\hat{H}$ | Fourier transformed version of $H$ |
| $\mathbf{I}$ | Identity matrix |
| $\hat{\mathbf{i}}$ | Unit vector with the direction of the core field |
| $\hat{\mathbf{j}}$ | Unit vector of the induced field |
| $K$ | Kernel function |
| $\mathcal{K}_k$ | Krylov subspan associated with $\mathbf{A}$ and $\mathbf{b}$ (CGLS) |
| $k$ | Truncation parameter (TSVD) |
| $\mathbf{\Lambda}$ | Diagonal matrix consisting of eigenvalues of the matrix $\mathbf{C}$ |
| $\lambda$ | Regularization parameter (Tikhonov) |
| $\mathbf{L}$ | Discrete approximation of derivative operator |
| $\mu_i$ | Singular value in SVD |
| $\mu_{\mathtt{per}}$ | Magnetic permeability |
| $\mathbf{m}$ | Solution vector (discretized problem) |
| $m$, $n$ | Matrix dimensions: $A \in \mathbb{R}^{m\times n}$ (one dimensional case) |
| $M$, $N$ | Matrix dimensions: $A \in \mathbb{R}^{M\times N}$ (two and three dimensional case) |
| $m_x, m_y, m_z$ | Number of grid points in the observations |
| $n_x, n_y, n_z$ | Number of grid points in the solution domain |
| $\mathbf{\Omega}$ | Solution domain |
| $\mathbf{r}'$ | Multi dimensional coordinate vector (observation) |
| $\mathbf{r}$ | Multi dimensional coordinate vector (solution) |
| $r'$ | Observation coordiante (1-D) |
| $r$ | Solution coordinate (1-D) |
| $\sigma_i$ | Singular value in SVE |
| $\mathbf{\Sigma}$ | Diagonal matrix containing singular values $\sigma_i$ |
| $T_i$ | Toeplitz block in BTTB matrix |
| $t_i$ | Element in a Toeplitz matrix |
| $\mathbf{u}_i$ | Left singular vector |
| $\mathbf{U}$ | $[\mathbf{u}_1, ..., \mathbf{u}_n]$ |
| $\mathbf{v}_i$ | Right singular vector |
| $\mathbf{V}$ | $[\mathbf{v}_1, ..., \mathbf{v}_n]$ |
| $w_i, w_j, w_k$ | Quadrature weights |
| $x_i, y_j, z_k,$ | Quadrature point |

| | |
|---|---|
| $x'_{i'}, y'_{j'}, z'_{k'}$ | Collocation point |
| $Y_{up}$ | Weighting function |
| $\doteq$ | Convergens in the mean |
| $\langle \cdot, \cdot \rangle$ | The usual inner product |
| $\\| \cdot \\|_2$ | 2 norm |
| $\mathbf{0}$ | Zero vector |

# Contents

Chapter 1

# Introduction

This thesis describes the algorithms and software for large-scale geophysical reconstructions with special interest in obtaining and maintaining Toeplitz structure in the models, in order to enable large-scale solution methods/algorithms. The essence of an inverse problem is described in [11] by posing a Jeopardy-like problem. The only problem being that the corresponding question is not well-defined. Consider for instance

"The answer is 4"

There are infinitely many questions that can be answered using this particular answer. Why should "What is 2+2?" be better than "How many corners of the world exist?". In the geophysical case we have the measurements but no knowledge of the data to be reconstructed. For this reason we need a mathematical model mapping the data to the measurements.

The work of this thesis is a new expanded version of the preexisting package `GravMagTools` as described in [5]. The problem with the existing implementation is that it is not possible to solve large-scale problems due to consumption of memory. In this thesis we perform a dicretization of a continuous function thereby obtaining a linear system of equations $\mathbf{Am} = \mathbf{b}$. In the left part of Figure 1.1 we illustrate the matrix $\mathbf{A}$ calculated using the preexisting package. The right part of Figure 1.1 shows that by simple column permutations we can achieve a systematic Toeplitz structure. This observation is the basis of the work of this thesis.

In the first part of the thesis the Toeplitz matrix theory that enables a com-

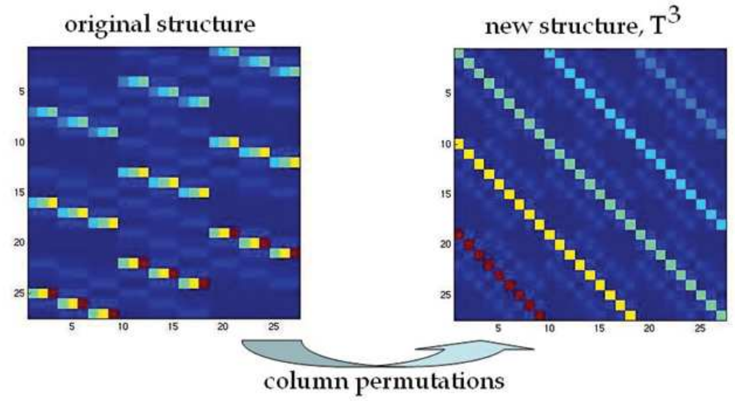


Figure 1.1: On the left-hand side the original structure in the `GravMagTools` package. On right part of the figure the new structure.

pression and the underlying theory of the geophysical problems is presented. Then an investigation of the surveying problems is conducted. We give a description of the constraints that need to be met in order to obtain Toeplitz structure in the surveying problems. This is done for gravity as well as the magnetic case in order to achieve a basic understanding of the problem.

The preexisting `GravMagTools` package uses full matrices and a straight forward multiplication. In this project we utilize the Toeplitz structures in the surveying problems to reduce the number of stored elements. Furthermore we implement a FFT matrix-vector multiplication method in the hopes that this implementation will speed up the calculations. In Chapter 4 we discuss the preprocessing of data which is needed before we can achieve the desired Toeplitz structures. We then describe the representation of data, the implementation, and the performance of the new implementation in Chapter 6.

In Chapter 5 we will present the tools that are needed to perform the inversions. These tools will be used when we in the final part of the thesis perform the actual inversions with and without topography. In this section we will perform large-scale inversions as well.

The software is enclosed on the CD in the back of this thesis. On the CD we have included demonstration scripts that illustrate some of the features of the package. Furtermore we have placed a `readme` file that explains how to install and use the package.

Chapter 2

# Toeplitz Matrix Theory

This chapter describes the underlying matrix theory we use throughout the project. Furthermore we will describe how the Toeplitz structure can be used in order to obtain a fast multiplication.

## 2.1 Toeplitz Structures

### 2.1.1 Toeplitz Matrix

A Toeplitz matrix is a matrix where all elements of the negative-sloping diagonals are identical. This special structure appears when the elements in the matrix depend entirely on the difference between the indices. The general form of an $m \times n$ Toeplitz matrix is shown in Figure 2.1 To store an $m \times n$ Toeplitz matrix it is only necessary to store $m + n - 1$ elements as opposed to $m \cdot n$ elements if the matrix does not possess any structure. This is easily realized, because all elements in a Toeplitz matrix appears in the first row and column.

- The full structure requires: $m \cdot n$.
- The Toeplitz structure requires: $m + n - 1$.

In the special case where the matrix is symmetric as well as Toeplitz it is only necessary to store the first row or column.

$$\begin{bmatrix} t_m & t_{m+1} & & \cdots & t_{m+n} \\ t_{m-1} & t_m & t_{m+1} & & \vdots \\ \vdots & t_{m-1} & t_m & \ddots & \vdots \\ \vdots & & & \ddots & t_{m+1} \\ \vdots & & & \ddots & t_m \\ \vdots & & & & \vdots \\ t_1 & & & & t_{m+n-1} \end{bmatrix}.$$

Figure 2.1: $m \times n$ Toeplitz matrix.

$$\begin{bmatrix} T_{M_{bttb}} & T_{M_{bttb}+1} & & \cdots & T_{M_{bttb}+N_{bttb}-1} \\ T_{M_{bttb}-1} & T_{M_{bttb}} & T_{M_{bttb}+1} & & \vdots \\ \vdots & T_{M_{bttb}-1} & T_{M_{bttb}} & \ddots & \vdots \\ \vdots & & T_{M_{bttb}-1} & \ddots & T_{M_{bttb}+1} \\ \vdots & & & \ddots & T_{M_{bttb}} \\ \vdots & & & & T_{M_{bttb}-1} \\ \vdots & & & & \vdots \\ T_1 & & & & T_{N_{bttb}} \end{bmatrix}.$$

Figure 2.2: BTTB matrix where $T_i$ are Toeplitz blocks and $i = 1, 2, ..., M_{bttb} + N_{bttb} - 1$.

The advantages of using the Toeplitz structures in matrices arise when operating with large matrices, because it is redundant to store all elements. If the Toeplitz structure appears in a matrix, it can be used as a way of compressing it and no information is not lost in the process. So we have in fact achieved a loss-less compression.

### 2.1.2 BTTB Matrix

BTTB is an abbreviation for Block Toeplitz with Toeplitz Blocks. This type of matrix consists of identical Toeplitz blocks in the negative-sloping diagonals. The general form of a BTTB matrix is illustrated in Figure 2.2. Figure 2.3 shows an illustration of a BTTB matrix with $100 \times 100$ elements. The main-diagonal negative-slope is highlighted in the BTTB matrix on the left-hand side in order to illustrate that all blocks in this slope are identical. The same holds for all other negative-sloping diagonals of the matrix. On the right side of the figure one of these block elements is illustrated to show that the structure is in fact a Toeplitz matrix.

The BTTB matrix consist of a number of $m \times n$ Toeplitz matrices, thus for

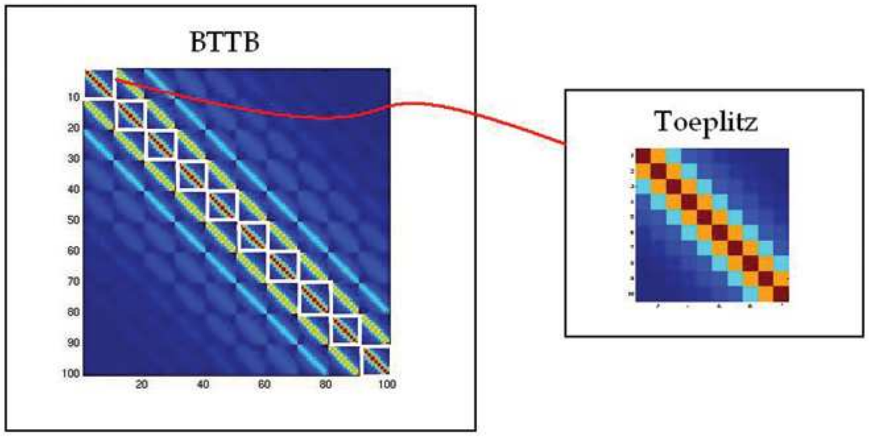


Figure 2.3: On the left-hand side a matrix with BTTB structure. On the right-hand side one of the Toeplitz blocks enlarged.

each of these Toeplitz matrices we can achieve a compression factor of $\frac{m \cdot n}{m+n-1}$. But the blocked Toeplitz structure of the BTTB matrix can also be used. All blocks in the BTTB structure appears in the first blocked-row and the first blocked-column. For this reason we only need to store $M_{BTTB} + N_{BTTB} - 1$ Toeplitz blocks as opposed to $M_{BTTB} \cdot N_{BTTB}$ blocks.

Assume that $M_{BTTB} = m$ and $N_{BTTB} = n$

- The full storage requires: $m^2 \cdot n^2$.
- The BTTB storage requires: $(m + n - 1)^2$.

### 2.1.3 $T^3$ Matrix

We now introduce yet another Toeplitz structured matrix class. These matrices consists of identical BTTB blocks in the negative-sloping diagonals. We have chosen to denote this class $T^3$ matrices. The structure of the $T^3$ matrix is predictably like the BTTB structure from Section 2.1.2 and illustrated in Figure 2.4. The illustration of a $T^3$ matrix is pictured in Figure 2.5 with a $300 \times 300$

$$\begin{bmatrix} B_{M_{t3}} & B_{M_{t3}+1} & & \cdots & B_{M_{t3}+N_{t3}-1} \\ B_{M_{t3}-1} & B_{M_{t3}} & B_{M_{t3}+1} & & \vdots \\ \vdots & B_{M_{t3}-1} & B_{M_{t3}} & \ddots & \vdots \\ \vdots & & B_{M_{t3}-1} & \ddots & B_{M_{t3}+1} \\ \vdots & & & \ddots & B_{M_{t3}} \\ \vdots & & & & B_{M_{t3}-1} \\ \vdots & & & & \vdots \\ B_1 & & & & B_{N_{t3}} \end{bmatrix}.$$

Figure 2.4: $T^3$ matrix where $B_i$ are BTTB blocks and $i = 1, 2, ..., M_{t3} + N_{t3} - 1$.

matrix. The BTTB blocks in the main diagonal are high-lighted and in the right-hand side of the figure we have enlarged one of the BTTB blocks of the diagonal. The enlarged BTTB matrix is identical to the matrix on the left hand-side of Figure 2.3. Because of the Toeplitz structure once again it is not

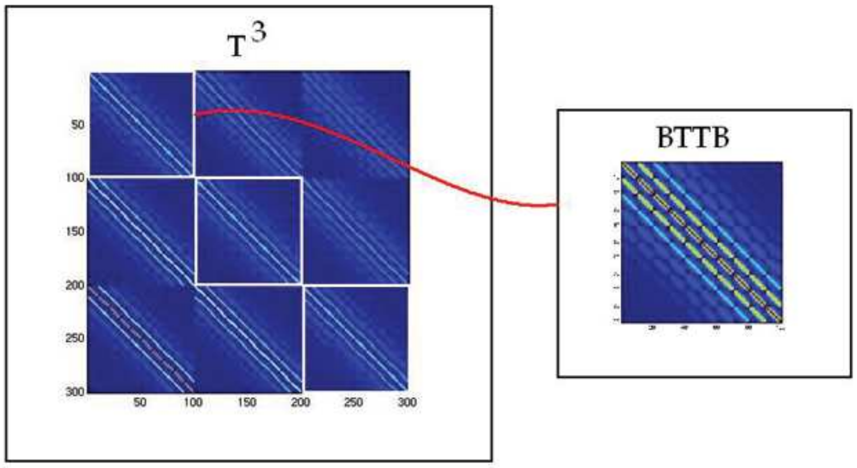


Figure 2.5: On the left-hand side a matrix with $T^3$ structure. On the right-hand side one of the blocks (a BTTB block) enlarged

necessary to store all BTTB blocks. We only need to store $M_{t3} + N_{t3} - 1$ BTTB blocks as opposed to $M_{t3} \cdot N_{t3}$ blocks.

Assume that $M_{t3} = M_{BTTB} = m$ and $N_{t3} = N_{BTTB} = n$

- The full storage requires: $m^3 \cdot n^3$.
- The BTTB storage requires: $(m + n - 1)^3$.

Using Toeplitz structures the compression is achieved without loss of information about a single element in the $T^3$ structure.

## 2.2 Circulant Matrices and FFT Multiplication

In a circulant matrix the columns are circularly shifted. Let $\mathbf{C}$ be a $n \times n$ circulant matrix then it takes the form

$$\mathbf{C} = \begin{bmatrix} c_0 & c_{n-1} & c_{n-2} & \cdots & c_1 \\ c_1 & c_0 & c_{n-1} & \cdots & c_2 \\ c_2 & c_1 & c_0 & \cdots & c_3 \\ \vdots & \vdots & \vdots & & \vdots \\ c_{n-1} & c_{n-2} & c_{n-3} & \cdots & c_0 \end{bmatrix}$$

Circulant matrices are in fact a special kind of Toeplitz structure where all elements are given in the first column. Each of the following columns are then obtained by shifting the previous column circularly. We consider the matrix-vector multiplication using FFT (Fast Fourier Transform). According to [7] this operation can be performed in $O(n\texttt{log}_2 n)$ flops (for an $n \times n$ matrix) as opposed to $2n^2$ flops for a general matrix-vector multiplication. In this section we will consider the matrix-vector multiplication $\mathbf{Ax} = \mathbf{y}$. The matrix-vector multiplication by FFT is described in [7]. The basic idea is that the $m \times n$ Toeplitz matrix, $\mathbf{A}$, is embedded into a larger $p \times p$ circulant matrix $\mathbf{C}$. In the following we have chosen $p = m + n$. In Figure 2.6 a $p \times p$ circulant matrix is illustrated, the embedded Toeplitz matrix is seen in the upper left corner. The elements of the matrix marked with a red circle are the free elements that can assume any value.

In order to perform the multiplication we construct the first column of $\mathbf{C}$

$$\mathbf{C}(:, 1) = [\mathbf{A}(1, 1),\ \mathbf{A}(2, 1), ..., \mathbf{A}(n, 1),\ 0,\ \mathbf{A}(1, 2), ..., \mathbf{A}(1, m)]^T\ .$$

In this context the free element is represented by a single zero.

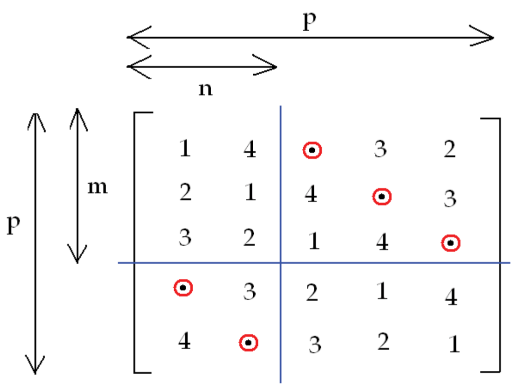


Figure 2.6: An example of the $m \times n = 3 \times 2$ Toeplitz matrix (upper left corner) embedded in the $p \times p = 5 \times 5$ circulant matrix.

Given the $m \times n$ Toeplitz matrix $\mathbf{A}$ (also represented in Figure 2.3) we have $\mathbf{C}(:,1)$

$$\mathbf{A} = \begin{pmatrix} 1 & 4 \\ 2 & 1 \\ 3 & 2 \end{pmatrix} \quad \rightarrow \quad \mathbf{C}(:,1) = (1,\, 2,\, 3,\, 0,\, 4)^T \ .$$

In [7] it is shown that $\mathbf{C}$ can be factorized as:

$$\mathbf{C} = \mathbf{F}^{-1} \boldsymbol{\Lambda} \mathbf{F} \ . \tag{2.1}$$

where $\mathbf{F}$ is the $p \times p$ Fourier matrix (a complex symmetric matrix) and $\boldsymbol{\Lambda}$ is a diagonal matrix with the eigenvalues of $\mathbf{C}$.

When calculating the product of $\mathbf{C}$ with $\hat{\mathbf{x}} = \begin{pmatrix} \mathbf{x} \\ \mathbf{0} \end{pmatrix}$ we get

$$\hat{\mathbf{y}} = \mathbf{C}\hat{\mathbf{x}} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \hat{\mathbf{B}} & \hat{\mathbf{A}} \end{pmatrix} \hat{\mathbf{x}} = \begin{pmatrix} \mathbf{A}\mathbf{x} \\ \hat{\mathbf{y}}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{y} \\ \hat{\mathbf{y}}_2 \end{pmatrix} ,$$

where $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ are not necessarily identical to $\mathbf{A}$ and $\mathbf{B}$, respectively, in fact in any non-square system they are not identical. $\mathbf{y}$ is the real data from the first $n$ components of $\hat{\mathbf{y}}$.

When multiplying $\mathbf{A}^T$ with $\mathbf{y}$ (in order to achieve $\mathbf{A}^T\mathbf{y} = \mathbf{z}$)

$$\hat{\mathbf{z}} = \mathbf{C}^T\hat{\mathbf{y}} = \begin{pmatrix} \mathbf{A}^T & \hat{\mathbf{B}}^T \\ \mathbf{B}^T & \hat{\mathbf{A}}^T \end{pmatrix} \hat{\mathbf{y}} = \begin{pmatrix} \mathbf{A}^T\mathbf{y} \\ \hat{\mathbf{z}}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{z} \\ \hat{\mathbf{z}}_2 \end{pmatrix} \ ,$$

where $\mathbf{C} = \mathbf{F}^{-1}\mathbf{\Lambda}\mathbf{F} \Leftrightarrow \mathbf{C}^T = \mathbf{F}^T\mathbf{\Lambda}^T\mathbf{F}^{-T}$

$\mathbf{F}^T$ is symmetric complex thus $\mathbf{F}^T = \mathbf{F}$, and hence

$$\mathbf{C}^T = \mathbf{F}\mathbf{\Lambda}\mathbf{F}^{-1} \ .$$

We know from [7] that $\mathbf{F}^{-1} = \frac{1}{p}\bar{\mathbf{F}}$ where $\bar{\mathbf{F}}$ is the notation for the complex conjugated and $p = m + n$. Thus

$$\begin{aligned} &\mathbf{F}^{-1} = \tfrac{1}{p}\bar{\mathbf{F}} = \tfrac{1}{p}\bar{\mathbf{F}}^T \\ &\Updownarrow \\ &\mathbf{F} = (\tfrac{1}{p}\bar{\mathbf{F}}^T)^{-1} = p\bar{\mathbf{F}}^{-T} \\ &\Updownarrow \\ &\bar{\mathbf{F}}^{-T} = \tfrac{1}{p}\mathbf{F} \ . \end{aligned}$$

Hence

$$\begin{aligned} \mathbf{C}^T &= \bar{\mathbf{C}}^T = \bar{\mathbf{F}}^T\bar{\mathbf{\Lambda}}^T\bar{\mathbf{F}}^{-T} \\ &= p\mathbf{F}^{-1}\bar{\mathbf{\Lambda}}^T(\frac{1}{p}\mathbf{F}) \\ &= p\mathbf{F}^{-1}\bar{\mathbf{\Lambda}}(\frac{1}{p}\mathbf{F}) \\ &= \mathbf{F}^{-1}\bar{\mathbf{\Lambda}}\mathbf{F} \ . \end{aligned} \tag{2.2}$$

When comparing the factorization of $\mathbf{C}$ (Equation (2.1)) and the factorization of $\mathbf{C}^T$ (Equation (2.2)) it becomes clear that the only difference is the conjugation of $\mathbf{\Lambda}$.

### 2.2.1 Pseudocode

In the following we present pseudocode for $\mathbf{A} * \mathbf{x}$ and $\mathbf{A}^T * \mathbf{y}$ using FFT matrix-vector multiplication

Pseudocode for $\mathbf{A} * \mathbf{x} = \mathbf{y}$ using FFT matrix-vector multiplication:

```
$\lambda = \text{fft}(\mathbf{C}(:,1));$
$\hat{\mathbf{x}} = \text{fft}\begin{pmatrix}\mathbf{x}\\ \mathbf{0}\end{pmatrix};$
$\hat{\mathbf{y}} = \lambda. * \hat{\mathbf{x}};$
$\mathbf{y} = \text{ifft}(\hat{\mathbf{y}});$
$\mathbf{y} = \text{real}(\mathbf{y}(1:n))$
```

Pseudocode for $\mathbf{A}^T * \mathbf{y} = \mathbf{z}$ using FFT matrix-vector multiplication:

```
$\lambda = \text{fft}(\mathbf{C}(:,1));$
$\hat{\mathbf{y}} = \text{fft}\begin{pmatrix}\mathbf{y}\\ \mathbf{0}\end{pmatrix};$
$\hat{\mathbf{z}} = \texttt{conj}(\lambda). * \hat{\mathbf{y}};$
$\mathbf{z} = \text{ifft}(\hat{\mathbf{z}});$
$\mathbf{z} = \text{real}(\mathbf{z}(1:n))$
```

Considering the two pseudocodes it is clear how the only difference (apart from the naming of the variables) is the conjugation of the eigenvalues.

Chapter 3

# The Surveying Problems

In this thesis we consider geophysical problems more precisely the gravity and magnetic surveying problems. We consider the gravity problem in one, two, and three dimensions, respectively. For the magnetic surveying problem we will only consider the three dimensional problem.

This project has special interest in achieving and maintaining the Toeplitz structures. Hence, we will carefully cover the constraints that need to be met in order to achieve the structures as covered in Chapter 2. However, we will first describe the Fredholm integral equation of the first kind which provides a model for our surveying problems.

## 3.1 The First-kind Fredholm Integral Equation

The generic model for the geophysical problems is illustrated in Figure 3.1. The figure shows the relationship between the hidden data that we wish to reconstruct and the measurements. Knowing the hidden source (hidden data in Figure 3.1), the measurements can be found using a forward calculation. However, in geophysical problems the solution is rarely known. The challenge is then: given the measurements to reconstruct an inverse problem in order to find the hidden source. We face the inverse problem of computing properties of

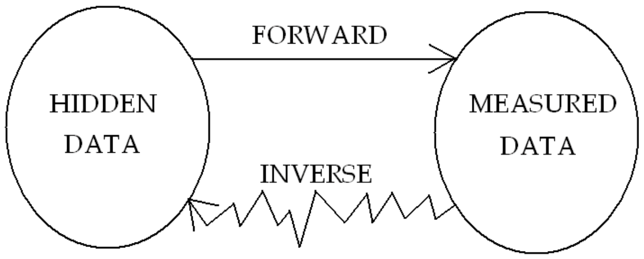


Figure 3.1: The generic model.

the interior of a domain given measurements taken outside the domain and a mathematical model for their interrelation. The model provides the relationship between the measurements and the interior of the domain. This relationship can be described by a Fredholm integral equation of the first kind.

The Fredholm integral equation of the first kind in one dimension is given as follows

$$\int_0^1 K(r',r)f(r)dr = g(r'), \qquad a \leq r' \leq b \ , \tag{3.1}$$

where $g(r')$ is the observation at point $r'$ and $f(r)$ describes the physical quantity we are trying to reconstruct at $r$. $K(r',r)$ is called the kernel and it is a function that depends on the geometric placement of observation point $r'$ and the source point $r$.

In multiple dimensions the Fredholm integral equation of the first kind takes the form

$$\int_{\mathbf{\Omega_r}} K(\mathbf{r}',\mathbf{r})f(\mathbf{r})d\mathbf{r} = g(\mathbf{r}'), \qquad \mathbf{r}' \in \mathbf{\Omega}_{\mathbf{r}'} \ . \tag{3.2}$$

The Fredholm integral equation we deal with has the special property that the kernel only depends on the difference between $\mathbf{r}'$ and $\mathbf{r}$ which implies that $K(\mathbf{r}',\mathbf{r}) = k(\mathbf{r}'-\mathbf{r})$ where $k$ is some function.

The Fredholm integral equation of the first kind can be discretized so it can be solved numerically. The equation is discretized using the midpoint quadrature method. Note that $f$ is now substituted with $\tilde{f}$ because $f$ can not be calculated exactly.

$$\sum_{j=1}^{n} w_j K(\mathbf{r}'_i,\mathbf{r}_j)\tilde{f}(\mathbf{r}_j) = g(\mathbf{r}'_i) \ , \qquad i = 1,...,m \ . \tag{3.3}$$

where $\mathbf{r}'_1, \mathbf{r}'_2, ..., \mathbf{r}'_m$ are the collocation points and $\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_n$ are the abscissas for the midpoint quadrature rule. In matrix notation

$$\begin{pmatrix} K(w_1\mathbf{r}_1, \mathbf{r'}_1) & w_2K(\mathbf{r}_1, \mathbf{r'}_2) & ... & w_nK(\mathbf{r}_1, \mathbf{r'}_n) \\ K(w_1\mathbf{r}_2, \mathbf{r'}_1) & w_2K(\mathbf{r}_2, \mathbf{r'}_2) & ... & w_nK(\mathbf{r}_2, \mathbf{r'}_n) \\ \vdots & \vdots & \ddots & \vdots \\ K(w_1\mathbf{r}_m, \mathbf{r'}_1) & w_2K(\mathbf{r}_m, \mathbf{r'}_2) & ... & w_nK(\mathbf{r}_m, \mathbf{r'}_n) \end{pmatrix} \begin{pmatrix} \tilde{f}(\mathbf{r}_1) \\ \tilde{f}(\mathbf{r}_2) \\ \vdots \\ \tilde{f}(\mathbf{r}_n) \end{pmatrix} = \begin{pmatrix} g(\mathbf{r'}_1) \\ g(\mathbf{r'}_2) \\ \vdots \\ g(\mathbf{r'}_m) \end{pmatrix} .$$

In simple notation

$$\mathbf{Am} = \mathbf{b} \ . \tag{3.4}$$

In the above linear system of equations $\mathbf{A}$ is the coefficient matrix, $\mathbf{m}$ is the solution, and $\mathbf{b}$ is the right hand-side.

In this project the linear inverse problems we consider are by nature ill-posed. A system of equations is considered to be ill-posed if a small change in the coefficient matrix or a small change in the right hand side results in large changes in the solution vector. There are many examples of physical problems that lead to this form. A couple of these problems are described in the following.

## 3.2 The Gravity Surveying problem

The gravity surveying problem is based on studying anomalies in the gravity field. The anomaly is caused by contrast of density under ground.

### 3.2.1 The One dimensional Case

We consider the Fredholm integral equation of the first kind in one dimension. In the following we substitute $r$ with $x$ and $r'$ with $x'$. The measured signal (the right hand-side of the equation) will then be a function of $x'$ and the mass distribution is a function of $x$. First we will shortly describe the geometry of the gravity surveying problem in one dimension.

In Figure 3.2 the measured signal $g(x')$ is the vertical component of the gravity field and it is illustrated by a blue arrow. The measuring points are in the interval between $x'_{start}$ and $x'_{end}$. The mass distribution $f(x)$ which causes the gravity field is placed at the depth $d$ from 0 to 1 on the $x$ axis[1]. The kernel $K$ in the problem is derived in [8] and it is given by

$$K(x', x) = \gamma \frac{d}{(d^2+(x'-x)^2)^{\frac{3}{2}}} \ ,$$

[1]The figure is slightly modified, but taken from [7] Figure 1 pp. 327.

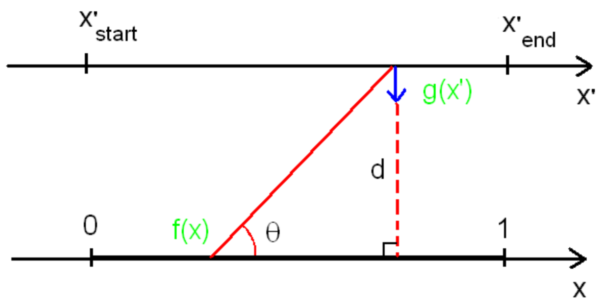


Figure 3.2: The geometry of the gravity surveying model problem. The measured signal $g(x')$ is the vertical component of the gravity field due to a 1-D mass distribution $f(x)$ at depth $d$.

where $\gamma$ is the gravitational constant ($\gamma = 6.673 * 10^{-11} \frac{\mathtt{m}^3}{\mathtt{kg}\cdot\mathtt{s}^2}$). We assume that $d > 0$ in order to ensure that we do not divide by 0 in the model.[2]. The problem is discretized so it can be reconstructed using a computer. $x'$ is divided in to $m$ points and $x$ in to $n$ points. The matrix of the discretized problem is denoted $\mathbf{A}$ and to derive this we make use of the fact that the elements in $\mathbf{A}$ are given by

$$a_{i'i} = w_i K(x'_{i'}, x_i) \ , \qquad i' = 1, 2, ..., m \text{ and } i = 1, 2, ..., n \ .$$

The quadrature weight $w_i$ is calculated by using the midpoint quadrature rule

$$w_i = \frac{|1-0|}{n} = \frac{1}{n} \ .$$

The midpoint quadrature rule is also used to find the quadrature points

$$x_i = h_x i - \frac{h_x}{2} \ , \qquad h_x = \frac{1}{n} \ .$$

We choose the collocation points $x'_{i'}$ to be

$$x'_{i'} = x'_{start} + h_{x'} i' - \frac{h_{x'}}{2}, \qquad \text{where } h_{x'} = \frac{|x'_{end} - x'_{start}|}{m} \ .$$

[2]This assumption also applies to the 2-D and 3-D gravity surveying models.

Hence, the matrix elements $a_{i'i}$ are

$$\begin{aligned}
a_{i'i} &= w_i \gamma \frac{d}{(d^2 + (x'_{i'} - x_i)^2)^{\frac{3}{2}}} \\
&= \frac{\gamma}{n} \frac{d}{(d^2 + (x'_{start} + h_{x'} i' - \frac{h_{x'}}{2} - (h_x i - \frac{h_x}{2}))^2)^{\frac{3}{2}}} \\
&= \frac{\gamma}{n} \frac{d}{(d^2 + (x'_{start} - \frac{h_{x'}}{2} + \frac{h_x}{2} + (h_{x'} i' - h_x i))^2)^{\frac{3}{2}}} \ .
\end{aligned} \qquad (3.5)$$

In order to achieve Toeplitz structure, the matrix $\mathbf{A}$ must only depend on the differences between the indices. This implies that $a_{i'i} = a_{i'+k,i+k}$ for all allowed values of $k$. When considering Equation (3.5) it is clear that this is the case if $h_{x'} i' - h_x i = i' - i$ and hence

$$\begin{aligned}
\frac{|x'_{end} - x'_{start}|}{m} i' - \frac{1}{n} i &= i' - i \quad \Leftrightarrow \\
\frac{|x'_{end} - x'_{start}|}{m} &= \frac{1}{n} \quad \Leftrightarrow \\
\frac{1}{|x'_{end} - x'_{start}|} &= \frac{n}{m} \ .
\end{aligned}$$

It means that the ratio between $n$ and $m$ must be equal to the ratio between 1 and $|x'_{end} - x'_{start}|$. In the special case where $n = m$ the length of $|x'_{end} - x'_{start}|$ is consequently equal to 1. However, it is of no importance where the interval from $x'_{start}$ to $x'_{end}$ is placed on the $x'$ axis. The depth of the mass distribution $d$ does not affect the structure of the matrix $\mathbf{A}$. If matrix $\mathbf{A}$ is a symmetric Toeplitz structure the following condition must be met

$$a_{ii'} \ = \ a_{i'i} \Leftrightarrow$$

$$\begin{aligned}
\frac{\gamma}{n} \frac{d}{(d^2 + (x'_{start} - \frac{h_{x'}}{2} + \frac{h_x}{2} + (h_{x'} i' - h_x i))^2)^{\frac{3}{2}}} &= \\
\frac{\gamma}{n} \frac{d}{(d^2 + (x'_{start} - \frac{h_{x'}}{2} + \frac{h_x}{2} + (h_x i - h_{x'} i'))^2)^{\frac{3}{2}}} & \ .
\end{aligned}$$

It follows that $a_{ii'} = a_{i'i}$ if $i = i'$. In the case where $i \neq i'$ then $x'_{start} - \frac{|x'_{end} - x'_{start}|}{2m} + \frac{1}{2n} = 0$ and $\frac{|x'_{end} - x'_{start}|}{m} = \frac{1}{n}$. This can only be the case if three different properties are met, namely:

$$\begin{aligned}
|x'_{end} - x'_{start}| &= 1 \\
m &= n \ . \\
x'_{start} &= 0 \ .
\end{aligned}$$

Thus the two intervals must be the same length and discretized into the same numbers of points. The last property implies that the two intervals must be placed directly above each other.

Now we will continue the theory for the two dimensional case.

### 3.2.2 The Two dimensional Case

The geometry of the gravity surveying problem in 2-D is slightly more complicated than in 1-D, but the general idea is the same. The measured signal is now two dimensional and we have chosen to denote $g(\mathbf{r}') = g(x', y')$. The measured signal $g(x', y')$ is the vertical component of the gravity field and the measuring points are in the interval $[x'_{start}\ x'_{end}] \times [y'_{start}\ y'_{end}]$. The mass distribution which causes the gravity field $f(\mathbf{r}) = f(x, y)$ is placed at depth $d$ in the interval $[0\ 1] \times [0\ 1]$. The geometry is illustrated in Figure 3.3.

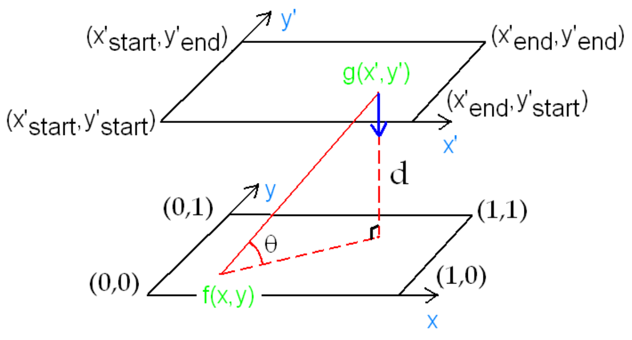


Figure 3.3: The geometry of the gravity surveying model problem in 2-D. The measured signal $g(x', y')$ is the vertical component of the gravity field due to a mass distribution $f(x, y)$ at depth $d$.

In the 2-D case the kernel takes the form:

$$K(x', y', x, y) = \gamma \frac{d}{(d^2 + (x'-x)^2 + (y'-y)^2)^{\frac{3}{2}}} \ .$$

The $x$ and $y$ intervals are discretized into $n$ points and the $x'$ and $y'$ into $m$ points. To derive a matrix $\mathbf{A}$ for the discretized problem we utilize that the elements now are given by

$$a_{MN} = \gamma w_i w_j K(x_i, y_j, x'_{i'}, y'_{j'}) \ , \qquad i, j = 1, 2, ..., n \ , \quad i', j' = 1, 2, ..., m \ ,$$

where $M = (j' - 1)\, m + i'$, $\quad N = (i - 1)\, n + j$. The quadrature weights using the midpoint quadrature rule are

$$w_i = \tfrac{1}{n} \text{ and } w_j = \tfrac{1}{n} \ .$$

The quadrature points are

$$\begin{aligned} x_i &= ih_x - \tfrac{h_x}{2}, \qquad & h_x &= \tfrac{1}{n} \ . \\ y_j &= jh_y - \tfrac{h_y}{2}, \qquad & h_y &= \tfrac{1}{n} \ . \end{aligned}$$

The collocation points we choose as

$$\begin{aligned} x'_{i'} &= x'_{start} + i'h_{x'} - \tfrac{h_{x'}}{2}, \qquad & h_{x'} &= \tfrac{|x'_{end} - x'_{start}|}{m} \\ y'_{j'} &= y'_{start} + j'h_{y'} - \tfrac{h_{y'}}{2}, \qquad & h_{y'} &= \tfrac{|y'_{end} - y'_{start}|}{m} \ . \end{aligned}$$

Thereby we can formulate the matrix elements $a_{MN}$

$$a_{MN} \quad = \quad \frac{\gamma}{n^2} \frac{d}{Q_{ij}^{MN}} \ , \tag{3.6}$$

where
$Q_{ij}^{MN} = (d^2 + (x'_{i'} - x_i)^2 + (y'_{j'} - y_j)^2)^{\frac{3}{2}}$
$= (d^2 + (x'_{start} + h_{x'}i' - \frac{h_{x'}}{2} - (h_x i - \frac{h_x}{2}))^2 + (y'_{start} + h_{y'}j' - \frac{h_{y'}}{2} - (h_y j - \frac{h_y}{2}))^2)^{\frac{3}{2}}$
$= (d^2 + (x'_{start} - \frac{h_{x'}}{2} + \frac{1}{2n} + (h_{x'}i' - h_x i))^2 + (y'_{start} - \frac{h_{y'}}{2} + \frac{1}{2n} + (h_{y'}j' - h_y j))^2)^{\frac{3}{2}}$ .

In order for **A** to achieve a BTTB structure the following conditions must be met

$$\begin{aligned} \frac{|x'_{end} - x'_{start}|}{m_x} i' - \frac{1}{n} i \quad &= \quad i' - i \Leftrightarrow \\ F \frac{n}{m_x} \quad &= \quad \frac{1}{|x'_{end} - x'_{start}|} \ , \end{aligned}$$

and similarly in the $y$-direction. As for the 1-D case, the placement of the intervals $[x'_{end}\ x'_{start}]$ and $[y'_{end}\ y'_{start}]$ are of no importance to the BTTB structure. Furthermore we notice that the depth $d$ has no effect on the structure of **A**.

The matrix **A** will be symmetric if the symmetry properties listed for the one dimensional case are satisfied. When using real data the properties are rarely met - especially the property which requires that the intervals to be placed directly above each other can be difficult. Due to this we decide to assume that **A** will not have a symmetric BTTB structure, but simply a BTTB structure.

### 3.2.3 The Three dimensional Case

For this case we will derive the general formulation. In three dimensions we once again consider the geometry which can be used to reconstruct the underlying problem. The mass causing the gravity field is placed in a volume from $[x_{start}\ x_{end}] \times [y_{start}\ y_{end}] \times [z_{start}\ z_{end}]$. In this case $g(\mathbf{r}')$ is substituted with $g(x', y', z')$ and the mass distribution is substituted with $f(x, y, z)$. The kernel $K$ is

$$K(x', y', z', x, y, z) = \gamma \frac{z' - z}{((z'-z)^2 + (x'-x)^2 + (y'-y)^2)^{\frac{3}{2}}} \ .$$

In the three dimensional case the numerator of the fraction takes into account the depth. The model consists of a measurement grid and a solution grid which are both three dimensional. The solution grid is discretized into $n_x \times n_y \times n_z$ grid points and the measurement volume into $m_x \times m_y \times m_z$ points. The geometry is illustrated in Figure 3.4.

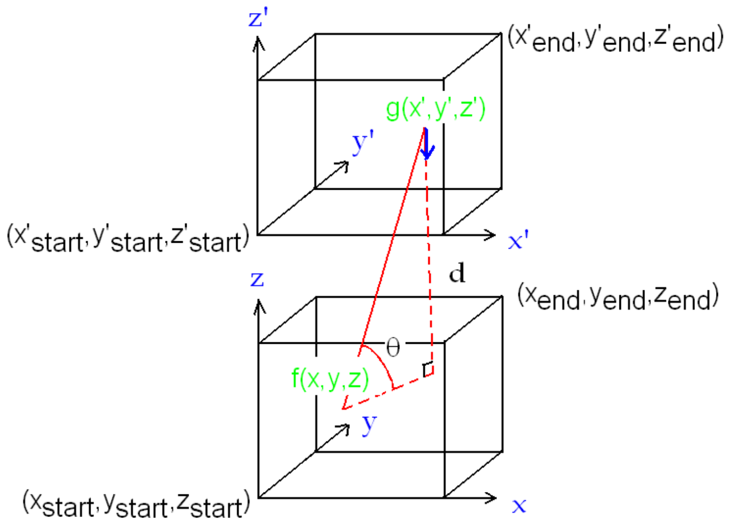


Figure 3.4: The geometry of the gravity surveying model in 3-D. The measured signal $g(x', y', z')$ is the vertical component of the gravity field due to a mass distribution $f(x, y, z)$ at depth $d$.

In the 3-D geometry the elements in $\mathbf{A}$ are given by

$$a_{MN} = w_i w_j w_k K(x_i, y_j, z_k, x'_{i'}, y'_{j'}, z'_{k'}) \ ,$$
$$i = 1, 2, ..., n_x, j = 1, 2, ..., n_y, k = 1, 2, ..., n_z \ ,$$
$$i' = 1, 2, ..., m_x, j' = 1, 2, ..., m_y, k' = 1, 2, ..., m_z \ ,$$

where $M = i' + (j'-1)m_x + (k'-1)m_x m_y$ and $N = i + (j-1)n_x + (k-1)n_x n_y$. The quadrature weights using the midpoint quadrature rule are

$$w_i = \frac{|x_{end} - x_{start}|}{n_x}, \ w_j = \frac{|y_{end} - y_{start}|}{n_y}, \ w_k = \frac{|z_{end} - z_{start}|}{n_z} \ .$$

The quadrature points are

$$x_i = x_{start} + ih_x - \frac{h_x}{2}, \qquad h_x = \frac{|x_{end} - x_{start}|}{n_x} \ .$$
$$y_j = y_{start} + jh_y - \frac{h_y}{2}, \qquad h_y = \frac{|y_{end} - y_{start}|}{n_y} \ .$$
$$z_k = z_{start} + kh_z - \frac{h_z}{2}, \qquad h_z = \frac{|z_{end} - z_{start}|}{n_z} \ .$$

The collocation points are

$$x'_{i'} = x'_{start} + i'h_{x'} - \frac{h_{x'}}{2}, \qquad h_{x'} = \frac{|x'_{end} - x'_{start}|}{m_x} \ .$$
$$y'_{j'} = y'_{start} + j'h_{y'} - \frac{h_{y'}}{2}, \qquad h_{y'} = \frac{|y'_{end} - y'_{start}|}{m_y} \ .$$
$$z'_{k'} = z'_{start} + k'h_{z'} - \frac{h_{z'}}{2}, \qquad h_{z'} = \frac{|z'_{end} - z'_{start}|}{m_z} \ .$$

The elements in $\mathbf{A}$ is in this model given by

$$a_{MN} = \gamma w_i w_j w_k \frac{z'_{start} - z_{start} + \frac{h_z}{2} - \frac{h_{z'}}{2} + (h_{z'}k' - h_z k)}{Q_{ij}^{MN}} \ ,$$

where
$Q_{ij}^{MN} = ((z'_{start} - z_{start} + \frac{h_z}{2} - \frac{h_{z'}}{2} + (h_{z'}k' - h_z k))^2 + (x'_{start} - x_{start} + \frac{h_x}{2} - \frac{h_{x'}}{2} + (h_{x'}i' - h_x i))^2 + (y'_{start} - y_{start} + \frac{h_y}{2} - \frac{h_{y'}}{2} + (h_{y'}j' - h_y j))^2)^{\frac{3}{2}}$ .

In this model both the solution and the observation grids are three dimensional which, under suitable conditions, causes $\mathbf{A}$ to have a blocked Toeplitz structure where the blocks are BTTB matrices. This is what we in Chapter 2.1.3 denoted a $T^3$ structure. An example of a $T^3$ structure is plotted in Figure 3.5. In the figure the clearly visible blocks consists of a number of Toeplitz blocks.

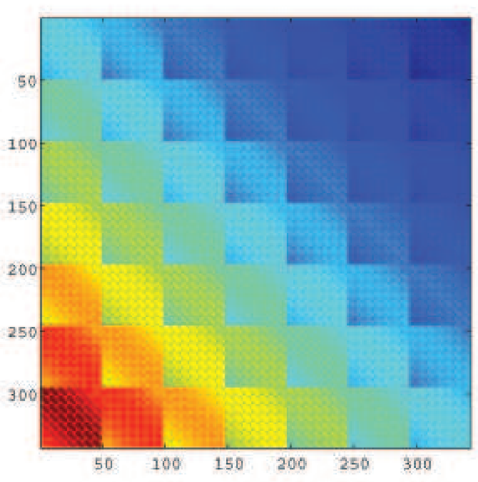


Figure 3.5: A matrix with $T^3$ structure.

If the following properties hold for **A** then the matrix will have a $T^3$ structure

$$\begin{aligned} \frac{|x'_{end} - x'_{start}|}{m_x} i' - \frac{|x_{end} - x_{start}|}{n_x} i &= i' - i \Leftrightarrow \\ \frac{n_x}{m_x} &= \frac{|x_{end} - x_{start}|}{|x'_{end} - x'_{start}|} . \end{aligned}$$

Similarly we have:

$$\frac{n_y}{m_y} = \frac{|y_{end} - y_{start}|}{|y'_{end} - y'_{start}|}, \qquad \frac{n_z}{m_z} = \frac{|z_{end} - z_{start}|}{|z'_{end} - z'_{start}|} .$$

When storing any $T^3$ structure we need to store:[3] $(n_x + m_x - 1) \cdot (n_y + m_y - 1) \cdot (n_z + m_z - 1)$ elements.

A special case of the 3-D model is where only one level of observations is used as illustrated in Figure 3.6. A model like this would result in a blockwise BTTB structure which is a subclass of the $T^3$ structure. An example of a blockwise BTTB structure is illustrated in Figure 3.7. When considering only one level of observations $m_z = 1$ thus we only need to store $(n_x + m_x - 1) \cdot (n_y + m_y - 1) \cdot n_z$ elements. However, the system should preferably be square as opposed to an underdetermined system. For this reason $m_x$ and $m_y$ can not be the same in the

[3]We postpone the explanation for Chapter 6.1.1.

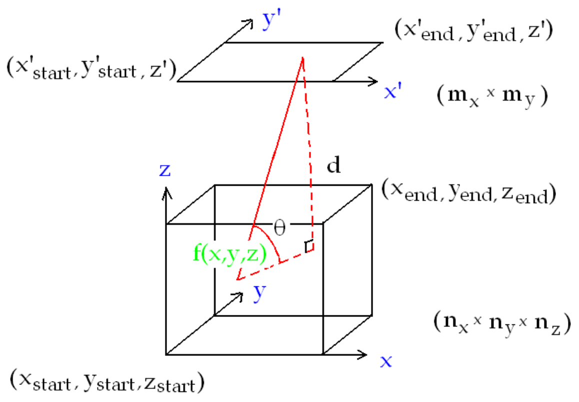


Figure 3.6: The geometry of the gravity surveying model problem in three dimensions. The measured signal $g(x', y', z')$ is the vertical component of the gravity field due to a mass distribution $f(x, y, z)$ at depth $d$.

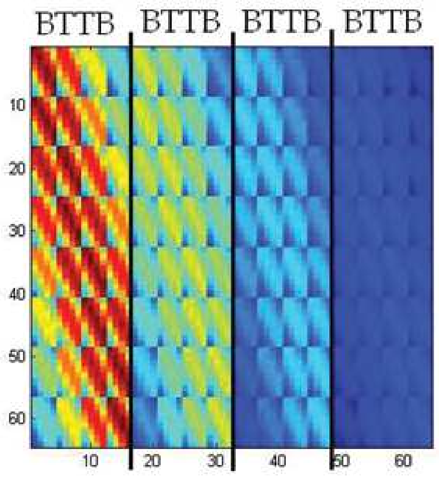


Figure 3.7: Blockwise BTTB structure. $m_x = m_y = 8$, $n_x = n_y = n_z = 4$.

two cases. In Figure 3.8 we have illustrated the two different setups for a specific example. Table 3.1 lists the numbers of elements needed to store for a setup where $n_x = n_y = m_x = m_y = 100$ points and $n_z = m_z = 36$ levels for the $T^3$ structure. To store the model using only one observation level $n_x = n_y = 100$,

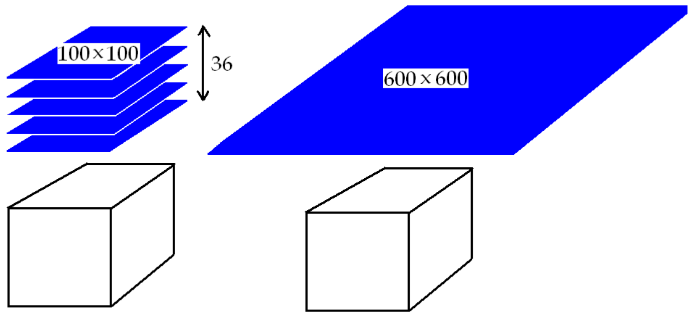


Figure 3.8: A specific example of the two kinds of setup of the 3-D geometry.

$m_x = m_y = 600$, $n_z = 36$, and $m_z = 1$. We have also listed the number of

| Structure | No. elements (example) |
|---|---|
| $T^3$ | $\sim 2.8 \cdot 10^6$ |
| Blockwise BTTB | $\sim 1.8 \cdot 10^7$ |
| No structure | $\sim 1.3 \cdot 10^{11}$ |

Table 3.1: Comparison of the storage of elements for the different structures.

elements to be stored if there is no structure in the matrix. When considering the numbers in the table we see that data compression would be a good idea. The most beneficial structure would be the $T^3$ structure or the blockwise BTTB.

A disadvantage of the $T^3$ structure is that the observation data need to be in several levels. When dealing with real data it can be a problem getting multi-level data in fact all sets that we have received have been a single level of data. In Chapter 4 we return to how we achive multi-level data.

## 3.3 The Magnetic Surveying Problem

In this section we will explain the formulation of the magnetic surveying model and the assumptions made in order to achieve the Toeplitz structure. We will for the magnetic surveying model only consider the three dimensional case.

The magnetic field measured above the Earth is the sum of different fields. In this project we assume that all magnetic fields originate from the interior of the

Earth and consists of the sum of the core field and the crust field. According to [9] only a small fraction of the measured magnetic field originates from outside the Earth. The core of the Earth generates a field which is well defined all over the world. The crust field arises because of the presence of geomagnetic material e.g. rocks containing iron which can be located several kilometers under ground. The crust field can locally have a different direction than the core field and therefore gives an anomaly in the core field. This anomaly will be referred to as the total field anomaly and it is this anomaly we are interested in. We describe the magnetization by using a dipole formulation. This implies that the magnetization consists of small magnetic dipoles. Each of these dipoles can have a individual direction, but for reasons of simplicity we assume that they all have the same direction. A magnetic dipole induces a magnetic field, which is shown in Figure 3.9, in the figure it is illustrated how the field will change around the dipole. The strength of the magnetic field weakens the further away from the dipole the measurements are performed. The kernel $K$ for the magnetic surveying model is derived in [5] and is given by the following formulation

$$K(\mathbf{r}, \mathbf{r}') = \frac{\mu_{\texttt{per}}}{4\pi} \frac{\hat{\mathbf{j}} \cdot (3(\hat{\mathbf{i}} \cdot \hat{\mathbf{d}})\hat{\mathbf{d}} - \hat{\mathbf{i}})}{\|\mathbf{r} - \mathbf{r}'\|_2^3}, \quad \hat{\mathbf{d}} = \frac{\mathbf{r} - \mathbf{r}'}{\|\mathbf{r} - \mathbf{r}'\|_2} \ . \tag{3.7}$$

In the above equation $\mathbf{r}$ is the location of a magnetic dipole source and given by $\mathbf{r} = \begin{bmatrix} x \ y \ z \end{bmatrix}^T$, $\mathbf{r}' = \begin{bmatrix} x' \ y' \ z' \end{bmatrix}^T$ is an observation point. $\hat{\mathbf{i}} = \begin{bmatrix} i_x \ i_y \ i_z \end{bmatrix}^T$ is a unit vector with the direction of the core field induced by the Earth. $\hat{\mathbf{j}} = \begin{bmatrix} j_x \ j_y \ j_z \end{bmatrix}^T$ is a unit vector with the direction induced by the dipole source in $\mathbf{r}$ and $\hat{\mathbf{d}}$ is a unit vector pointing in the direction from $\mathbf{r}$ toward $\mathbf{r}'$. $\mu_{\texttt{per}}$ is the magnetic permeability and is equal to $4\pi \times 10^{-7} N \cdot A^{-2}$. Figure 3.10 gives a graphical illustration of the model. The components $K(\mathbf{r}, \mathbf{r}')$ in the kernel is the magnetic field at $\mathbf{r}$ along the direction of $\hat{\mathbf{i}}$ due to the dipole source located in $\mathbf{r}'$ with a unit intensity and the direction $\hat{\mathbf{j}}$. In order to use equation (3.7) we must assume that the observation points are placed outside the solution domain, otherwise the magnetic field would have a rotation and the model would be incorrect. $\hat{\mathbf{i}}$ and $\hat{\mathbf{j}}$ can be expressed in the terms of the angels inclination and declination in the Spherical coordinate system. Throughout this thesis the inclination will be 90 degrees and declination 0 degrees unless otherwise is stated.

We discretize the problem and derive the conditions which has to be met before the kernel obtains the $T^3$ structure. The details of the discretization and deduction are listed in appendix A. As it turns out the conditions to be met to obtain $T^3$ structure are the same as for the gravity surveying model in 3-D,

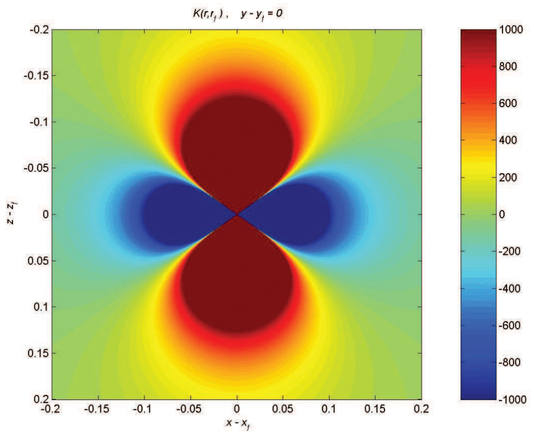


Figure 3.9: Illustration of a cross section of a dipole locate in (0,0,0). In the plot the $y$ coordinate is fixed to 0.

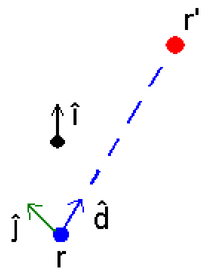


Figure 3.10: The geometry of the magnetic surveying model where $\hat{\mathbf{j}}$ is the unit vector in the direction of the magnetization in $\mathbf{r}$. $\hat{\mathbf{i}}$ is the unit vector with the direction of the core field and $\hat{\mathbf{d}}$ is the unit vector from $\mathbf{r}$ towards $\mathbf{r}'$.

namely:

$$\frac{n_x}{m_x} = \frac{|x_{end} - x_{start}|}{|x'_{end} - x'_{start}|} \; .$$
$$\frac{n_y}{m_y} = \frac{|y_{end} - y_{start}|}{|y'_{end} - y'_{start}|} \; .$$
$$\frac{n_z}{m_z} = \frac{|z_{end} - z_{start}|}{|z'_{end} - z'_{start}|} \; .$$

Hence it is possible to achieve $T^3$ structure for the magnetic surveying model as well.

Chapter 4

# Preprocessing of Data

As discussed in the previous chapter we need multi-level data sets. However, this choice requires additional preprocessing of the data before the right-hand side **b** can be set up. In this chapter we will describe the decisions we have made and the steps that lead to the setup.

It is important to notice that the existing version of the `GravMagTool` package is implemented in such a way that the units are of no importance as long as the use of units are consistent. We decide to implement the same feature in the new version.

## 4.1 Data

In the new implementation of the `GravMagTools` package the user is required to input an observation set in order to setup and reconstruct an inverse problem. In Chapter 3 it was shown that several levels of data are needed in order to achieve the $T^3$ structure. All data sets we have access to have been measured at the same height above sea level. For some of the surveys it is difficult to keep the exact same height when taking the measurements and for this reason it is sometimes necessary to interpolate the measurements to a single level. This is the first step in Figure 4.1. The next step is to interpolate the measurements so they are placed on a grid. When the data is placed on a grid it is possible to perform an upward continuation. We will return to the upward continuation in Section 4.1.2. The University of Naples has provided us with several data sets

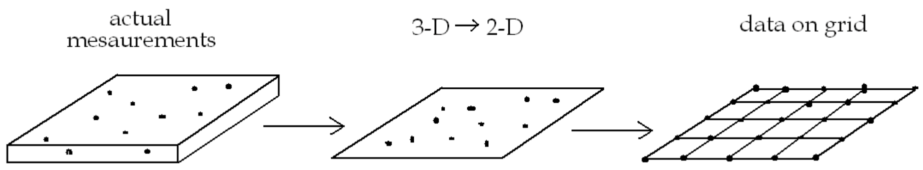


Figure 4.1: Illustration of some of the initial pre-processing of data.

where the data points have all been placed in one level for each set. For this reason we decide that the geophysicists will preprocess the data so the input will be data in a plane. This choice is based on the fact that a geophysicist knows more about the physics of the problems and in some cases they have a priori knowledge about the fields they are trying to reconstruct. Either way they are more likely to know which sources of error they introduce. This knowledge is out of the scope of this project, that deals with inversion of data. Hence, the input to our algorithm is a measurement plane, and from there we will perform the interpolation that will place the data on a grid.

### 4.1.1 Interpolation

Before the interpolation can be conducted a dicretization needs to be performed. The measurement data is in a plane. To be able to perform an interpolation to a grid of a certain fineness the user specifies an interval in the $x$ and $y$ direction and the number of points in each direction. From this input the software calculates the grid spacing in the $x$ and $y$ directions:

$$\Delta_x = \frac{|x'_{\texttt{end}} - x'_{\texttt{start}}|}{m_x}, \qquad \Delta_y = \frac{|y'_{\texttt{end}} - y'_{\texttt{start}}|}{m_y}$$

From Section 3.2.3 we recall the conditions that need to be met in order to achieve a $T^3$ structure do not include that $\Delta_x = \Delta_y$. For this reason the new implementation of the `GravMagTools` package had initially no requirements to the relationship between the two grid spacings. However, the geophysicists from the University of Naples require that the spacings are identical in the $x$ and $y$ direction. The software they use require that $\Delta x = \Delta y$ and the data sets are thus designed for this purpose. Because we want to preserve a consistency between the work in Naples and the package, we have chosen to make this another requirement of the user. When performing the discretization we use a midpoint quadrature rule. Hence we place the first grid point at $(x'_{\mathtt{start}} + \frac{1}{2}\Delta_x, y'_{\mathtt{start}} + \frac{1}{2}\Delta_y)$ and we place the last grid point at $(x'_{\mathtt{end}} - \frac{1}{2}\Delta_x, y'_{\mathtt{end}} - \frac{1}{2}\Delta_y)$. In Figure 4.2 we have illustrated an example of a discretization. In this example $x' = [0\ \ 3.5]$ and $y' = [0\ \ 2]$ and $m_x = 7$ and $m_y = 4$.

To perform the actual interpolation we have to choose the method to use.

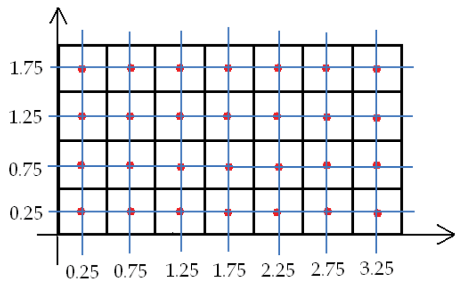


Figure 4.2: Illustration of example of the discretization.

We need a method that can take randomly placed measurement points and place them on a 2-D grid as illustrated in Figure 4.3[1]. An important quality of the interpolation method is that it has to be able to perform an interpolation that does not magnify errors in the data. We need to consider the limitations concerning the placement of the measurement points with respect to the discretization points. We have chosen to consider the `Dace` package [10] and Matlab's `griddata` function. Both interpolation methods are able to take randomly located data points and interpolate to a grid in a plane. Neither method can calculate the function value of a discretization point using extrapolation. For this reason we have chosen to exclude the option of extrapolation of data, it simply leaves too many possibilities of errors.

We choose to perform the interpolation using the `griddata` function. This choice is based on the fact that in this context none of the packages seem to hold any relevant advantages over the other. Since the `griddata` function is included in Matlab it requires the minimum effort of the user. Using the `griddata` function we have different options with respect to the interpolation method. We list the options in Table 4.1. The method needs to be continuous in the 0$^{\mathtt{th}}$ derivative (the function itself). This demand rules out nearest neighbor interpolation. We have no need for derivatives - other then the 0$^{\mathtt{th}}$. We decide that

[1]Even if the user inputs data in a 2-D grid there is most likely still a need for interpolation/regridding for instance in order to adjust the coarseness of the grid.

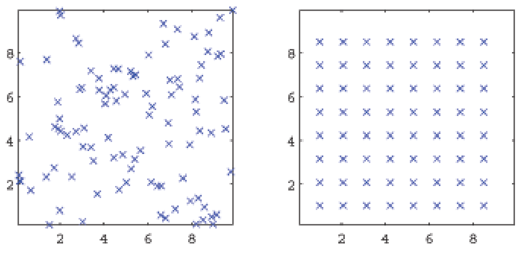


Figure 4.3: Illustration of the interpolation.

there is no reason to compromise the speed of the method and choose the linear interpolation method. Another clear advantage using linear interpolation is

| Interpolation method | Smoothness | Remark |
|---|---|---|
| Linear | Discontinuous in 1st derivative | Faster than cubic and the matlab4 griddata methods. |
| Nearest neighbor | Discontinuous in 0st derivative | |
| Cubic | Smooth | |
| matlab4 griddata | Smooth | |

Table 4.1: Interpolation options using `griddata`.

that according to [1] the maximum induced error is no larger than the maximum error in each of the endpoints. Thus the error in data is not enhanced when using a linear interpolation method.

We illustrate the interpolation using a real data set (the Campanian Plain gravity field). This is done in order to illustrate how the interpolation and in this case the regridding operates. The interpolation is performed using a data set consisting of $53 \times 84$ measurements. The new interpolated set is a $13 \times 13$ set in approximately the same interval. In Figure 4.4 the visual result of the interpolation is illustrated on the right-hand side (on the left-hand side the full data set is illustrated). The result looks correct as it is a coarse version of the full data set.

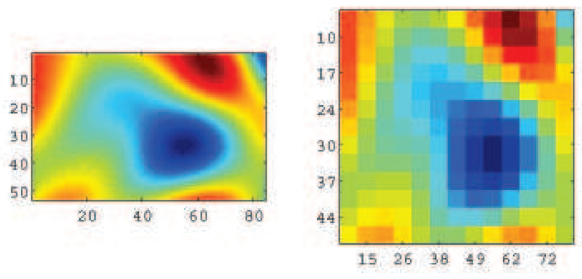


Figure 4.4: Illustration of interpolation. A $53 \times 84$ set is interpolated to a $13 \times 13$ set. The axis properties are not correct in the right figure.

### 4.1.2 Upward Continuation

After the data have been interpolated to a grid specified by the user, the upward continuation is performed in order to achieve the multiple data levels that is required for the $T^3$ structure.

The upward continuation software we utilize is code provided to us by Valeria Paoletti of the University of Naples. The method takes as input the first level of data, and using a fast Fourier transformation calculates each of the following levels as illustrated in Figure 4.5.

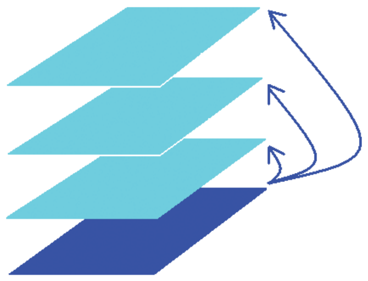


Figure 4.5: Illustration of the upward continuation. The dark blue color represents the input level, while the light blue levels are the computational levels.

Upward continuation is a 2 dimensional deconvolution. According to [12] upward continuation can be expressed using the Dirichlet integral

$$H(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{\frac{h}{2\pi}}{(h^2+\alpha^2+\beta^2)^{\frac{3}{2}}} \cdot F(x-\alpha, y-\beta)d\alpha d\beta \ , \tag{4.1}$$

where $H(x, y)$ is the field measured on a plane at the height $h$ above the source, and $F(x, y)$ is the field at the source plane. If the field at the source plane is known all over then the integral is a physical model. However, we only have measurements in a finite surface. Furthermore the function is not known all over only samplings are known. For these reasons the Fourier based upwards continuation is only an approximation to the physical model. We know that if the field converges towards 0 in a proper manner at the edges of the measurement surface then the approximation to the physical model is good.

The corresponding input-output frequency equation to (4.1) is

$$\hat{H}(u, v) = \hat{F}(u, v)Y_{up}(u, v) \ , \tag{4.2}$$

where $\hat{H}$ is the 2-D Fourier transformed version of $H$, $\hat{F}$ is the 2-D Fourier transformed version of $F$, and $Y_{up}$ is the weighting function and is the 2-D Fourier transformed version of $\frac{\frac{h}{2}}{(h^2+\alpha^2+\beta^2)^{\frac{3}{2}}}$.

According to [12], the upward continuation filter response is given by

$$Y_{up}(u, v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{\frac{h}{2\pi}}{(h^2+\alpha^2+\beta^2)^{\frac{3}{2}}} e^{-i(ux+vy)}dxdy = e^{-h\sqrt{v^2+u^2}} \tag{4.3}$$

The upward continuation is performed using FFT, which assumes that the signals are periodic. Due to the periodicity it is important that the transitions are smooth in order to avoid border effects[2]. For this reason it is useful to place an artificial[3] border around the data before the upward continuation is performed. This border has experimentally proven to be most efficient if the total size of the altered data array is a power of 2. There are several options when choosing the type of border, however, a smooth extension of order 0 has experimentally been the most successful. In Figure 4.6 an illustration of a $16 \times 16$ data array that is expanded with an artificial border of size 120 in each direction resulting in a total size of $256 = 2^8$. The type of border is a smooth extension of order 0. In Figure 4.7 we give a small example of a matrix $\mathbf{P}$ that we wish to expand 2 data points in each direction using a smooth extension of order 0. The resulting matrix $\tilde{\mathbf{P}}$ is then illustrated on the right hand-side of the figure. The original data set is highlighted and appears in the middle of the new set.

To show an example of an upward continuation we take a magnetic observation level and perform upward continuation in 11 levels. The result is plotted in Figure 4.8 where each of the levels is plotted separately. Level 0 is the input data level and levels 1-11 are the levels created in the upward continuation process. As expected we see in the figure how the field grows wider and weaker as the height increases.

[2]Border effects are errors in the outer edges of the data levels.

[3]By artificial we mean not field measurements.

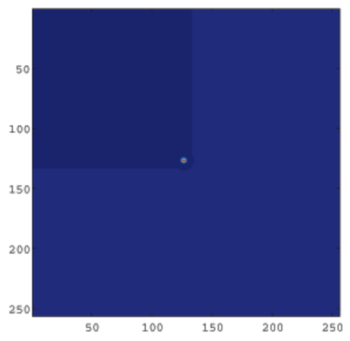


Figure 4.6: Example of a bordered matrix. The original data is the tiny spot in the middle.

$$P = \begin{array}{ccc} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{array} \longrightarrow \widetilde{P} = \begin{array}{cc|ccc|cc} 1 & 1 & 1 & 2 & 3 & 3 & 3 \\ 1 & 1 & 1 & 2 & 3 & 3 & 3 \\ \hline 1 & 1 & 1 & 2 & 3 & 3 & 3 \\ 4 & 4 & 4 & 5 & 6 & 6 & 6 \\ 7 & 7 & 7 & 8 & 9 & 9 & 9 \\ \hline 7 & 7 & 7 & 8 & 9 & 9 & 9 \\ 7 & 7 & 7 & 8 & 9 & 9 & 9 \end{array}$$

Figure 4.7: Illustration of a smooth extension of order 0.

## 4.2 Studies of Interpolation and Upward Continuation

### 4.2.1 Test of Interpolation

In the first test, two different sets of observations are calculated for the same magnetic source. The only difference being the number of measurement points. The interpolated observations (red crosses) are calculated using the values of the measurement points (blue circles) as shown in Figure 4.9. On the left-hand side a set with few measurement points and on the right-hand side a set with several measurement points. In both figures the blue circles represent the actual measurement points and the red crosses are the grid that the blue circles are to be interpolated to. In order to have something to compare the interpolated

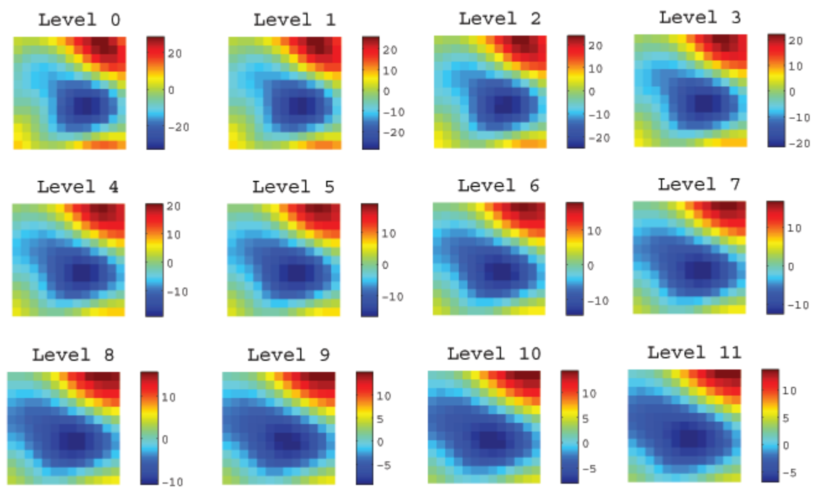


Figure 4.8: An example of an upward interpolation from level 0 to levels 1-11.

values with we calculate the exact measurement values in the red crosses (for the same source).

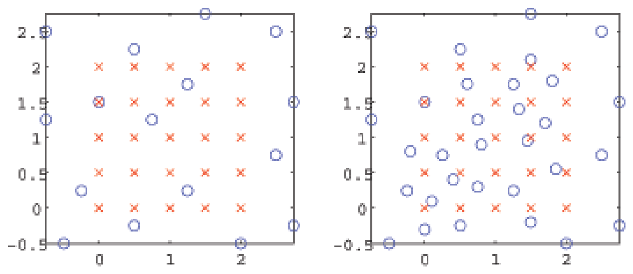


Figure 4.9: Placement of measurement points(blue circles) in two separate interpolations. The red crosses are the grid we wish to interpolate to.

The interpolation is performed using `griddata` and it is now feasible to compare the exact observation values of the red crosses with the interpolated values. Since measurement errors are unavoidable we have chosen to perform the interpolation with and without noise (order of magnitude 10% relative noise normally distributed) in order to see the influence the errors has on the interpolation re-

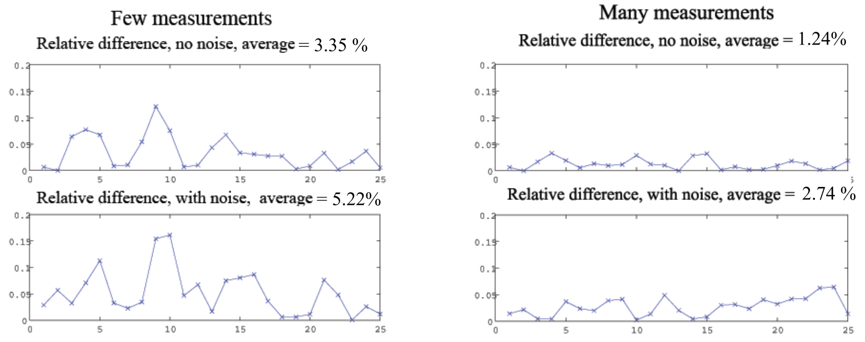


Figure 4.10: Relative differences between the exact set and the interpolated sets with and without noise.

sult. The relative differences in each of the 25 grid points are plotted in Figure 4.10. The relative differences has been calculated as

$$\frac{\|g_{\texttt{interpolate}} - g_{\texttt{exact}}\|_2}{\|\texttt{max}(g_{\texttt{interpolate}}) - \texttt{min}(g_{\texttt{interpolate}})\|_2} \, .$$

When considering the relative differences with no noise in Figure 4.10 (the top plots) the importance of the number of measurement points in the interpolation becomes clear. The average of the relative differences on the left-hand side is 3.35% whereas using several points yields an average relative difference of 1.24%. The conclusion is self-evident: the more measurement points that are used in the interpolation the better result. When taking into account the noisy data it can generally be stated that the relative differences grows when noise is added for both the test with few points and the test with several points. However, it is important to mention that the noise is created using `randn` which is a Matlab function creating normally distributed random numbers. Thus the tests including noise varies depending on the testrun. But still we see how using several measurement points makes the interpolation more accurate.

### 4.2.2 Test of Upward Continuation

In this line of testing we perform an upward continuation. In the test we fix the border size to 512 and the type of border to be a smooth extension of order 0. The setup for the following tests are visualized in Figure 4.11. On the left side the simulated[4] multi-level observation set is shown and on the right hand-

[4]By simulated data we mean data levels for all altitudes of the specified grid.

side the set calculated using the upward continuation method. In the upward continuation the input level is the simulated bottom level from the left hand-side of the figure. This level is then used to calculate the remaining levels using upward continuation. Now we consider the relative differences in each of the 4 data levels of this particular system. The relative difference is calculated as follows:

$$\frac{\|(g_i - g_{i,\mathtt{ex}})\|_2}{\|(g_{i,\mathtt{ex}})\|_2} ,$$

where $g_i$ is the calculated value at the $i^{\mathtt{th}}$ level and $g_{i,\mathtt{ex}}$ is the exact values of the $i^{th}$ level.

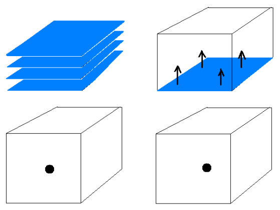


Figure 4.11: Setup for testing the accuracy of the upward continuation.

The relative differences are plotted as a function of the number of level on the left hand-side of Figure 4.12. As the levels grow the relative difference between the simulated and the upward continuation data grows. This is expected behavior seeing how the distance to input level grows for each level. On the right hand-side of Figure 4.12 we have projected the source onto the first level of data (the source illustrated by the green square). In Figure 4.12 the source is, when projected onto the lowest level of observations, placed approximately in the middle of the observations. For this test the maximum relative error is 0.9% and the development of the error is approximately linear. We then repeat the test, this time placing the source so that the projection of the source is placed along the border of the observations. The resulting plots are shown in Figure 4.13. When considering the maximum relative error of 44% it is clear that the placement of the source is of great importance.

#### 4.2.2.1 Natural Border

Until now we have been performing the upward continuation using only the artificial border in the upward continuation method. Now we wish to examine the effect of placing a border of natural data around the input data level, by natural data we mean actual field measurements. After performing the upward

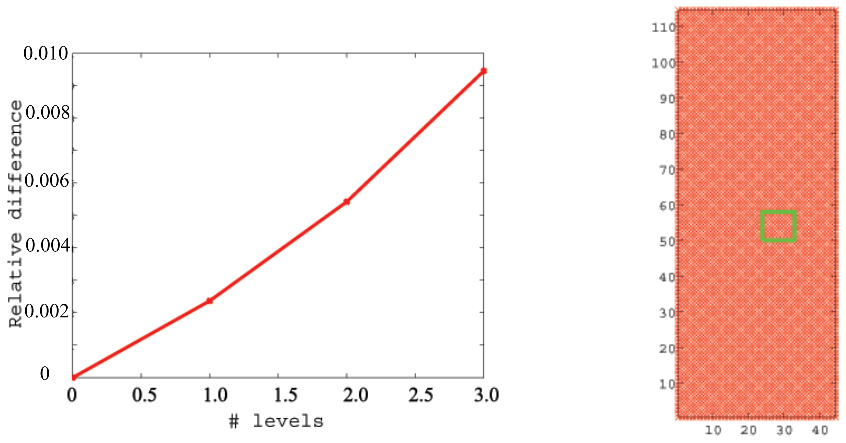


Figure 4.12: Relative difference between the levels of the simulated set and the set calculated using upward continuation. To the right is a illustration of one of the observation levels and the projection of the source(the green box)

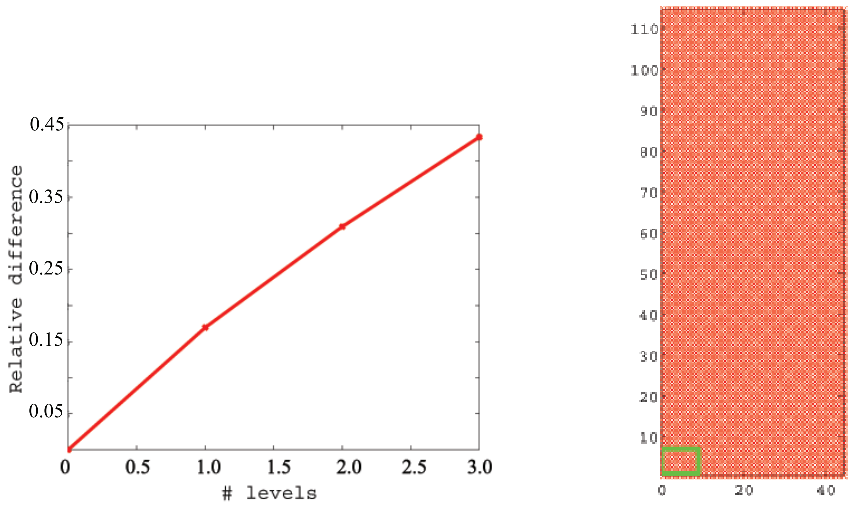


Figure 4.13: Relative difference between the levels of the simulated set and the set calculated using upward continuation. To the right is a illustration of one of the observation levels and the projection of the source(the green box)

continuation of the extended input data level, all data levels of the resulting observation grid will then be cut to the size of the original input. The following line of testing will show whether a natural border reduces the error of the upward continuation and furthermore the effect of the placement of the source when performing the upward continuation.

We create a forward problem as shown in Figure 4.14. We are not interested in the inversion of the problem solely the observations. The source is a cube placed under the surface of the earth. We have as shown in the figure, calculated the observations in all the 4 levels. But seeing how a geophysicist would (most likely) present us with a single plane of observations, we take out the lowest plane and perform an upward continuation of this to a total of 4 levels (3 additional levels). It is then possible to compute the relative difference of each of the levels comparing the exact values with the values of the upward continuation.

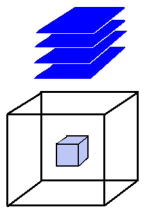


Figure 4.14: Illustration of the forward problem. The solution volume is the discretization domain and the light-blue cube is the source with intensity of 1 (the rest of the discretization domain is 0).

We perform three different tests. For each of the three tests we plot two figures. One illustrating the observations at the input level. In this figure the blue crosses illustrate the area that represent the input level (without the natural border) of the upward continuation, the red crosses represent the natural border, and a green square shows the projection of the source onto the observation grid. The second figure illustrates the relative differences between the simulated set and the upward continued data using a natural border (blue line) and using no natural border (red line).

- **Large natural border, placement of source in the middle**

In the first test we create a relatively large natural border placed around the data that we wish to perform an upward continuation of. First we perform an upward continuation of the blue crosses on the left-hand side of Figure 4.15.

The 4 levels are then compared to the exact data and the relative differences are plotted using the red line on the right hand-side of the Figure 4.15. Second we perform an upward continuation using the data set with the natural border (shown by the red crosses). The comparison of this result with the exact data is shown using the blue line on the right hand-side of the figure. When considering

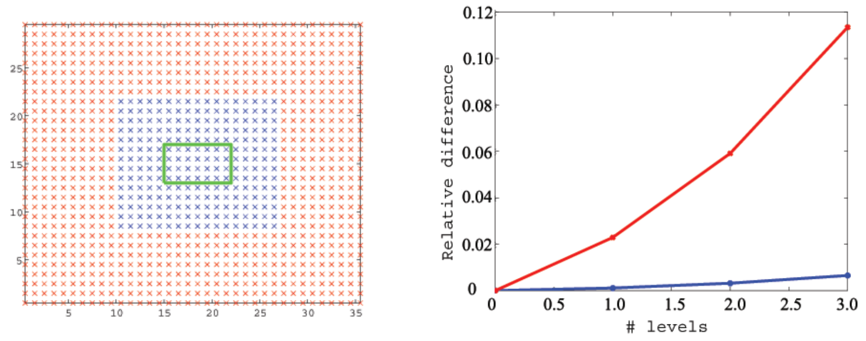


Figure 4.15: To the left an illustration of the observations used in the upward continuation. On the right hand-side the relative difference of no natural border compared with the exact observations (red line) and the large natural border compared to the exact observations (blue line).

the comparisons in Figure 4.15 is becomes clear that the natural border is of great importance when wanting to reduce the error of the upward continuation. The comparison using no border reaches a relative error of 11%, whereas when using the large natural border reaches about 1%. Now the question becomes will a smaller natural border also be able to reduce the error?

- **Small natural border, placement of source in the middle**

In test number two we reduce the width of the natural border to a single data point. The relative error when using no natural border is still 11%. When using the natural border of one data point the relative difference drops to about 7.9%. Thus, the natural border has an effect even when it is small.

- **Large natural border, placement of source in lower right corner**

In this third and final test we work with the large natural border once again. This time we place the source differently this can be seen by observing the

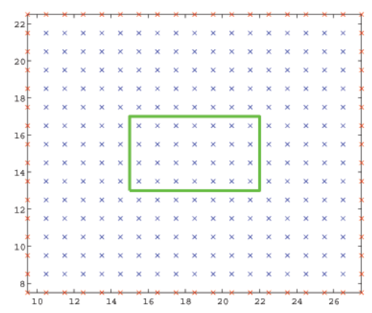
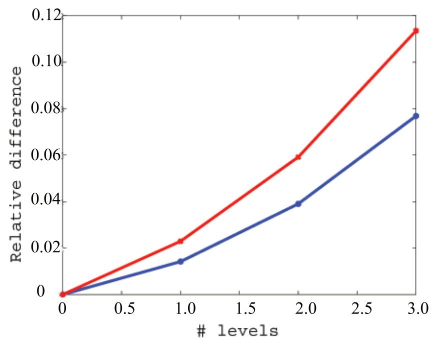



Figure 4.16: On the left-hand side an illustration of the observations used. To the right the relative difference of no natural border compared with the exact observations (red line) and the small natural border compared to the exact observations (blue line).

projection of the source in Figure 4.17 that it is now placed at the edge of the blue crosses. This time the importance of the natural border becomes apparent. The difference between the exact levels and the data using no natural border is about 52%. The difference when using the large natural border is still about 2%. This is of great importance because it is not always possible to predict where the projected source will be.

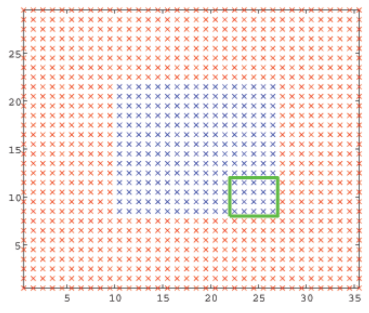
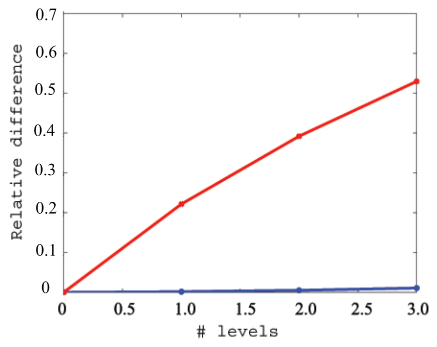



Figure 4.17: To the left an illustration of the observations used in the upward continuation. The source is in this case placed on the edge of the blue crosses. To the right the relative difference of no natural border compared with the exact observations (red line) and the large natural border compared to the exact observations (blue line).

#### 4.2.2.2 Placement of Solution- and Observation grid

In the following test we wish to investigate how the distance between the solution and the observation grid effects the upward continuation. The grids are both of size $15 \times 15 \times 6$. The grid spacing in $x$ and $y$ direction is 0.5 and 0.25 in z direction. The source is a gaussian and placed in the middle of the solution grid. In the test we fix the placement of solution grid whereas the observation grid (placed directly above the solution grid) is moved upwards so that the distances between the two grids gradually increases. The 6 observation levels are simulated in one case and calculated using the upward continuation for the other case. We calculate the relative difference between the upward continued data and the simulated data. It is important to stress that the upward continuation method is given good conditions by the use of 7 points of natural border in each direction. The result of the test is illustrated in Figure 4.18. The upward continuation

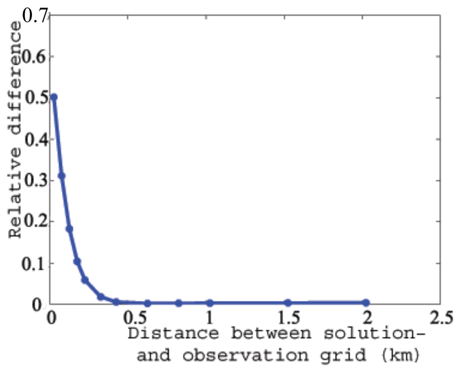


Figure 4.18: The relative difference between the upward continued data and the simulated data for different distances between the solution and observation grid

has problems calculating a good approximation to the simulated data when the distance between the grids is small. The further away the observation grid is from the source is the more it resembles a point source. The upward continuation performs well when approximating a point source.

We replicate the test but this time we change the inclination and declination from [90 0] degrees to [60 40] degrees. When comparing Figure 4.19 to Figure 4.18 it can be seen how the upward continuation can not perform well when the inclination and declination is different from [90 0]. To understand why we take a look at Figure 3.9, in this figure the dipole has an inclination of 90 degrees and a declination of 0 degrees. If the inclination and declination of the induced field from the dipole are changed then the field becomes more complicated. This is the cause of the bad performance of upwards continuation when the angles are

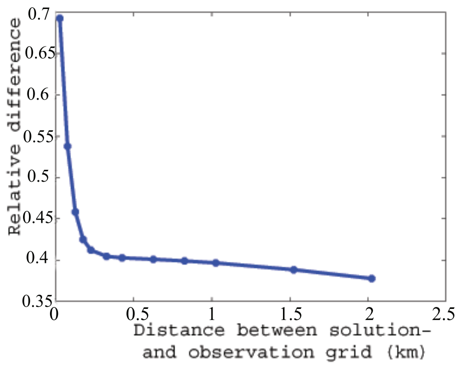


Figure 4.19: The relative difference between the upward continued data and the simulated data for different distances between the solution and observation grid. The inclination is 60 degrees and the declination is 40 degrees

changed.

#### 4.2.2.3 Summary

This line of testing clearly shows that if there is extra data then it is an advantage to use the extra information. The larger the natural border the better, but even a small border helps to reduce the error in the upward continuation method. For this reason we create an algorithm that ensures that the border is chosen to be as wide as possible. For further explanation of this algorithm see appendix B. Furthermore the importance of placing the observations appropriate with respect to the expected source is illustrated. But we realize that when dealing with a real field inversion it is not always possible to predict the placement of the source. In the last tests we performed we see an indication that performance of the upward continuation is not as stable as we had hoped. It seems to be limited even in the cases where it performs well. In Chapter 7 we will perform inversions with upward continued data.

Chapter 5

# Regularization Algorithms

We want to solve the discretized problem $\mathbf{Am} = \mathbf{b}$. The naive solution as found using a least squares minimization

$$\texttt{min}_{\mathbf{m}} \|\mathbf{Am} - \mathbf{b}\|_2 \ . \tag{5.1}$$

is useless when dealing with ill-posed problems. The problem is that infinitely small perturbations in the input data $\mathbf{b}$ can cause infinitely large perturbations in the solution $\mathbf{m}$ due to the nature of the ill-posed problem. For this reason we introduce regularization methods that according to [8] enforce the regularity on the computed solution - typically in the form of a requirement that the solution is smooth, in some sense. This means that the high frequencies are damped in the solution.

In this section we will perform an Singular Value Decomposition[1]. In order to obtain an insight into the problems. This insight can be used in some regularization methods as well as for understanding other methods. However the task of computing an SVD is computationally heavy making the regularization methods depending on an SVD feasible only for small problems. We will then describe the theory of the Tikhonov regularization method and CGLS regularization algorithm. Both Tikhonov and CGLS are, if implemented for the purpose, suitable for large-scale problems.

[1]The Singular Value Decomposition will from this point on be referred to as the SVD

## 5.1 Singular Value Expansion

The method of this investigation is a Singular Value Decomposition. To be able to understand how the SVD works and which information we gain using it, we will first take a look at the Singular Value Expansion[2].

We recall Equation (3.2) which shows that the generic problem

$$\int_{\mathbf{\Omega_r}} K(\mathbf{r}',\mathbf{r})f(\mathbf{r})d\mathbf{r} = g(\mathbf{r}'), \quad \mathbf{r}' \in \mathbf{\Omega}_{\mathbf{r}'} \ .$$

To be able to perform an SVE analysis of the kernel $K$ we must assume that $K$ is square integrable and therefore $\int_{\mathbf{\Omega r}}\int_{\mathbf{\Omega r}'} K(\mathbf{r}',\mathbf{r})^2 d\mathbf{r}d\mathbf{r}'$ must be finite. The reason for this assumption is that in reality the quantities of the magnetization and gravitation are finite. From [8] we know that the SVE for any square integrable kernel is given by

$$K(\mathbf{r}',\mathbf{r}) \doteq \sum_{i=1}^{\infty} \mu_i \mathbf{u}_i(\mathbf{r}')\mathbf{v}_i(\mathbf{r}) \ .$$

The $\doteq$ in the above formulation means that the right side converges in the mean to the left side. The function $\mathbf{u}_i$ is designated the left singular function and $\mathbf{v}_i$ is designated the right singular function. The functions $\mathbf{v}_i, i = 1, 2, ...$ are orthonormal to $\mathbf{v}_i, i = 1, 2, ...$ with respect to the usual inner product. The same applies to $\mathbf{u}_i$. The inner product between the functions is defined as $\langle \phi, \psi \rangle = \int_{\mathbf{\Omega}} \phi(\mathbf{r})\psi(\mathbf{r})d\mathbf{\Omega}$. This implies that

$$\langle \mathbf{u}_i, \mathbf{u}_j \rangle = \langle \mathbf{v}_i, \mathbf{v}_j \rangle = \begin{cases} 1, & i = j \\ 0, & \text{otherwise} \end{cases} ,$$

where $\mu_i$ are the singular values which are a non-increasing sequence.

From [8] we know by expanding the functions $f$ and $g$ and by using the "fundamental relation", an expression for the solution $f(\mathbf{r})$ can be derived

$$f(\mathbf{r}) \doteq \sum_{i=1}^{\infty} \frac{\langle \mathbf{u}_i, g \rangle}{\mu_i} \mathbf{v}_i(\mathbf{r}) \ .$$

The solution $f$ has to be square integrable. This implies that the following expression for the 2-norm of $f$ must be satisfied

$$\|f\|_2^2 = \int_{\mathbf{\Omega}} f(\mathbf{r})^2 d\mathbf{\Omega} = \sum_{i=1}^{\infty} \left( \frac{\langle \mathbf{u}_i, g \rangle}{\mu_i} \right)^2 < \infty \tag{5.2}$$

[2]The Singular Value Expansion will from this point on be referred to as the SVE

The 2-norm is used to derive Equation (5.2) because the usual inner product is used to derive the expressions in the SVE. Now we take a closer look at (5.2) to understand what this expression says about the solution. The expression is true if the solution coefficients $\langle \mathbf{u}_i, g \rangle$ decay in a proper way faster towards zero than the singular values $\mu_i$. The relation (5.2) is also known as the Picard condition. The SVE is not suited for numerical computations, but based on the SVE we are able to create a discrete version which works on finite dimensional matrices. This version is know as the SVD.

## 5.2 Singular Value Decomposition

We know from Equation (3.4) that the discretized problem takes the form

$$\mathbf{A}\mathbf{m} = \mathbf{b} \ .$$

The SVD can be performed on any rectangular matrix and it holds a fundamental relationship to the SVE, which we will describe below. In the following expression of the SVD we assume that the matrix either has more rows than columns or is square for reasons of simplicity. An SVD of any matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ with $m \geq n$ will take the form

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \sum_{i=1}^{n} \mathbf{u}_i \sigma_i \mathbf{v}_i^T \ .$$

The columns in matrices $\mathbf{U} \in \mathbb{R}^{n\times n}$ and $\mathbf{V} \in \mathbb{R}^{n\times n}$ are called the left and right singular vectors. The columns in $\mathbf{U}$ and $\mathbf{V}$ are orthonormal

$$\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_n] \ , \mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_n] \ ,$$

$$\mathbf{U}^T\mathbf{U} = \mathbf{V}^T\mathbf{V} = \mathbf{I} \ .$$

The diagonal matrix $\mathbf{\Sigma} \in \mathbb{R}^{m\times n}$ contains the singular values. These singular values have the property that they are a non-increasing sequence

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0 \ . \tag{5.3}$$

As mentioned before the SVD can be regarded as an approximation to the SVE. In this approximation the singular values $\sigma_i$ of the matrix $\mathbf{A}$ are approximations to the singular values $\mu_i$ from the kernel $K$. In the same way the left and right singular functions can be approximated. If these approximations are denoted $\tilde{\mathbf{u}}_j(\mathbf{r}')$ and $\tilde{\mathbf{v}}_j(\mathbf{r})$ where $j = 1, \ldots, n$ then we know from [8] that the inner product $\langle \tilde{\mathbf{u}}_j, \tilde{g} \rangle$ approximates the inner product in the SVE and will have the definition

$$\langle \tilde{\mathbf{u}}_j, g \rangle = \mathbf{u}_j^T \mathbf{b} \ .$$

We can now derive expressions for $\mathbf{m}$, $\mathbf{b}$ and $\mathbf{Am}$

$$\mathbf{m} = \sum_{i=1}^{n} (\mathbf{v}_i^T \mathbf{m})\mathbf{v}_i \ , \quad \mathbf{b} = \sum_{i=1}^{n} (\mathbf{u}_i^T \mathbf{b})\mathbf{u}_i \ ,$$

$$\mathbf{Am} = \sum_{i=1}^{n} \sigma_i (\mathbf{v}_i^T \mathbf{m})\mathbf{u}_i \ .$$

The expression for $\mathbf{Am}$ and $\mathbf{b}$ can be used to formulate an expression for the solution $\mathbf{m}$. If $\mathbf{A}$ has rank $n$ then all the singular values are positive, and the expression will be

$$\mathbf{m} = \mathbf{A}^{-1}\mathbf{b} = \sum_{i=1}^{n} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i \tag{5.4}$$

It is now possible to define a discrete Picard condition by using Equation (5.4). It is clear by looking at Equation (5.4) and (5.2) that the two are related. The formulation of the discrete Picard condition is formulated in the box below. An

> The Picard condition is satisfied if for all singular values $\sigma_i$ which are not dominated by errors, then the coefficients $|\mathbf{u}_i^T \mathbf{b}|$ must decay faster on average than the singular values $\sigma_i$.

easy way to check if the Picard condition is met is to construct a Picard plot. A Picard plot is a plot where the singular values $\sigma_i$, the coefficients $|\mathbf{u}_i^T \mathbf{b}|$, and the solution coefficients $|\mathbf{u}_i^T \mathbf{b}|/\sigma_i$ are plotted as a function of $i$. The Picard plot makes it possible to make a visual inspection of the Picard condition.

In the following we will illustrate how the Picard plot can be used. We create two examples of a one dimensional gravity surveying problem[3]. It is important to notice that we do not use upward continuation to generate data, instead all data is simulated. In the first example we invert a problem where the right hand-side $\mathbf{b}$ is without any influence of noise. In the second example the same problem is set up except for the fact that the right hand-side is effected by white noise with a relative noise level of $10^{-5}$. Figure 5.1 shows the Picard plot for the first example. The coefficients $|\mathbf{u}_i^T \mathbf{b}|$ (illustrated by $\times$) decay faster on average than the singular values $\sigma_i$ (illustrated by ●) until the singular values become so small that they are dominated by the rounding errors. Thus, the Picard condition is satisfied for the first problem. The second example has a noisy right hand-side $\mathbf{b}$ and is therefore of more interest to us, because real field measurements are noisy. The Picard plot for the second example is illustrated in Figure 5.2. The SVD coefficient $|\mathbf{u}_i^T \mathbf{b}|$ decay until they reach the noise level

[3]The details of the discretized problem can be seen in Section 3.2.1.

at $10^{-5}$, where they level out. The singular values $\sigma_i$ decay until they get so small that the rounding errors start to dominate. We can see in Figure 5.2 that the Picard condition is satisfied until the SVD coefficients $|\mathbf{u}_i^T\mathbf{b}|$ reach the noise level. After this point the singular values decay faster than the SVD coefficients. In the Picard plot we also plot the solution coefficients (illustrated by $\circ$). The solution coefficients increase drastically, beginning when the Picard condition is no longer satisfied. This implies that the solution no longer is square integrable. To understand what happens we take a closer look at the term $|\mathbf{u}_i^T\mathbf{b}|$ in the SVD formulation. We know that the right hand-side $\mathbf{b}$ is affected by noise, hence

$$\mathbf{u}_i^T\mathbf{b} = \mathbf{u}_i^T(\mathbf{b}_{exact} + e) \ ,$$

where $e$ is the noise and $\mathbf{b}_{exact}$ is the exact noise-free right hand-side. Let

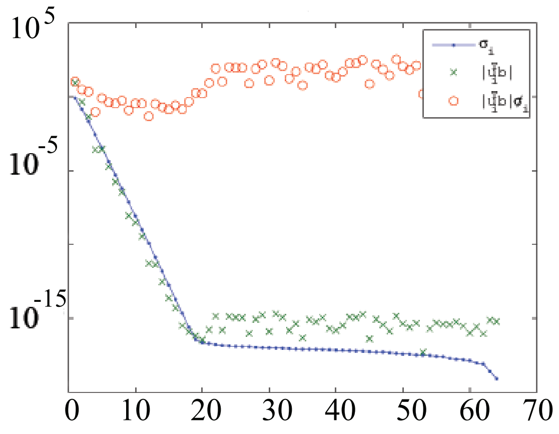


Figure 5.1: Picard plot for a 1-D gravity surveying problem without noise. The figure shows the singular values $\sigma_i$ ($\bullet$), the right hand-side coefficients $|\mathbf{u}_i^T\mathbf{b}|$ ($\times$) and the solution coefficients $\frac{|\mathbf{u}_i^T\mathbf{b}|}{\sigma_i}$ ($\circ$)

$\tau$ denote the index of which the coefficients $|\mathbf{u}_i^T\mathbf{b}|$ no longer decay faster than $\sigma_i$ thus the Picard condition is no longer satisfied. Hence we can write $\mathbf{u}_i^T\mathbf{b}$ in another way

$$\mathbf{u}_i^T\mathbf{b} = \mathbf{u}_i^T\mathbf{b}_{exact} + e_i \simeq \begin{cases} \mathbf{u}_i^T\mathbf{b}_{exact}, & \tau \gg i \\ \mathbf{u}_i^T e, & \tau \ll i \ . \end{cases} \tag{5.5}$$

Simplified there are two different kinds of right hand-side coefficients. The first kind is where $\tau \gg i$. These are the coefficients we can trust and which contribute to the solution. The other coefficients where $\tau \ll i$ are dominated by the noise and corresponds to small singular values. These noisy coefficients do not

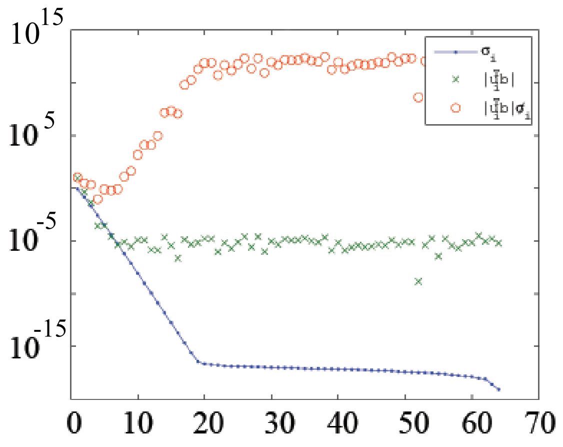


Figure 5.2: Picard plot for a 1-D gravity surveying problem with a noise level of $10^{-5}$. The figure shows the singular values $\sigma_i$ (●), the coefficients $|\mathbf{u}_i^T\mathbf{b}|$ (×) and the solution coefficients $\frac{|\mathbf{u}_i^T\mathbf{b}|}{\sigma_i}$ (○).

contribute to the exact solution, because they are samples of the noise function. As mentioned the noisy coefficients corresponds to small singular values and therefore these solution coefficients will become very large and dominate the solution.

## 5.3 Truncated Singular Value Decomposition

Truncated Singular Value Decomposition[4] is a regularization method that requires an SVD computation.

The idea behind the TSVD method is relatively simple. In Equation (5.5) we saw how we can trust the first SVD coefficient where the Picard condition is fulfilled. These first coefficients contribute to the solution so the idea of the TSVD is only to use the first $k$ components. The TSVD solution $\mathbf{m}_k$ is defined as

$$\mathbf{m}_k \equiv \sum_{i=1}^{k} \frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}\mathbf{v}_i,$$

where the constant $k$ is the truncation parameter which ensures that we only use the first $k$ SVD components. $k$ can be determined by considering the Picard

[4]The Truncated Singular Value Decomposition will from this point on be denoted TSVD.

plot of the problem. When the size of the problems increase is it no longer possible to calculate an SVD. For this reason we need a regularization method which do not rely on an SVD.

## 5.4 Tikhonov Regularization Method

The Tikhonov regularization method has a formulation which includes regularization. The main idea is to obtain a small solution norm, but then accepting that the residual is larger than zero. The Tikhonov formulation takes the general form

$$\mathtt{min}_{\mathbf{m}}\{\|\mathbf{Am}-\mathbf{b}\|_2^2+\lambda^2\|\mathbf{Lm}\|_2^2\}\ , \tag{5.6}$$

where $\lambda \geq 0$ is the regularization parameter that controls the weighting between the first and the second term of the function. Note that when $\lambda = 0$ we obtain the naive solution.

- $\|\mathbf{Am}-\mathbf{b}\|_2^2$ is a measure of how well the solution $\mathbf{m}$ predicts the data $\mathbf{b}$.
- $\|\mathbf{Lm}\|_2^2$ is the regularization term. The term is able to suppress the high frequencies of the large noise components. $\mathbf{L}$ can be a discrete approximation of a derivative operator (or the identity matrix). By including the stabilizer $\mathbf{L}$ we improve the reconstruction because this way we include a priori knowledge about the solution.

The quality of the Tikhonov solution now depends on the regularization parameter. If $\lambda$ is set too low the noisy data is fitted, and if $\lambda$ is set too high too much weight is given to the minimization of the solution norm (and hence for $\lambda \to \infty$ it approaches the nullspcae of $\mathbf{L}$). In [8] the common choices (the first and second derivative respectively) of L are listed for 1-D problems

$$\mathbf{L}_1 = \begin{pmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix} \in \mathbb{R}^{(n-1)\times n} \text{ and}$$

$$\mathbf{L}_2 = \begin{pmatrix} 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \end{pmatrix} \in \mathbb{R}^{(n-2)\times n}\ .$$

Equation (5.6) can be formulated as a least-squares problem of the form

$$\mathtt{min}_{\mathbf{m}} \left\| \begin{pmatrix} \mathbf{A} \\ \lambda\mathbf{L} \end{pmatrix} \mathbf{m} - \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix} \right\|_2 \tag{5.7}$$

We perform an SVD analysis with $\mathbf{L} = \mathbf{I}$ in order to understand the theory of the Tikhonov solution, $\mathbf{m}_\lambda$.

The normal equations for the least-squares problem (5.7) are

$$(\mathbf{A}^T\mathbf{A} + \lambda^2\mathbf{I})\mathbf{m} = \mathbf{A}^T\mathbf{b} \Leftrightarrow \mathbf{m}_\lambda = (\mathbf{A}^T\mathbf{A} + \lambda^2\mathbf{I})^{-1}\mathbf{A}^T\mathbf{b} \ .$$

We now insert the SVD of $\mathbf{A}$ into the normal equations. Furthermore we use the fact that $\mathbf{I} = \mathbf{V}\mathbf{V}^T$

$$\begin{aligned} \mathbf{m}_\lambda &= (\mathbf{V}\mathbf{\Sigma^2}\mathbf{V}^T + \lambda^2\mathbf{V}\mathbf{V}^T)^{-1}\mathbf{V}\mathbf{\Sigma}\mathbf{U}^T\mathbf{b} \\ &= \mathbf{V}(\mathbf{\Sigma}^2 + \lambda^2\mathbf{I})^{-1}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}\mathbf{U}^T\mathbf{b} \\ &= \mathbf{V}(\mathbf{\Sigma}^2 + \lambda^2\mathbf{I})^{-1}\mathbf{\Sigma}\mathbf{U}^T\mathbf{b} \ . \end{aligned}$$

Inserting the singular values $\sigma_i$ and the vectors $\mathbf{u}_i$ (left singular vector) and $\mathbf{v}_i$ (right singular vector) we get

$$\mathbf{m}_\lambda = \sum_{i=1}^{n} f_i \frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}\mathbf{v}_i \ . \tag{5.8}$$

where $f_i$ are the so-called filter factors for $i = 1, ..., n$ satisfying

$$f_i = \frac{\sigma_i^2}{\sigma_i^2 + \lambda} \simeq \begin{cases} 1 & \sigma_i \gg \lambda \\ \frac{\sigma_i^2}{\lambda^2} & \sigma_i \ll \lambda \end{cases}$$

- For singular values greater than the regularization parameter, $\lambda$, the associated filter factor will be (close to) one. This means that the corresponding SVD component contribute to the solution vector with full strength.
- For singular values smaller than the regularization parameter the SVD components are filtered.

Figure 5.3 illustrates the filter factors for Tikhonov (red line) and the filter for TSVD (blue line) respectively. The regularization parameter $\lambda = \sigma_k$ where $k$ is the truncation parameter for TSVD. It is obvious from the figure that the filters used by Tiknonov is smoother than the filter used in TSVD. Thus the TSVD

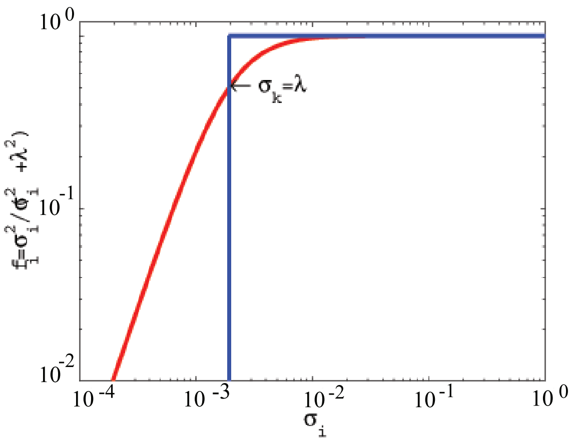


Figure 5.3: The blue line is the filter factor for the TSVD with the truncation parameter $k$. The red line is the filter factor for the Tikhonov where $\lambda = \sigma_k$

and Tikhonov will converge towards different solution. These solution might have same visual appearance but they will be different because the smoother Tikhonov filter add more components then the TSVD filter.

The Tikhonov method can be used for large scale problems if it solved in apporiate way. One way can be to use a stabel implementation of the Conjugate Gradient.

## 5.5 Conjugate Gradients (Least Squares)

Conjugate gradients (CG) is another method for solving a system of linear equations. We have seen how the TSVD is able to make a reconstruction using the first $k$ singular values. This solution is spanned by the first $k$ right singular vectors. The idea in the CG method is to find a set of basis vectors with the same overall features as the first $k$ right singular vectors. We recall the least squares problem $\texttt{min}_{\mathbf{m}}\|\mathbf{Am}-\mathbf{b}\|_2$ and it can be shown that the solution $\mathbf{m}^{(k)}$ is obtained after applying $k$ steps of the CG algorithm

$$\mathbf{m}^{(k)} = \texttt{argmin}_{\mathbf{m}}\|\mathbf{Am}-\mathbf{b}\| \quad s.t. \quad \mathbf{m} \in \mathcal{K}_k \ , \tag{5.9}$$

where $\mathcal{K}_k$ is the Krylov subspace associated with $\mathbf{A}$ and $\mathbf{b}$ defined as the span of powers of $\mathbf{A}^T\mathbf{A}$ applied to $\mathbf{A}^T\mathbf{b}$. The advantage of the CG method is that there is no need to store all the basis vectors explicitly. All that is needed during the iterations is matrix-vector multiplications with $\mathbf{A}$ and $\mathbf{A}^T$ (and some vector operations). In [8] it is stated that the most stable implementation of the CG algorithm is the CGLS algorithm. The pseudocode of this algorithm is:

```
$m^{(0)}$ = starting vector (often zero vector)
$r^{(0)} = b - Am^{(0)}$
$d^{(0)} = A^T r^{(0)}$
for $k = 1, 2, ...$
  $\bar{\alpha}_k = \|A^T r^{(k-1)}\|_2^2 / \|Ad^{(k-1)}\|_2^2$
  $m^{(k)} = m^{(k-1)} + \bar{\alpha}_k d^{(k-1)}$
  $r^{(k)} = r_{(k-1)} - \bar{\alpha}_k A d^{(k-1)}$
  $\bar{\beta}_k = \|A^T r^{(k)}\|_2^2 / \|A^T r^{(k-1)}\|_2^2$
  $d^{(k)} = A^T r^{(k)} + \bar{\beta}_k d^{(k-1)}$
end
```

There are two ways that CGLS can be used for regularization

- apply directly to $\mathbf{Am} = \mathbf{b}$ or
- apply to Tikhonov problem in the least squares formulation (5.7) .

We choose the first approach because for the Tikhonov method we have to find an appropriate value of the regularization parameter $\lambda$. The obvious way to find this value would be to conduct test with different values and this is not suited for large scale problems. When expressing the iteration vector in terms of the SVD of $\mathbf{A}$ we get

$$\mathbf{m}^{(k)} = \sum_{i=1}^{n} f_i^{(k)} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i \tag{5.10}$$

where $f_i^{(k)}$ are the filter factors depending on both $\mathbf{b}$ and the singular values. It can be shown that

$$f_i^{(k)} = \begin{cases} O(\sigma_i^2), & \texttt{for small } \sigma_i \\ 1, & \texttt{for large } \sigma_i \end{cases}$$

In [7] it is shown that CGLS can compute an approximation to a TSVD or Tikhonov solution by using regularizing iterations.

Chapter 6

# Implementation and Representation

The GravMagTools package is an object-oriented package that allows for setting up and inverting geophysical problems. The reason why the package use objects is to obtain a user friendly interface. The use of objects ensures that the user does not need to keep track of where and how the information is stored. Nevertheless the program stores the information in an appropriate way and it is still accessible for the user. The GravMagTools package draws on the object-oriented M$\mathcal{OO}$ReTools package [4]. The GravMagTools package is able to use the collection of iterative algorithms for linear inversion which are implemented in M$\mathcal{OO}$ReTools. The objects of the existing implementation are all maintained in the new GravMagTools package so a user is still able to perform inversions using full structured coefficient matrices. We have implemented new objects and modified some of the existing to work on both $T^3$ structure and full structure. In this section we will describe the objects and the new implementation of the package. We give a thorough description of the representation of the coefficient matrix when using a $T^3$ structure. Furthermore we describe the bookkeeping of the FFT multiplication. Finally we will give a graphical overview of the package.

## 6.1 Objects in GravMagTools

In this section we describe the objects of the package, but only the object connected to the new version of package[1] in which the coefficient matrix is of a $T^3$ structure. We start by describing the solution and data objects which are modified versions of the `GMData` and `GMSolution` objects from the existing implementation. Furthermore we will describe the new regularization object which is closely related to the old object. For these objects we will put great emphasize on describing the modifications and the reasoning behind these. Then we describe the object `GMT3Operator` which stores the representation of the coefficient matrix with a $T^3$ structure. In order to give a overview of the object we have made tables which describe the fields in each of the objects. In these tables a • represents a field. For some of the fields we have chosen not to store all data, but instead store a filename hereby saving memory.

⋆ `GMSolution Object`
This object is designed to contain the solution $\mathbf{m}$ when we solve the problem $\mathbf{Am} = \mathbf{b}$. The object has been modified from the previous version of the package to contain an extra field with information about the type of structure of the corresponding coefficient matrix. It is important to notice that the change in the ordering of the columns of the coefficient matrix causes a change in the ordering of the solution. This will not affect the solution object in other regards than in the respect that the data is arranged differently compared to the solution object of the existing implementation. The extra field we have added to the object is necessary because of the rearranging of the solution. This field enables us to pass along information about the rearrangement of data. In Table 6.1 the fields of the `GMSolution` object is listed.

⋆ `GMData Object`
The information about the right hand-side $\mathbf{b}$ is stored in the `GMData` object. We add three fields to this object; one field with information about the structure of the coefficient matrix, one with information about the solution grid, and finally one containing information about the observation grid. These fields are added in order for the visualization routine to be able to retrieve the required information. Fields contained in `GMData` is shown in Table 6.2.

⋆ `GMT3Regularizer Object`
The `GMT3Regularizer` object is similar to the `GMRegularizer` object in the existing implementation except it uses the different multiplication routine. The new multiplication routine takes into account that when using a $T^3$ structure the data contained in the `GMData` and the `GMSolution` objects are rearranged[2].

[1]The objects of the existing implementation are covered in [5]
[2]The multiplication routine is covered in appendix C.

| Object | Mathematical quantity | Information included in the object |
|---|---|---|
| `m` | Solution vector $\mathbf{m}$ | • Solution elements<br>• Unit vector $\hat{\mathbf{j}}$ with the direction of the field induced by the source (magnetization only) or the string 'gravity' (gravitation only)<br>• Information about the solution grid<br>• A structure containing the filename of topography file and a vector of the points of the solution which is inside the topography<br>• Information about the structure of the corresponding coefficient matrix. |

Table 6.1: `GMSolution` object.

| Object | Mathematical quantity | Information included in the object |
|---|---|---|
| `b` | Data vector $\mathbf{b}$ | • Measured data<br>• Unit vector $\hat{\mathbf{i}}$ with the direction of the induced core field (magnetization only) or a string 'gravity' (gravitation only)<br>• The filename of the file containing the coordinates of observation points<br>• The filename of the file containing the topography data<br>• Information about structure of corresponding coefficient matrix<br>• Information about the solution grid<br>• Information about the observation grid<br>• String containing information about problemtype (inverse or forward) |

Table 6.2: `GMData` object.

The `GMT3Regularizer` object contains the fields listed in Table 6.3

$\star$ `GMT3Operator Object`
The `GMT3Operator` differs significantly from the representation of the coefficient matrix in the existing implementation, `GMMatrix`. `GMMatrix` stores the full matrix while `GMT3Operator` only stores the values of the **EIG** structure[3]. As `GMMatrix` the `GMT3Operator` carries around a structure with information

[3]The **EIG** structure will be explained in details in Section 6.1.1

| Object | Mathematical quantity | Information included in the object (field) |
|---|---|---|
| `L` | Regularization matrix **L** | • Derivative operator<br>• Solution object<br>• Derivatives to use |

Table 6.3: `GMT3Regularizer` object.

about the solution grid, the observation grid, and a filename of the file containing the observations. The remaining fields are identical to those of the existing implementation. The multiplication routine uses an FFT matrix-vector multiplication routine[4]. Table 6.4 lists the fields in `GMT3Operator`.

| Object | Mathematical quantity | Information included in the object |
|---|---|---|
| `A` | Coefficient matrix **A** | • Elements of the array **EIG** (representation of **A**)<br>• Filename of the file containing topography data<br>• Structure containing information about the solution grid, the observation grid and the filename of the file with the observations<br>• A structure with the unit vector $\hat{\mathbf{i}}$ and the unit vector $\hat{\mathbf{j}}$ (magnetization only) or the string 'gravity' (gravitation only)<br>• Information about problemtype (inverse, forward, or solution only) |

Table 6.4: `GMT3Operator` object.

### 6.1.1 Representation of A

In this project we will use iterative methods to solve the generic model $\mathbf{Am} = \mathbf{d}$. For this reason it is important that we can perform a quick matrix-vector multiplication. In the following section we will present our representation of **A** that enables us to implement a fast FFT matrix-vector multiplication. The representation of **A** is denoted **EIG**.

Due to the structure of a $T^3$ coefficient matrix it is no longer necessary to store all elements of **A**. An example of the elements that need to be stored from

[4]The implementation of the FFT multiplication routine will be explained in Section 6.1.2

**A** when $n_x = m_x = n_y = m_y = n_z = m_z = 3$ is illustrated in Figure 6.1. The number of elements when $m_x = m_y = m_z = n_x = n_y = n_z = n$ is given by $2n \cdot (2n-1)^2 \simeq 8n^3$ which takes up much less storage space than the $n^6$ we would need to store if there where no structure in **A**. In general, the number of elements that need to be stored in the new representation of the coefficient matrix is: $((n_x + m_x) - 1) \cdot ((n_y + m_y) - 1) \cdot ((n_z + m_z) - 1)$.

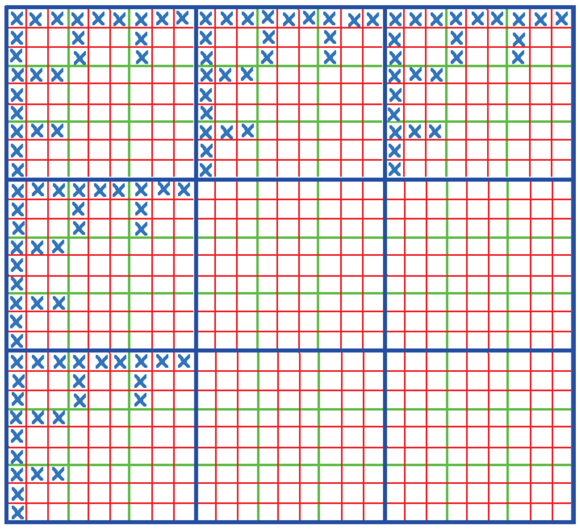


Figure 6.1: Illustration of needed storage elements from **A**

In order to save time in the FFT multiplication as described in Section 2.2, we represent each of the Toeplitz matrices of **A** as a vector containing the eigenvalues of the first row and the first column from the circulant matrices defined from the blocks of **A** (in Section 2.2 this is equivalent to $\mathbf{C}(:, 1)$). When storing eigenvalues we store complex elements. This means that we need twice as much storage space compared to storing one number in double precision, but by doing this we save time in every iteration of the iterative solvers.

The eigenvalue vector for each of the circulant matrices, are stored in the three-dimensional array **EIG**. Each of the columns in **EIG** represent the eigenvalues of one of the circulant matrices from **A**. In Figure 6.2 we illustrate how we store the elements from **A**, but a little explanation is needed. The red circle on the left hand-side of Figure 6.2 that contains 9 double elements is a Toeplitz matrix. For this reason we only need to store 5 of the 9 elements ($(n_x + m_x - 1)$ elements). The 5 elements we take out are changed into a circulant matrix and the first column is taken out. The eigenvalues of the circulant matrix are calculated by computing the FFT of the first column, and it is the resulting vector of eigenvalues that is then stored as a column in **EIG**. The blue circle contains the 9 Toeplitz matrices of the particular BTTB block and they are all stored in the first level of the **EIG** structure ($(n_y + m_y - 1)$ columns). Each level of **EIG** contains information about one BTTB matrix and **EIG** contains $(n_z + m_z - 1)$ levels.

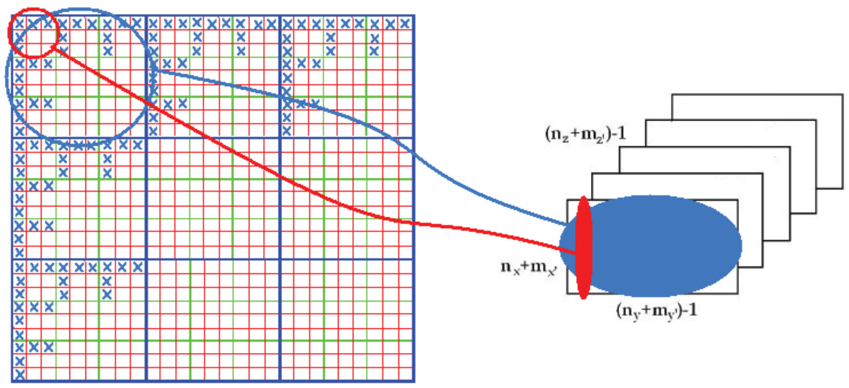


Figure 6.2: On the left hand-side **A**. On the right hand-side the representation of **A** that contains the eigenvalues of the circulant matrices stored in a three dimensional array **EIG**.

This way we compress the coefficient matrix **A** significantly, but we do not loose information about a single element in **A**. There is also another advantage in not creating a full coefficient matrix; the fact that it is not necessary to calculate all elements of **A**, but only the elements that we need to create **EIG**. Furthermore we are able to implement the algorithm using vectorization as opposed to using nested `for`-loops.

A representation of $\mathbf{A}^T$ is also needed. In Section 2.2 we show that multiplying with $\mathbf{A}^T$ is done by conjugating the diagonal matrix $\mathbf{\Lambda}$. However, when storing the data in the **EIG** structure there is a need for rearrangement of data. But instead of storing the data we simply take into account the unique rearrangement in each step of the multiplication.

### 6.1.2 New Multiplication

The multiplication of the `GMT3Operator` is based on FFT matrix-vector multiplication. The multiplication method depends solely on the bookkeeping. In Section 2.2 the FFT matrix-vector multiplication is described for a Toeplitz matrix. When operating on BTTB and $T^3$ matrices it is now only necessary to keep track of the Toeplitz matrices that make up each of these structures. In this section we will give an overview of the different things to keep track of during the multiplication.

We multiply the Toeplitz matrices of an entire BTTB block at a time. In Figure 6.3 we illustrate the representation of the BTTB blocks in the **EIG** structure for an example where $n_x = m_x = n_y = m_y = n_z = 3$ and $m_z = 4$. Below we

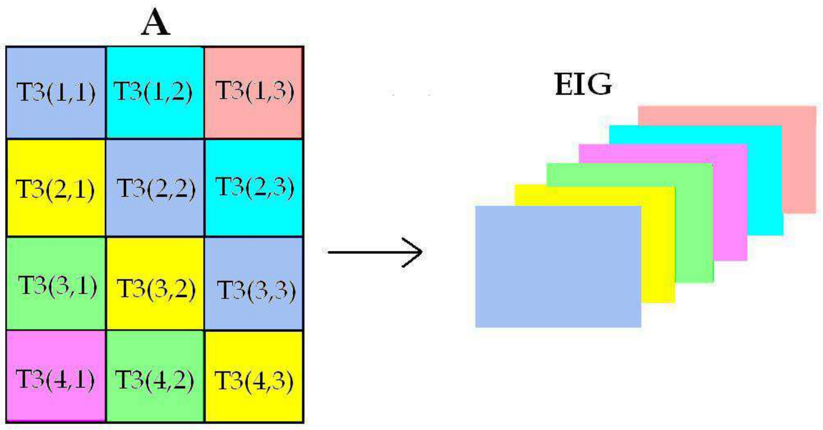


Figure 6.3: The bookkeeping visualized with respect to **EIG**

will list the pseudocode for the selection and bookkeeping of BTTB matrices from the $T^3$ structure. However, it is important to notice that the multiplication is performed using the **EIG** structure this means that when referring to e.g `T3`$(2,3)$ it translates into **EIG**(:,:,5) for the example in Figure 6.3.

Pseudocode for `multiplyT3`:

```
function [b] = multiplyT3(T3,x)
for i= 1:mz
  for j=1:nz
    b_sub = multiplyBTTB(T3(j,i),x((i-1)*nx*ny+1 :  i*nx*ny))
    b((j-1)*mx * my+1:j*mx*my) += b_sub
    end
  end
```

Now for the multiplication of the BTTB block with the corresponding subpart of the vector. Figure 6.4 illustrates a BTTB block on the left hand-side this could for instance be `T3`(2, 3) and on the right side of the figure the representation of the BTTB block in **EIG** in this case **EIG**(:,:,5). We have also illustrated the subpart of the vector (Figure 6.5). In the psuedocode `multiplyBTTB` (referred to in the first pseudocode of this section) the actual FFT multiplication is performed as described in Section 2.2. The only difference being that we perform several multiplications of the Toeplitz matrices and vectors simultaneously:

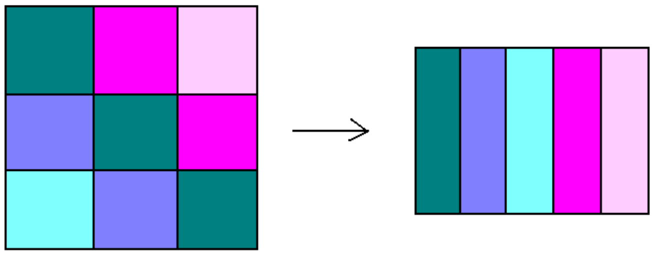


Figure 6.4: The bookkeeping visualized with respect to **EIG**$(:,:,i)$

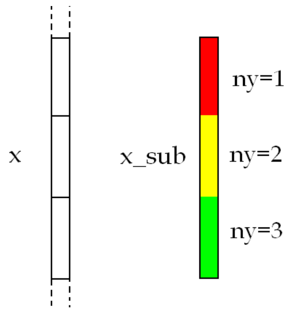


Figure 6.5: The subvector given as input to multiplyBTTB

Pseudocode for `multiplyT3`

```
function [b_sub] = multiplyBTTB(BTTB,x_sub)
z = fft([reshape(x_sub,nx,ny) ;  zeros(mx,ny)])
for i=1:ny
  vec = index of all columns of the BTTB representation (Toeplitz matrices)
       to be multiplied with the part of x_sub where ny = i
  z_sub = repmat(z(:,i),1,my)
  if (A)
    b_sub += (BTTB(:,vec).*z_sub)
  elseif ($A^T$)
    b_sub += (conj(BTTB(:,vec)).*z_sub)
  end
end
b_sub = ifft(b_sub)
b_sub = real(b_sub(1:mx,1:my))
b_sub = reshape(b_sub,mx*my,1)
```

### 6.1.3 Test of FFT matrix-vector multiplication

In this chapter we will perform a test of the implemented FFT matrix-vector multiplication. We will compare the results with the straight-forward matrix-vector multiplication.

We start out testing whether the multiplication is performed correctly. In the first test we multiply our representation of the coefficient matrix `GMT3Operator` with a solution vector. The result is then placed in a data object. We then convert the **EIG** structure of the `GMT3Operator` into a full structured matrix and extract the values from the solution object and place them in a vector. The matrix and vector is then multiplied using the straight-forward multiplication in Matlab (this is the multiplication method of the existing implementation). Finally the relative difference between the resulting vector and the data object is calculated. We use the same approach when testing the multiplication of $\mathbf{A}^T$ with the observations. In both cases the relative difference is in the order of magnitude of $10^{-16}$ which is the highest level of accuracy we can achieve on a computer.

## 6.2 Graphical Overview of Package

In this section we will give a graphical overview of the package and explain the basic use of it.

Figure 6.6 is a flowchart illustrating the input and output when using our

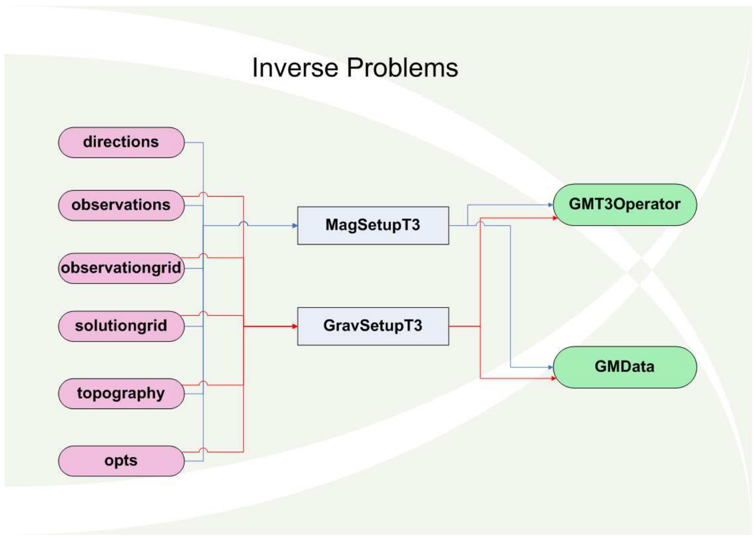


Figure 6.6: Flowchart showing input and output from the setup files.

two different setup files `MagSetupT3` and `GravSetupT3`. The `MagSetupT3` is used for magnetic problems and `GravSetupT3` is used for gravity problems. The flowchart is only for setup of inverse problems - for solution only and forward setups there would be small adjustments to the flowchart. The pink circles on the figure represent the input parameters/files, the blue squares represent the actual setup files, and the green circles represent the output objects. The input files are ASCII files and contain the information the `GravMagTools` package require to set up the problems. We will briefly explain the different input files.

• `directions file`
This file is only used by magnetic problems and it contains the inclination and declination of $\hat{\mathbf{j}}$ which is the direction of the magnetic dipole source. If the inclination and declination are identical to those of the induced core field the file is not necessary.

• `observation file`
The `observation` file will for a magnetic problem contain the inclination and declination of the induced core field. For inverse magnetic problems the file will also contain the observation points of the format $(x_k, y_k, z_k, \texttt{data}_k)$ where $z_k$ is the clearance above the topography. $\texttt{data}_k$ is the component of the magnetic field along the induced field. For a gravity setup the file will contain the observation points in the same format where $\texttt{data}_k$ is the vertical component of the

gravity field. The file will not contain any observation points if the problem is a forward problem for any of the setups.

• `solutiongrid file`
The `solutiongrid` file contains information about the number of grid points in the $x$, $y$, and $z$ direction in the solution domain and the location of the domain. Forward and solution only problems also include the solution in the file.

• `observationgrid file`
This file contains information about the observation grid. For inverse problems the number of observation points in the $x$, $y$, and $z$ direction and the location of the observation grid in the $x$ and $y$ direction. For forward problems the file will furthermore contain the $z$ bottom coordinate of the observationgrid.

• `topography file`
This file consist of the coordinates $(x_k, y_k, z_k)$ to the topography where $z_k$ is the height above sea level. If only one point is present, $z_1$, then a flat topography in $z_1$ is used. If the file is omitted a flat topography at sea level is used.

• `opts file`
The `opts` file controls the parameters of upward continuation. In the file the size of the total domain and the type of border is specified. If the file is omitted a default value of 512 for the size of the domain and 'sp0' (smooth extension of order 0) will be used.

In order to achieve an understanding of which steps the setup files executes before being able to return the objects we have in Figure 6.7 and 6.8 presented some, in some sense, more detailed versions of the chart in Figure 6.6. In Figure 6.7 the preliminary operations are performed. They consists of reading the input data. The routine which reads the topography also locates the solution points above the topography. These point are not valid solution points and will be set to 0 in the solution. The next step is testing whether the input meets the requirements of the software. If they are not met, an error is returned to the user. On the other hand if all requirements are met the interpolation (and upward continuation) is performed resulting in a new observation set. The chart continues in Figure 6.8. Here it is important to notice that the input files are not read twice in the setup file. In flowchart 6.8 the files are used to calculate the different fields of the output objects. The pink circles in this figure represent the objects that are returned to the user.

`GravMagTools` have two different methods for computing a large-scale regularized reconstructions Tikhonov and CGLS. Figure 6.9 presents a flowchart showing the input and output when using one of the two regularization methods. The input object are computed with `MagSetupT3` or `GravSetupT3`. Tikhonov require a regularizer object, `GMT3Regularizer`, if no regularizer object is given

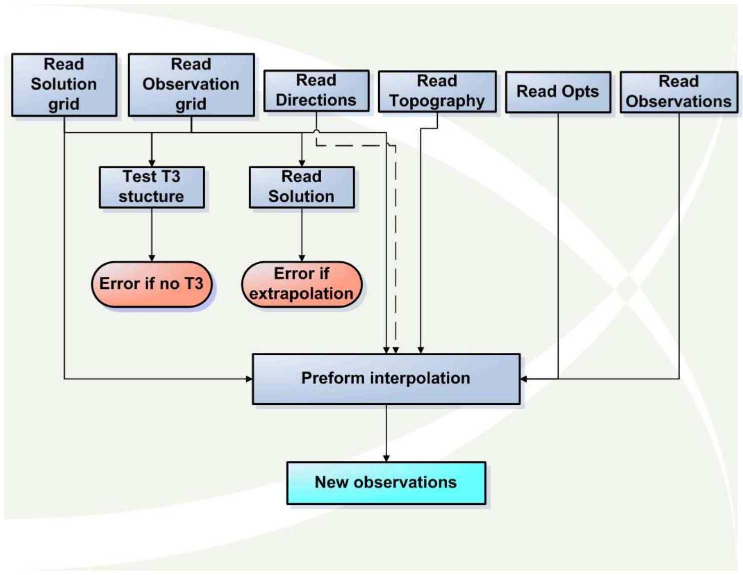


Figure 6.7: Flowchart showing a more detailed version of the first stage of the setup files.

the identity matrix is used. Furthermore the method requires a regularization parameter, $\lambda$. Tikhonov returns a `GMSolution` object or a VectorCollection of `GMSolution` objects. A VectorCollection is a M$\mathcal{OO}$ReTool object used to store many solution object. The CGLS method takes as additional input the number of iterations to be performed and returns a `GMSolution` object. It is possible to give several numbers and the method will then store the solutions in a VectorCollection.

In this section we have described the design of the new objects and we have seen how some of these are associated with uniquely designed multiplication functions. These functions are overloaded operators which ensures correct multiplication and at the same time they ensure the user friendliness that is essential for the package. We design several different overloaded functions for instance the `show` function. The `show` function works on both the `GMSolution`, `GMData`, and `GMT3Operator`. In appendix D we have listed the routines of the new implementation of the `GravMagTools` package. For a more detailed description we have implemented a help topic in Matlab for each of the routines.

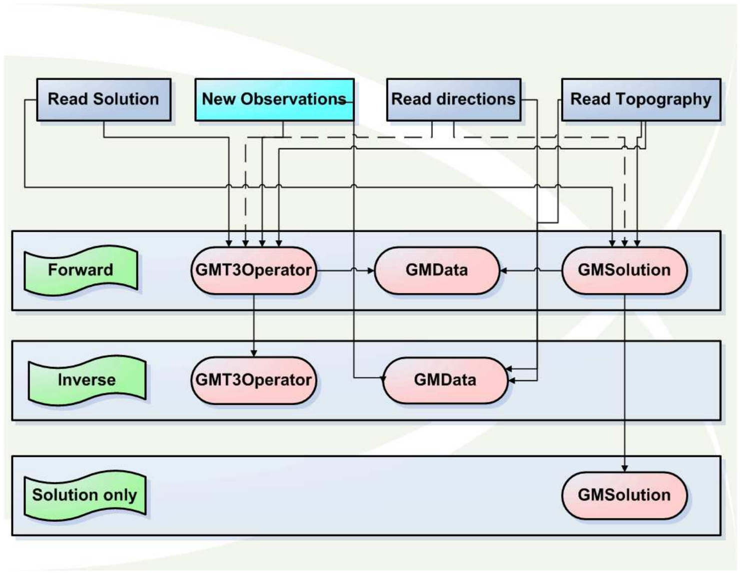


Figure 6.8: Flowchart showing a more detailed version of the second stage of the setup files.

## 6.3 Performance

We set up a forward gravity surveying problem in order to test the new implementation of the `GravMagTools` package versus the existing implementation. We conduct a series of tests to illustrate the time and memory consumption for the two implementations

After setting up the problem the system an inversion is conducted using the CGLS solver implemented in the M$\mathcal{OO}$ReTools package ([4]). We recall how the CGLS solver performs multiple matrix-vector multiplications in order to calculate the solution, this way the efficiency of the newly implemented multiplication should become apparent. The only differences between the new implementation and the existing one that could effect the performance of the package is the representation of $\mathbf{A}$ and the matrix-vector multiplication, so by performing tests of various sizes we will have an indication of our implementation versus the existing implementation.

The number of CGLS iterations is set to 10. This number is very low, but for now we are not interested in the quality of the reconstruction, but rather that it does not differ from the corresponding reconstruction of the thoroughly tested existing implementation of the package. We set the tolerance level of CGLS to

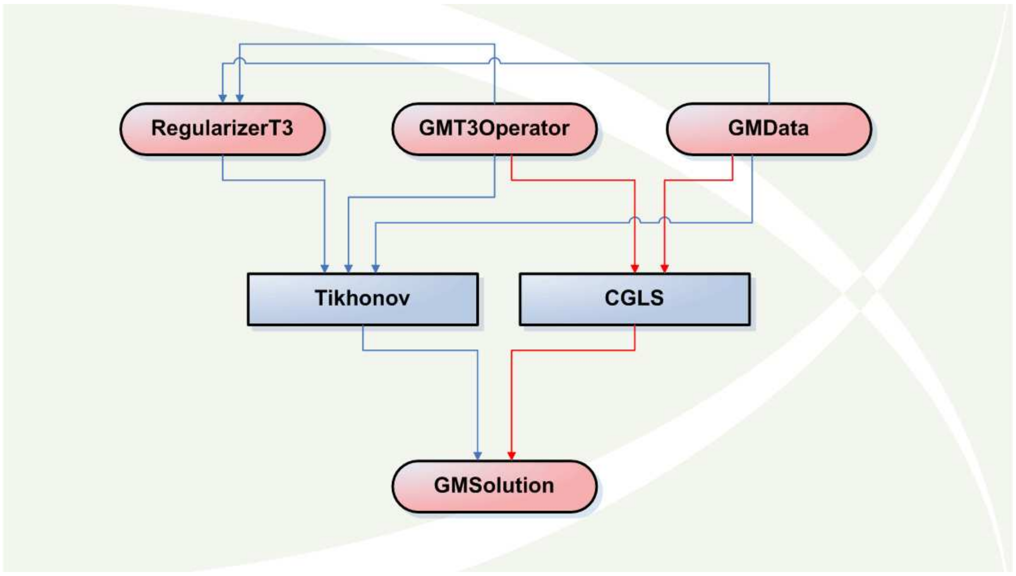


Figure 6.9: Flowchart showing the use of our regularization methods.

$10^{-6}$. For this test problem this choice ensures that the full number of iterations is performed. Throughout the tests only the solution- and the observation grid will change in size. Before commenting on the tests we list the results in Table 6.5 and 6.6 and we note that our solutions for the different grid sizes were equal to those calculated using the full structure. The time consumptions in

| nx = mx = ny = my | nz = nz | time (**A**) | time (**m**) | # elements (full **A**) |
|---|---|---|---|---|
| 10 | 5 | 1.42 | 0.18 | 2 MB |
| 15 | 5 | 5.01 | 1.61 | 10.13 MB |
| 15 | 10 | 18.31 | 6.46 | 40.5 MB |
| 20 | 10 | 53.70 | 21.29 | 128 MB |
| 20 | 15 | 126.89 | 50.67 | 288 MB |
| 25 | 20 | N/A | N/A | 1250 MB |
| 50 | 20 | N/A | N/A | 20000 MB |
| 100 | 25 | N/A | N/A | 500000 MB |

Table 6.5: `GravMagTools` with the existing representation of **A** and the straightforward multiplication. Where time (**A**) is the number of seconds it takes to create **A** and time (**m**) is the number of seconds it takes perform 10 CGLS iterations.

| nx = mx = ny = my | nz = nz | time (**EIG**) | time (**m**) | memory (**EIG**) |
|---|---|---|---|---|
| 10 | 5 | 0.10 | 0.70 | 0.054 MB |
| 15 | 5 | 0.13 | 1.10 | 0.126 MB |
| 15 | 10 | 0.22 | 4.18 | 0.264 MB |
| 20 | 10 | 0.29 | 6.63 | 0.474 MB |
| 20 | 15 | 0.40 | 14.49 | 0.724 MB |
| 25 | 20 | 0.70 | 41.18 | 1.528 MB |
| 50 | 20 | 1.73 | 227.92 | 6.178 MB |
| 100 | 25 | 6.16 | 2421.30 | 31.204 MB |

Table 6.6: `GravMagTools` with new representation of **A** and new multiplication. Where time (**EIG**) is the number of seconds it takes to create **EIG** (the new representation of **A**) and time (**m**) is the number of seconds it takes to perform 10 CGLS iterations.

Table 6.5 and 6.6 are visualized in Figure 6.10. In the figure we see how the straight-forward multiplication marked with a blue line is faster for small problems than the new implementation marked with a green line. However, when the problems grow in size the graphs intersect and the **EIG** structure with the new multiplication performs much faster. In the subplots of Figure 6.10 the red points marks where the existing implementation run out of memory and is no longer able to perform the test. The new implementation did not at any point in the test run out of memory, however the time consumption grew exponentially in size. Hence the iterative regularization methods of the package will (for most problems) run faster than the existing implementation. The last thing we want

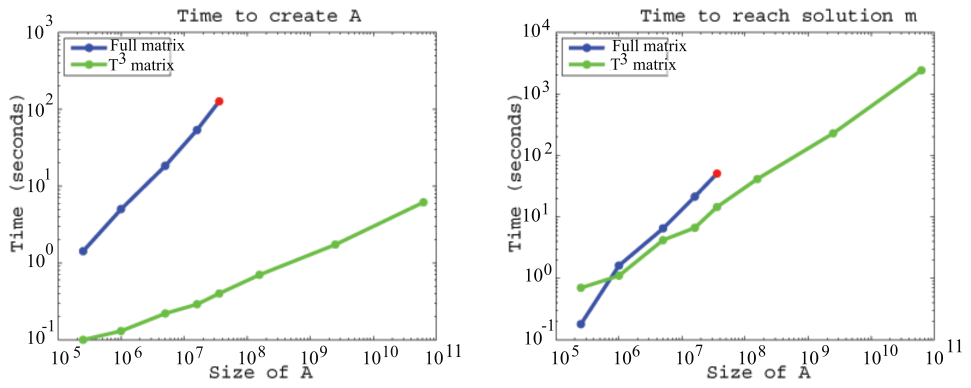


Figure 6.10: Time consumptions from our test.

to visualize from the tables is the memory consumptions for the two methods compared in a coordinate system. This is done in Figure 6.11 where it is becomes obvious why Matlab runs out of memory when solving the system using the existing objects.

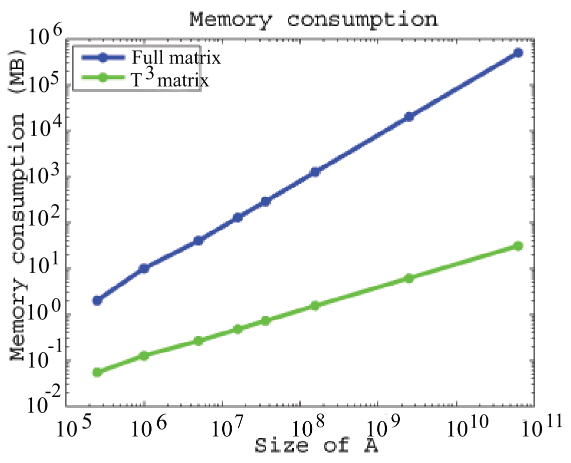


Figure 6.11: Memory consumptions from our test.

So in theory it is possible to set up large-scale problems using the new implementation. This conclusion is supported by our performance test that prove that not only can we setup up the large-scale problems we can also do it within a reasonable time consumption. Furthermore the new multiplication was faster than the straight-forward multiplication.

Large-scale problems are difficult to setup and reconstruct in Matlab because only a limited virtual memory is available. On 32-Bit architecture systems[5] 4 GB addressable memory is available. The operating systems reserve the upper 2 GB of address space, thereby limiting the available virtual memory to 2 GB. Upon launch of Matlab additional 0.8 GB is used. In practice this limits the available virtual memory to 1.2 GB. In addition to this Matlab requires that each variable is stored in a contiguous block of virtual memory. This means that when for instance a coefficient matrix is created at a particular point in time we are limited by the amount of contiguous free virtual address space.

Assuming that we have 1 GB virtual memory available at a time of creating a coefficient matrix, we have then in Table 6.7 listed the theoretical sizes for the solution- and observation grid (which we here assume to be identical) and the memory consumption for the full and the compressed coefficient matrix. Each element of the full structure is saved using double precision requiring 8 bytes of virtual memory whereas each element if the **EIG** structure is complex and

[5]The operating systems at IMM (unix) and on our private PC's (windows) are all 32-Bit architecture systems.

requires 16 bytes of memory to be stored in double precision. We have in both cases fixed $n_z = m_z = 12$ and the difference is astounding. The theoretical largest $T^3$ matrix would if stored in full structure take up $5.2 * 10^5$ GB virtual memory!

| Type | $n_x = n_y = m_{x'} = m_{y'}$ | $n_z = m_{z'}$ | Memory |
|---|---|---|---|
| Full | 30 | 12 | $\simeq 1$ GB |
| $T^3$ | 820 | 12 | $\simeq 1$ GB |

Table 6.7: Largest theoretical possible coefficient matrix (full and compressed).

Chapter 7

# Inversion of Data

The theory and tools are now in place and we are able to perform the inversions. In this chapter we wish to find a good setup for these type of problems. Then we will experiment with inversions using both simulated and upward continued multi-level data. Finally we will use the conclusions to perform large-scale inversions. Throughout the chapter we test using the 3-D magnetic setup with a gaussian source. The same tests will be performed using a dipole source and if nothing else is stated in the test, the findings match those of the gaussian source. The same goes for the gravity setup.

In parts of this chapter we will use tools from the `REGULARIZATION TOOLS` [6]. These tools are not intended for large-scale problems, but they are used in order to achieve an understanding of the problems before we are able to perform large-scale inversions.

## 7.1 Test of Setup

In order to achieve the best possible inversions we set out to find a good setup. In this section we experiment with different types of setups and use SVD analysis as a tool to find the best overall setup.

### 7.1.1 Number of Data Levels

In Chapter 3 we saw how the 3-D geometry behind the surveying models led to the $T^3$ structure. A special case of the $T^3$ structure is achieved when using only one observation level, this structure is called blockwise BTTB structure. We want to investigate whether there is a difference between the case using one level and using 3 observation levels for different dimensions of the solution domain.

We know that ill-posed problems can be hard to solve, so in this section we investigate if the setup is of influence with regard to the ill-posedness. We use SVD as the analysis tool and we only consider the coefficient matrix. No inversions are performed in this section. For a number of different setups we calculate the condition number of the coefficient matrices

$$\texttt{cond}(\mathbf{A}) = \frac{\sigma_{\texttt{max}}}{\sigma_{\texttt{min}}} \ . \tag{7.1}$$

But this is not sufficient information when investigating the level of ill-posedness. We also need to consider the development of the singular values. This process is a deciding factor when determining how many components that can be used in the reconstruction. In Figure 7.1 we illustrate two examples with the exact same condition number, but different development in the singular values. Assuming that the right hand-side **b** and the noise level are identical a significantly larger amount of components can be used in the example on the right hand-side.

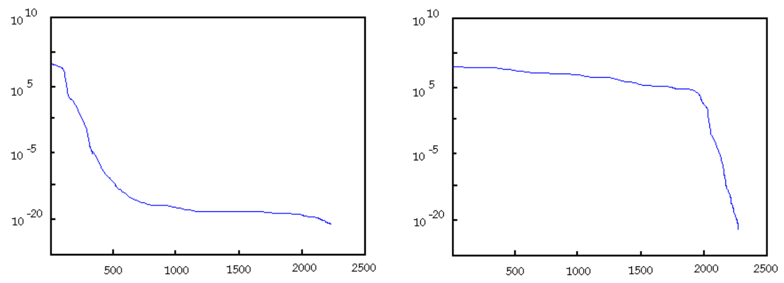


Figure 7.1: Two different developments in singular values for two coefficient matrices with the same condition number.

We set up two problems both approximately square. The first problem only contains one observation level with $64 \times 64$ grid points, the second problem contains three observation levels each of the levels is $37 \times 37$. For both solution grid and observation grid the grid spacings $\Delta x = \Delta y = \Delta z = 0.5$. For each setup we create four different solution grids illustrated in Figure 7.3 and listed

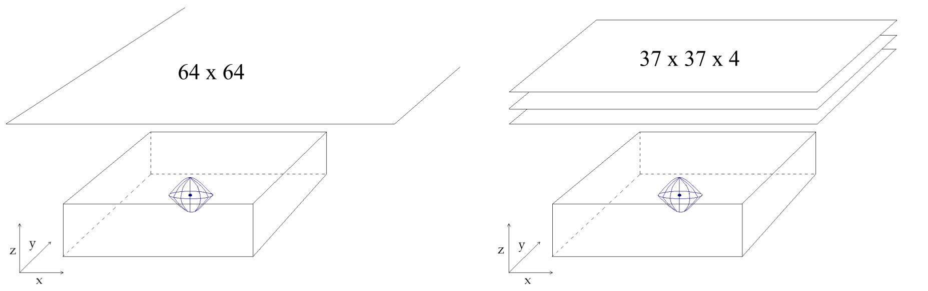


Figure 7.2: On the left hand-side the blockwise BTTB structure and on the right hand-side the $T^3$ structure.

in Table 7.1 and Table 7.2. In these tables we list the decrease in the singular values for each system in the test. When talking about the decrease we mean the span of the singular values in a specified range (for instance 1 to 1000). In Figure 7.4 the singular values are plotted for the solution grid of $36 \times 36 \times 4$. On the left hand-side the blockwise BTTB and on the right hand-side the $T^3$ structure.

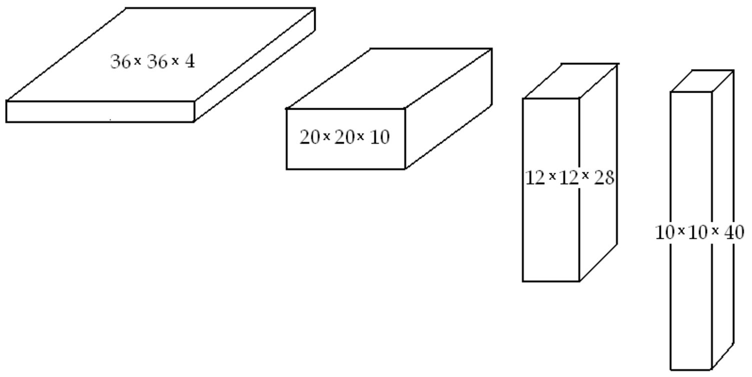
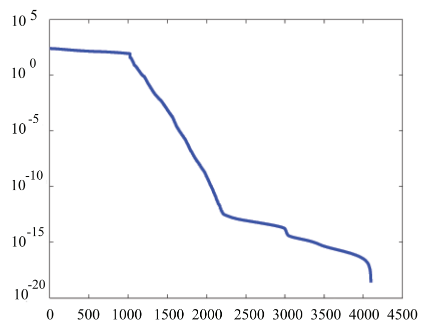


Figure 7.3: The different solution grids.

From Equation (5.3) we know the singular values are a non-increasing sequence. Hence the first singular values are least likely to be affected by errors. For this reason we are interested in a small decrease in the large singular values so they can contribute to the solution. When comparing Table 7.1 and 7.2 we see that the $T^3$ structure is preferable over the blockwise BTTB. This statement is based on the fact that the singular values decrease slower in Table 7.2 than in Table 7.1.

There are several conclusions to be drawn from this line of testing. The primary conclusion is that when working with the $T^3$ structure we should work



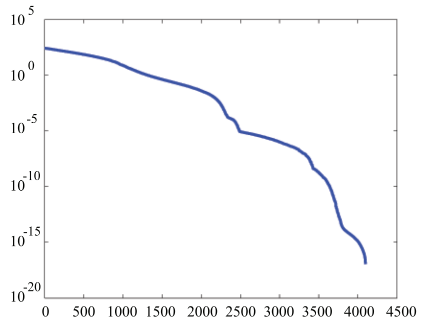


Figure 7.4: The singular values for the blockwise BTTB structure on the left hand-side and the $T^3$ structure on the right hand-side.

| Solution grid | decrease $\sigma_i$ 1-1000 | decrease $\sigma_i$ 1001-2000 | decrease $\sigma_i$ 2001+ |
|---|---|---|---|
| $36 \times 36 \times 4$ | $10^1$ | $10^{11}$ | $10^9$ |
| $20 \times 20 \times 10$ | $10^9$ | $10^7$ | $10^4$ |
| $12 \times 12 \times 28$ | $10^{15}$ | $10^3$ | $10^3$ |
| $10 \times 10 \times 40$ | $10^{16}$ | $10^2$ | $10^3$ |

Table 7.1: One observation level: Decrease in singular values for the different solution grids.

| Solution grid | Decrease $\sigma_i$ 1-1000 | Decrease $\sigma_i$ 1001-2000 | Decrease $\sigma_i$ 2001+ |
|---|---|---|---|
| $36 \times 36 \times 4$ | $10^2$ | $10^2$ | $10^{15}$ |
| $20 \times 20 \times 10$ | $10^4$ | $10^9$ | $10^7$ |
| $12 \times 12 \times 28$ | $10^{13}$ | $10^5$ | $10^2$ |
| $10 \times 10 \times 40$ | $10^{16}$ | $10^2$ | $10^2$ |

Table 7.2: Three observation levels: Decrease in singular values for the different solution grids.

with multi-level observations sets[1]. Working with only one observation level when using the $T^3$ structure leaves us vulnerable to any kinds of error whether that be measurement errors or interpolation errors. It is however, important to notice that we are still vulnerable to error working with multi-levels observation sets, but in a less degree than if using one observation level. Furthermore we know from the performance section of Chapter 6 that the $T^3$ structure requires

[1]However, under different conditions one observation level has proven good, see [3].

least memory. Another conclusion from this investigation is that when working with square systems, it is preferable to work with solution layers and observation levels of approximately the same size.

### 7.1.2 Cell Proportions

In the previous test the grid spacings for the three directions was chosen to be the same, i.e. $\Delta x = \Delta y = \Delta z = 0.5$. We have chosen to make it a requirement that $\Delta x = \Delta y$ thus when experimenting with the cell proportions we will only vary $\Delta z$. For the $T^3$ structure using three observation levels we fix $\Delta x = \Delta y = 0.5$ and then the singular values for three different values of $\Delta z$ are calculated - one larger (0.75), one identical (0.5) and one half size (0.25). In Figure 7.5 the visual result is illustrated and it is clear that the squeezed cells where $\Delta z = 0.25$ yields the best result. The same test has been performed for all the solution grids from the test in Section 7.1 and the conclusion for each test is identical to the conclusion of this first test.

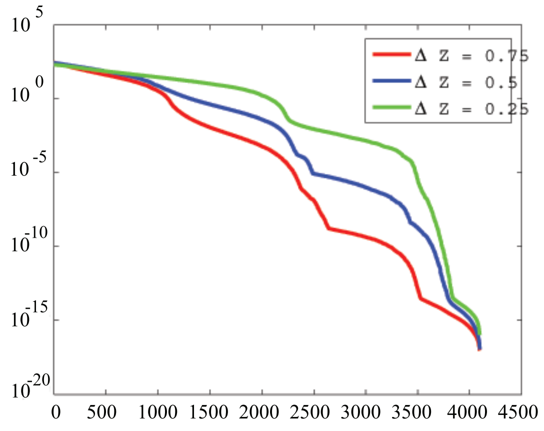


Figure 7.5: The singular values for three different values of $\Delta z$. For the red line $\Delta z = 0.75$, for the blue and green line $\Delta z = 0.5$ and $\Delta z = 0.25$, respectively. $\Delta x$ and $\Delta y$ are both fixed to 0.5.

### 7.1.3 Distance between Observation grid and Solution grid

The final part of the setup is a test that describes the importance of the placement of the observation and solution grid respectively. For this purpose we set up a forward problem where the observation and solution grid both contains $15 \times 15 \times 6$ grid points. We then place the two grids in 4 different distances and calculate the coefficient matrix. For each matrix we take out the singular values and plot them, the result can be seen in Figure 7.6. The placement of the solution grid is fixed and the observation grid is then placed at a distance of 10 meters, 400 meters, 800 meters, and 1.2 km respectively. From the decrease of the singular values it is clear to see that the problem is less ill-posed when the two grids are closest together.

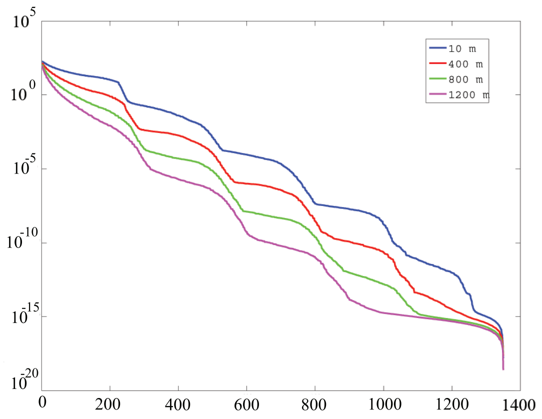


Figure 7.6: The singular values the different placements of the solution grid. The blue line represents the singular values for a distance of 10 meters, the red 400 meters, the green 800 meters, and finally the magenta 1200 meters.

To sum up the tests of this section we have seen that we achieve a good setup using multi-level observation sets with squeezed cells in the $z$ direction. Furthermore the observation and solution grid has to be approximately the same size and placed close together.

## 7.2 Convergence of Regularization Methods

Throughout this chapter inversions will be performed using some of the regularization methods described in Chapter 5. In this section we will invert the same setup using TSVD, CGLS, and Tikhonov, respectively.

We set up a square system containing $32 \times 32 \times 4$ grid points in both solution

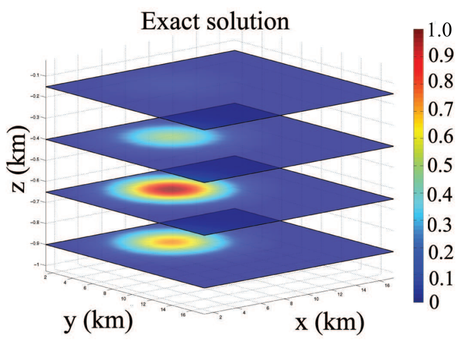


Figure 7.7: The exact solution.

and observation grid where $\Delta x = \Delta y = 0.5$ and $\Delta z = 0.25$. The gaussian source is placed off set from the center in the $x$ and $y$ direction and half way down the solution domain as shown in Figure 7.7 which illustrates the exact solution. We now perform the inversions using the methods as listed in Table 7.3. The table lists the specifications for each of the inversions. The time consumption in column 4 is the time to finish the inversion on the Solaris Sunfire server at IMM. In Figure 7.8 the reconstructions for the three different regularization methods Tikhonov, CGLS, and TSVD are shown. The inversions are all, in some sense, close to the exact solution in figure 7.7. At the same time they are also visually alike even though they do not convert towards the same solution.

| Method | Tolerance | # iter. | Time (sec) | Other | Fig. 7.8 |
|---|---|---|---|---|---|
| Tikhonov | $10^{-6}$ | 100000 | $5.1 * 10^4$ | $\lambda = 1.2 * 10^{-9}$ ($\sigma_{3700}$) | Top left |
| CGLS | $10^{-8}$ | 17939 | $8.4 * 10^3$ | N/A | Top right |
| TSVD | N/A | N/A | 1.5 | k = 3700 | Bottom |

Table 7.3: Methods used to perform inversion of the setup.

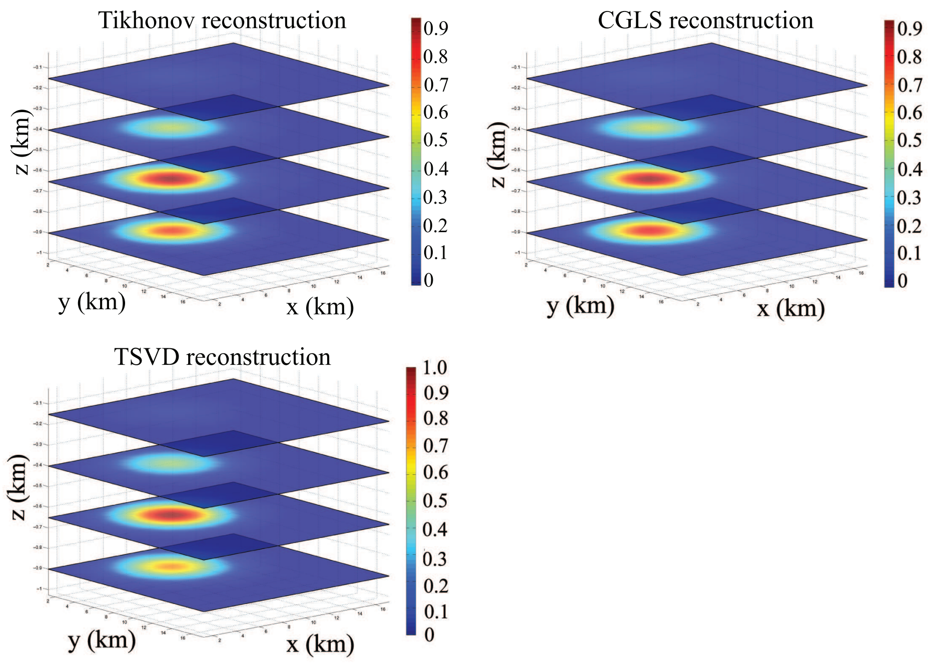


Figure 7.8: The solutions for the three different regularization methods. On the top Tikhonov and CGLS and on the bottom of the figure TSVD.

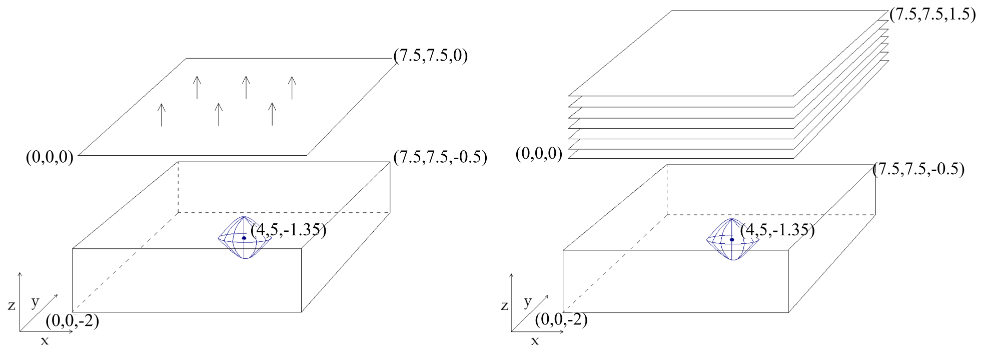


Figure 7.9: On the left hand-side the first setup using upward continuation. On the right hand-side the second setup using solely simulated data.

## 7.3 Investigations of Upward Continuation

In Section 4.1.2 we saw that given the right conditions, upward continuation created an observation set that corresponds well to the simulated data. However we also saw that small changes in the conditions could have damaging effects on the upward continued data. In this section we perform inversions using upwards continued data.

We set up two forward problems, in both problems the source is gaussian and placed in the exact same location and the solution grids are identical. The observation and solution grid are placed 500 meters apart in order to limit the relative error between the upward continued set and the simulated data while still achieving an acceptable decrease of the singular values. The observations for the first problem are simulated in one level (to imitate a real data set). This data set is then upward continued to a total of 6 levels (we provide the best condition placing a natural border of 7 data points around the data to be used in the upward continuation). The second observation set is simulated in 6 levels with the same dimensions and placement of the first set. Figure 7.9 illustrates the two setups. The relative difference between the first and the second observation set is 0.8%. We now perform inversions of the two setups using Tikhonov[2]. In Figure 7.10 the exact solution is shown on the top to the far left while the two inversions are illustrated on the top to the right (the inversion using upward continued data) and on the bottom (the inversion using the simulated data). The best inversion is clearly the solution calculated using the simulated data. To be sure that the difference is not caused by the upward continuation being performed in too many levels we perform an inversion using 3 data levels and the result is once again no depth resolution - for this reason

[2]The best regularization parameter $\lambda$ is found using a trial-and-error method.

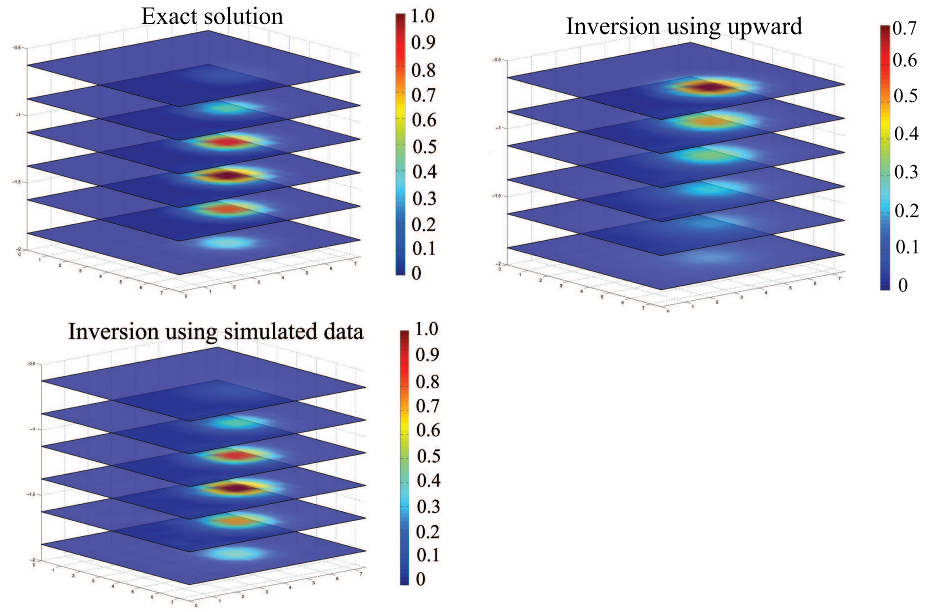


Figure 7.10: The top left figure is the exact solution. The inversion using the upward continued data in 6 levels is shown on top to the right and at the bottom the inversion using the simulated data in 6 levels.

we have chosen not to include the reconstruction.

### 7.3.1 SVD Analysis of the Test Problem

Now we investigate why the inversions using upward continued data are so different from the one using simulated data even though the vectors are almost identical. To understand why there is no depth resolution we perform an SVD analysis of the problem. On the left hand-side in Figure 7.11 we can see the Picard plot of our test problem using upward continued data. The Picard condition is violated after 300 components and this means that only these may contribute to the regularized solution. (We will return to the right hand-side of the plot later in this section.) We now take a closer look at the singular vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ for the two setups. From [8] we know that in 1-D the higher the index $i$, the more oscillations $\mathbf{u}_i$ and $\mathbf{v}_i$ have. To illustrate that the same holds true in 3-D we plot the $1^{\mathtt{st}}$, $3^{\mathtt{rd}}$, $6^{\mathtt{th}}$, and $9^{\mathtt{th}}$ left singular vectors in Figure 7.12 and in Figure 7.13 the $1^{\mathtt{st}}$, $300^{\mathtt{th}}$, $600^{\mathtt{th}}$, and $750^{\mathtt{th}}$ right singular vectors.
Figure 7.12 and 7.13 clearly shows that the higher the index the more os-

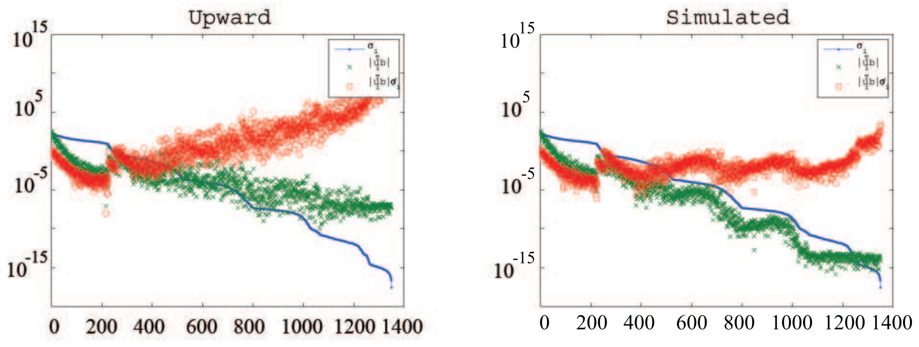


Figure 7.11: Picard plots of the two setups with 6 observation levels.

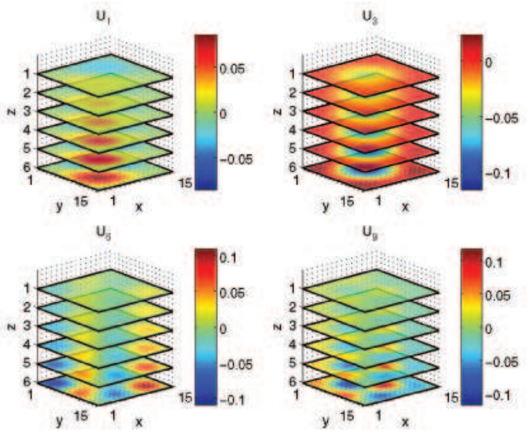


Figure 7.12: The 1st, 3rd, 6th, and 9th left singular vectors $\mathbf{u}_i$ of coefficient matrix.

cillations in the vectors. In Figure 7.12 the largest values of the left singular vectors is placed in the bottom of the sliceplot. This is logical because the left singular vectors are the expansion of the right hand-side $\mathbf{b}$ which contains the observation values and we know that the strongest observation values are located closest to the source. If we look at the right singular vector $\mathbf{v}_i$ in Figure 7.13 we see that when $i = 1$ the highest values are placed in top of the plot while they for $i = 300$, $i = 600$, and $i = 750$ moves down. This observation leads to an important tool when discussing depth resolution namely the Depth Resolution Plot[3]. The DRP is according to [3] calculated by rearranging every

[3]In the rest of this report the Depth Resolution Plot will be referred to as DRP

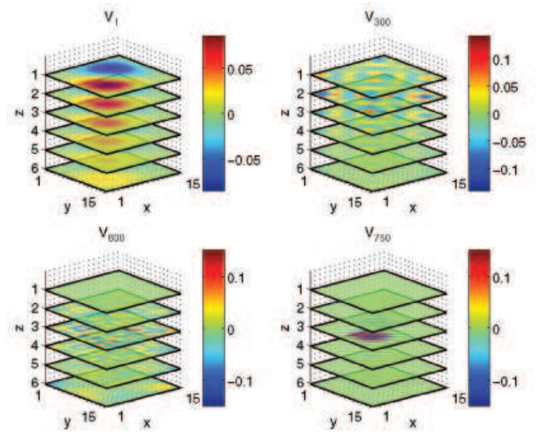


Figure 7.13: The 1st, 300th, 600th, and 750th right singular vectors $\mathbf{v}_i$ of coefficient matrix.

$\mathbf{v}_i$ vectors to a 3-D box[4]. The 2-norm is then calculated for every layer in these boxes, placed in a vector and plotted. We can in the DRP plot see where the $i^{\mathtt{th}}$ right singular vector contributes most to the solution. In [3] it is shown that the more components we are able to include the more depth resolution can be obtained. Figure 7.14 illustrates the DRP of the test problem. From the Picard plot in Figure 7.11 we know that we can trust the first 300 components, and when applying this information to the DRP in Figure 7.14 we can see that this give us sufficient information to reconstruct 2 layers; which also is what we see in the reconstruction in Figure 7.10 (top right plot).

On the right hand-side of Figure 7.11 we have plotted the Picard plot for the simulated data in 6 levels. The plot tells us that the Picard condition is violated after approximately 1000 components and when we look at the DRP in Figure 7.14 we can see it means we can reconstruct to layer 5. This is consistent with the reconstruction on the bottom of figure 7.10 it is clear how this plot is much closer to the exact solution than the inversion using upward continued data. This implies that the error introduced by the upward continuation is cause of the lack of depth resolution. Thus we investigate the error thoroughly in the following.

[4]It is important that this box have the same ordering as the solution would have if reshaped into a 3D box

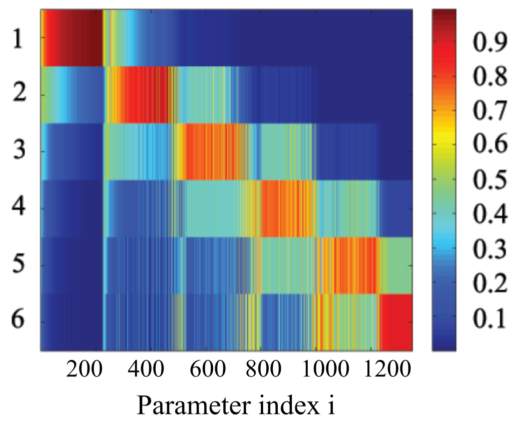


Figure 7.14: The Depth Resolution Plot of the simulated magnetic test problem.

### 7.3.2 Error Behavior in Upward Continuation

In this section the errors introduced by the upward continuation is examined. A forward problem like in Section 7.3 is set up. Hence we create a forward problem and from the bottom level we perform an upward continuation in 5 additional levels. The exact observation level from the forward problem is compared to the upward continued levels. The error of upward continuation is plotted in Figure 7.15 for each level. The figure shows how the error from level 1 to level 5 appears to change into something that looks like a signal in the upward continuation. The small difference in level 0 is cause by numerical errors. The upward continuation functions as a lowpass filter. That is the method passes low frequencies well and reduces high frequencies. White noise consists of high frequencies, but in our case the noise frequencies are low (due to the upward continuation) and for this reason it is not possible to separate the noise from the actual signal.

We now plot $|\mathbf{u}_i^T \mathbf{e}_{\mathtt{error}}|$ in order to see how the upward continuation error effects singular values. On the left hand-side in Figure 7.16 $|\mathbf{u}_i^T \mathbf{e}_{\mathtt{error}}|$ is plotted for the upward continuation error from our test problem. On the right hand-side we have created white noise in the same order of magnitude as the upward continuation error. In these figures it becomes quite clear how the specific noise of the upward continuation effects the large singular values in a way that white noise never would.

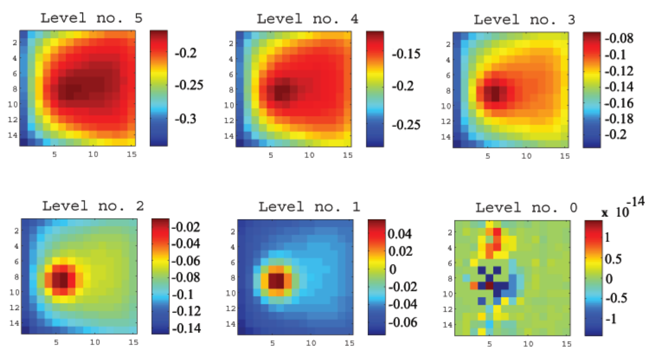


Figure 7.15: The error of the upward continuation compared to the exact observations from the forward problem.

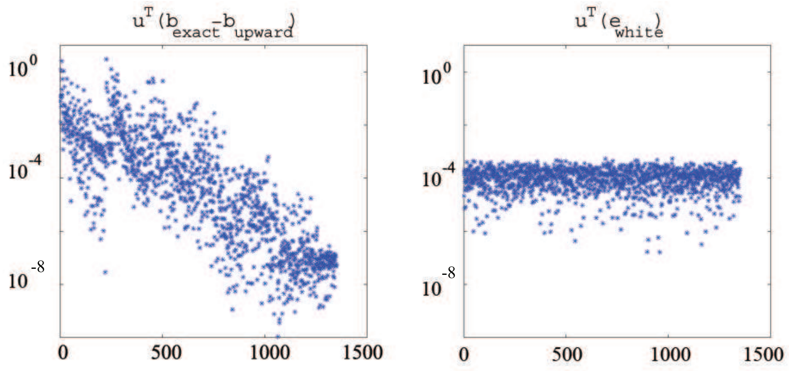


Figure 7.16: To the left is the plot of $|\mathbf{u}_i^T \mathbf{e}_{\texttt{upward}}|$, and to the right $|\mathbf{u}_i^T \mathbf{e}_{\texttt{whitenoise}}|$ where the noise level is $10^{-5}$.

### 7.3.3 Summary

These investigations lead to the rejection of the Fourier based upwards continuation for these type of problems using the $T^3$ structure. The main problem of the method is that it is a filtration and not a physical model. According to [12] the Fourier based transformation would be a physical model if the integrated area was infinite. However, this is not possible. The resulting error is developed into a signal that is unseparable from the actual signal of the measurements. Hence the largest singular values are affected causing bad inversions. The derivation and implementation of a new upward continuation model is out of the scope of this project. Thus from this point on we will only perform inversions using simulated data.

## 7.4 Box Solver

We have seen how we can obtain the best reconstructions using squeezed cells. But this means that we only can achieve a limited depth of the solution grid unless a very large number of grid points are used in the $z$ direction. For this reason we present the outlines of a solver technique denoted a box solver.

In Figure 7.17 we illustrate the basic idea behind this solver. The four different solution domains are the four boxes (1 to 4) and they are overlapping as illustrated in the figure. First we setup the problem including only box 1, then box 2, and so on. When inverting system 1-4 the reconstructions can be merged in an appropriate manner as shown to the far right in the figure.

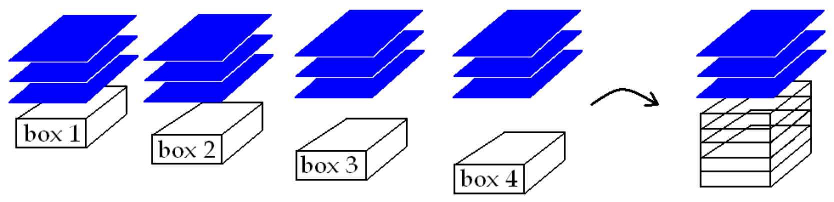


Figure 7.17: Illustration of the box solver.

For this test we choose to use the SVD analysis as a tool and TSVD as the regularization method to perform the inversions. We will look at each inversion separately and we will not perform the actual merging. We design a forward problem where the observations are 3 levels of $37 \times 37$ data points. The total solution domain is $32 \times 32$ in 10 layers and each of the boxes are $32 \times 32$ in 4 layers. Thus each of the systems to be inverted are approximately square. On the left part of Figure 7.18 the exact solution is visualized and the colorbar is set to [0 1]. In the following inversions the colorbars are set to [0 3]. On the right part of Figure 7.18 we therefore visualize the exact solution once again with a colorbar of [0 3]. This is done in order to ensure that the exact solution is visually comparable with the inversions. The source is gaussian and placed approximately in the 4th-6th layer. The 4 different solution domains are now created, the systems with the coefficient matrices and the right hand-side are

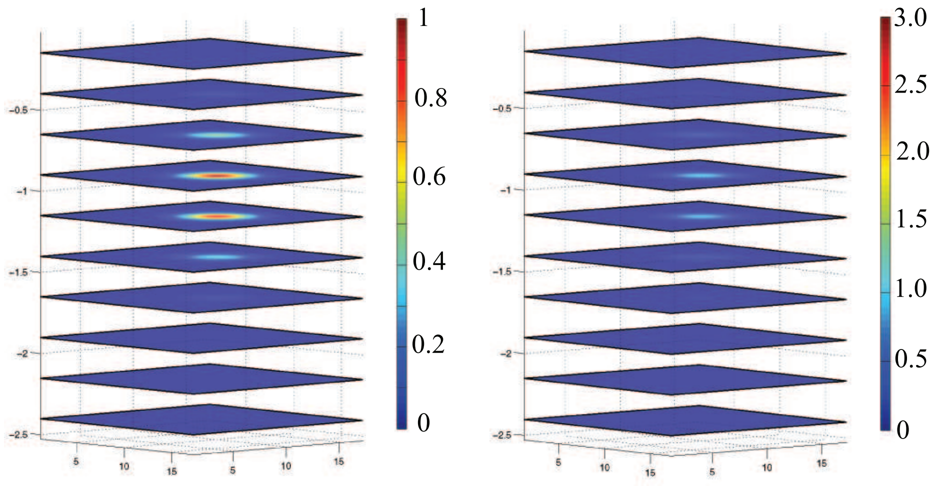


Figure 7.18: On the left side the exact solution with a colorbar set to [0 1]. On the right hand-side the exact solution with a colorbar set to [0 3].

set up. Below the solution domains are listed and opposite to each domain we list which of the layers the domain cover with respect to the solution domain as seen in Figure 7.18.

Box 1: Layer 1-4
Box 2: Layer 3-6
Box 3: Layer 5-8
Box 4: Layer 7-10

Figure 7.19 shows the Picard plots for each of the four systems. From the Picard plots it is clear to see that the first box from the top is the one that achieves the best depth resolution. About 3000 SVD components can be used before the Picard condition is no longer met. For the second box from the top about 3000 components can be used. In the third box approximately 750 components and in the fourth box the Picard condition is seemingly not met at all.

TSVD regularization method is now used for the actual inversion. For each box we perform 3 inversions adjusting only the truncation parameter. In Figure 7.20 the inversions of the first box from the top. For the middle plot using a truncation parameter of 2000 and it resembles the source. However, the intensity of layer 3 is a bit too high compared to the third layer of Figure 7.18, this is maybe not all that strange seeing how the main part of the source is actually not placed in this box. The second box from the top is the box in which most of the source is located. This makes it all the more strange that the method actually

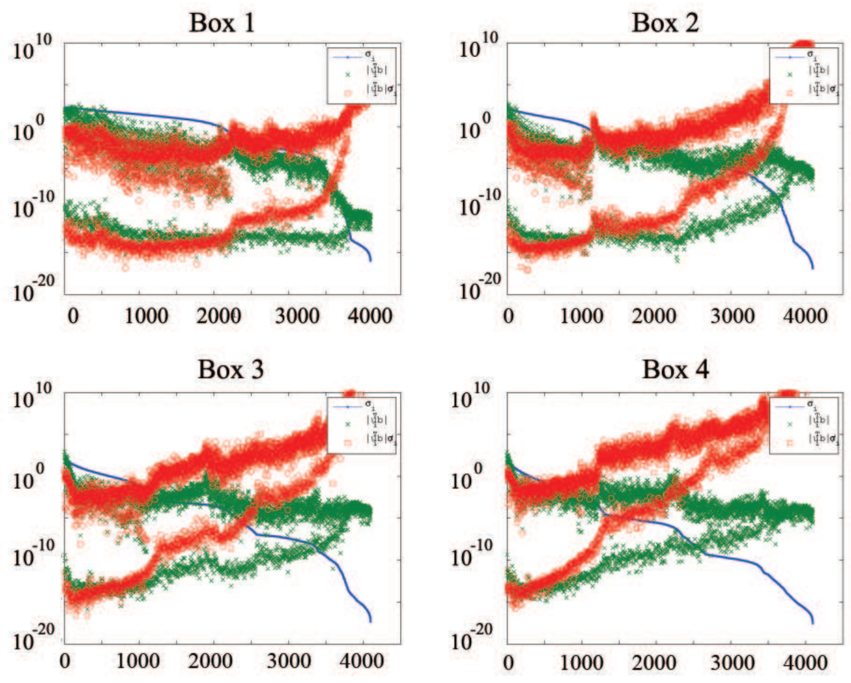


Figure 7.19: The Picard plots for each of the 4 boxes.

does not seem to reconstruct the source in this box. This is seen for the three inversions of Figure 7.21. In the third box the method places the source at the top of the domain - see Figure 7.22. Once again the method places too much magnetization in the domain, but this is understandable because the source is actually only partially placed in this box. We have chosen not to include the inversions of the fourth box because it does not meet the Picard condition for any number of components.

Based on this experiment we conclude that this way of reconstructing in

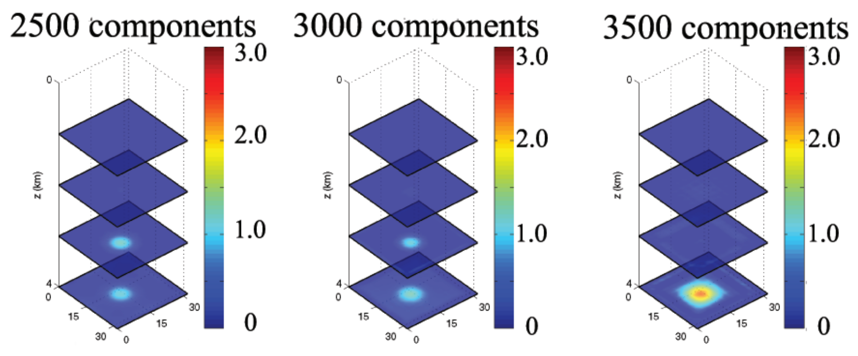


Figure 7.20: The inversions for the first box from the top.

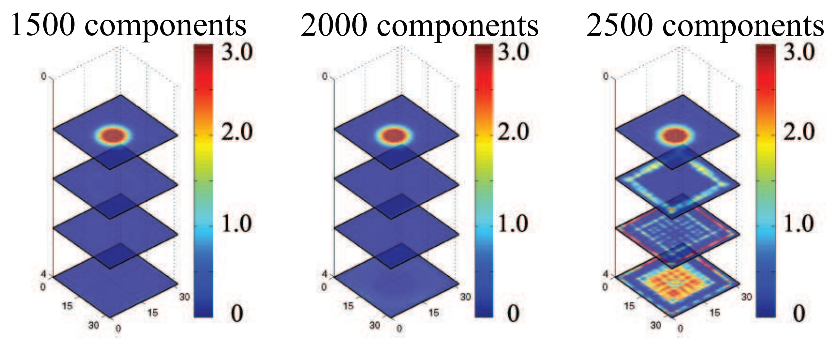


Figure 7.21: The inversions for the second box from the top.

order to achieve depth resolution would need some adjusting. Moreover the merging of domains must be done with great care and it would take a lot of experiments in order to design a stable algorithm to perform this type of inversion. This section has just been an outline and we decide not to include this type of solver in the package.

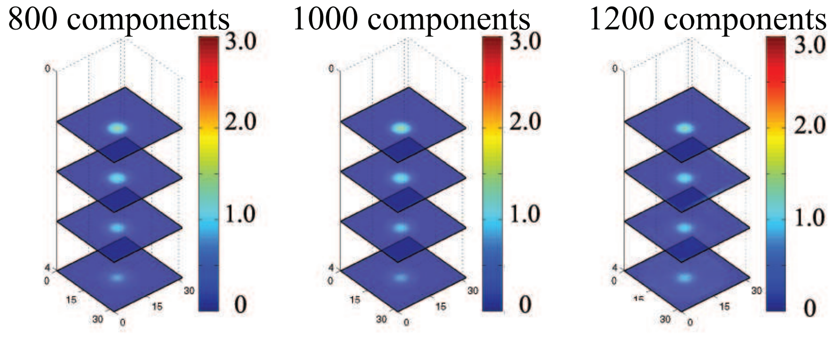


Figure 7.22: The inversions for the third box from the top.

## 7.5 Inversions with Topography

The `GravMagTools` package is capable of taking any given topography into consideration when performing an inversion. In this section we will setup a test problem with a topography. The topography is the Mount Vesuvius (Italy). After having rejected the Fourier based upward continuation for this type of problems when using $T^3$ structure, we are not able to use the observation set consisting of real observations. For this reason we create a forward problem simulating the observation set and placing a source. Figure 7.23 shows the placement of the source inside the topography - we have plotted the source using no interpolation. This means that each of the grid points of the solution domain in which the magnetic gaussian source is placed is marked with a yellow crosses. The solution grid is $25 \times 40 \times 4$ and the observation grid is 4 levels of $28 \times 43$ grid points. The distance between the two grids are 350 meters. The implementation deals with the solution points outside the topography by assigning them a value of 0. The inversion is then performed using CGLS with a tolerance level of $10^{-8}$ the method performs 3958 iterations and in Figure 7.24 the exact solution is illustrated on the left hand-side and the inversion on the right hand-side. The inversion is regarding depth resolution almost identical to the exact solution. The yellow line of the figure represents the mountain side. An interpolated shading is used this is the reason why it appears that not all points outside the topography is set to 0.

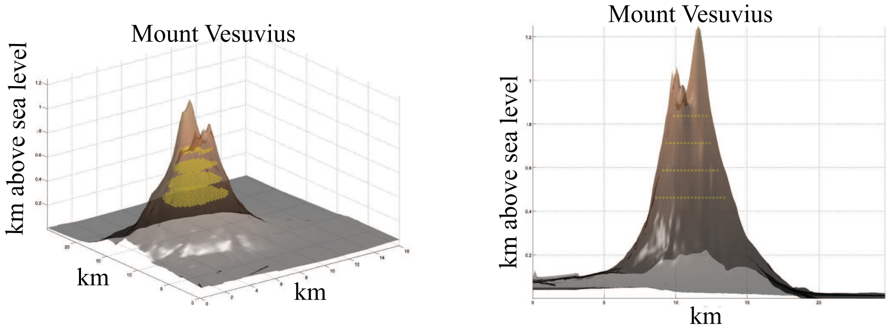


Figure 7.23: The exact solution plotted with topography. The yellow crosses represent the placement of the source.

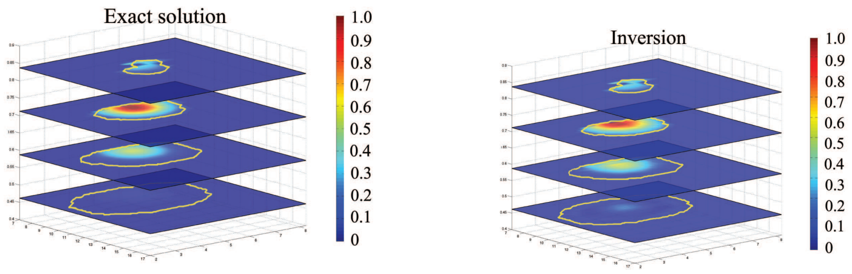


Figure 7.24: The exact solution on the left hand-side and on the right hand-side the inversion. The yellow line indicates where the mountain sides is placed.

## 7.6 Large-scale Problems

This section revolves around the large-scale problems and our ability to perform the inversions. In Section 7.3 we rejected the Fourier based upwards continuation when using the $T^3$ structure, so the observations of these problems are all simulated. We will setup two different large-scale problems one with an uncompressed coefficient matrix of 1.2 GB and one with an uncompressed matrix of approximately 49 GB.

We start with the test problem illustrated in Figure 7.25. The problem has

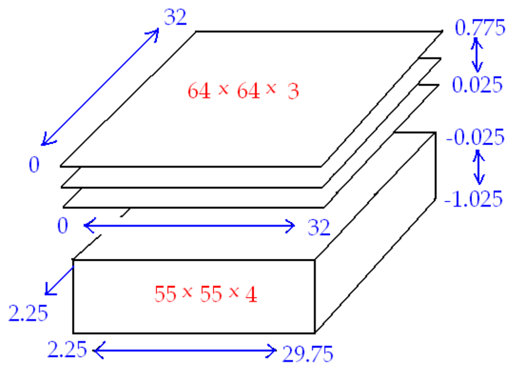


Figure 7.25: Problem setup for full coefficient matrix of 1.2 GB.

a $55 \times 55 \times 4$ solution grid and a $64 \times 64 \times 3$ observation grid. This produces an coefficient matrix of size 1.3 MB that if stored with full structure would take up 1.2 GB virtual memory. On the left hand-side of Figure 7.26 the exact solution of the forward problem is plotted. On the right hand-side the result of the inversion performed using the CGLS regularization method with a tolerance of $10^{-8}$. The result is visually almost identical to the exact solution. The reconstruction was performed in 13 hours and 40 minutes on the Solaris sunfire server at IMM (using Matlab 2006a) and the method stopped when reaching the tolerance level after having performed 30161 iterations. The relative difference between the inversion and the exact solution is 11%.

The second test problem was only solved on our private PC. For this problem we were not especially interested in the quality of the solution compared to the exact solution. The reason for this is that we saw in the first test problem that for this setup it is only a question of a low tolerance level and/or enough iterations when using CGLS. The setup of this test is bigger than the first one, in fact the uncompressed coefficient matrix is approximately 49 GB. Whereas our version of the coefficient matrix takes up only 8.9 MB virtual memory. The PC

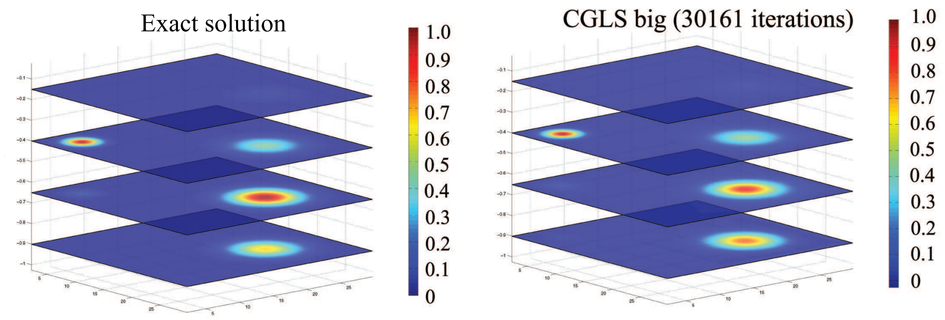


Figure 7.26: On the left hand-side the exact solution of the forward problem. On the right hand-side the CGLS solution with at tolerance of $10^{-8}$.

we use to perform the inversion is a 6 year old Fujitsu Simens with OS Windows XP professional and Matlab 2006a. The processor is an Mobile-Intel Pentium 4 with a CPU of 2.2 GHz, 1.0 GB Ram. The setup contains an solution grid of $125 \times 125 \times 5$ grid points and an observation grid containing $140 \times 140$ grid points in 4 levels - as seen in Figure 7.27. Once again we show the exact solution

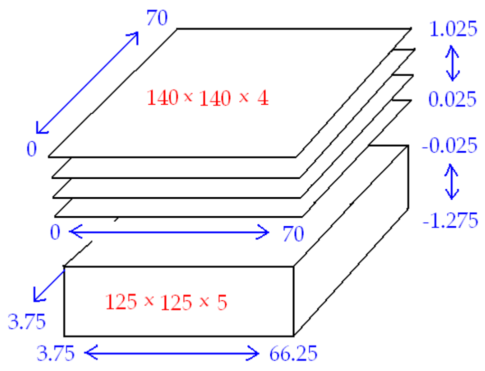


Figure 7.27: Problem setup for full coefficient matrix of 49 GB.

next to the inversion. This is done in Figure 7.28 where the exact solution is shown on the left hand-side. On the right hand-side the result of the CGLS inversion after 12770 iterations. The inversion took 110 hours and 52 minutes and was terminated on iterations before reaching the tolerance level of $10^{-7}$. The relative difference between this large-scale inversion and the exact solution is approximately 30%. Even though the inversion is not identical with the exact solution it is clear to see that the method has placed the magnetization correctly

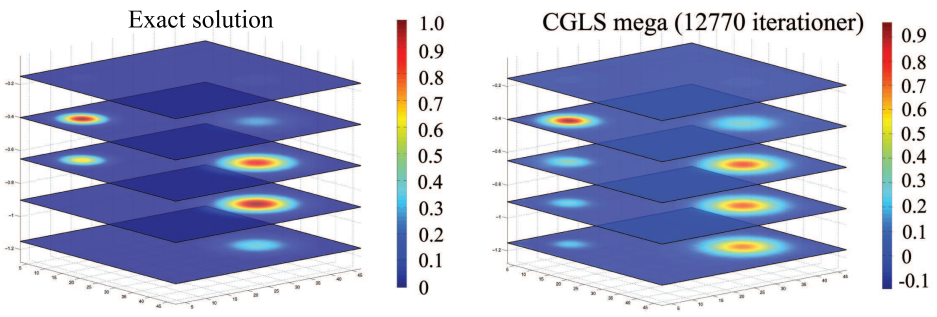


Figure 7.28: On the left hand-side the exact solution of the forward problem. On the right hand-side the CGLS solution with at tolerance of $10^{-7}$.

in the solution domain and we are convinced that with more iterations it would have been an inversion much closer to the exact solution. But as mentioned the main purpose of the test was to prove that the new implementation of the `GravMagTools` package is able to perform an extreme large-scale inversion using a standard PC.

Chapter 8

# Conclusion

In this project we have implemented a new version of the preexisting object-oriented `GravMagTools` package. The new implementation is designed and implemented in such a way that the existing implementation is still fully operational. The main purpose of the new implementation is inversions of large-scale geophysical problems. Our main focus has been on producing and maintaining Toeplitz structures in order to save memory and obtain speed in the computations.

For both the magnetic and the gravity surveying problem we were able to derive a 3-D geometry model in which we achieved the Toeplitz structure that we chose to denote the $T^3$ structure. This allowed us to design a new data structure storing only the necessary elements of the Toeplitz structured coefficient matrix. Thereby we compressed data significantly without loosing information about a single element in the matrix. Having achieved the lossless compression we implemented a uniquely designed FFT matrix-vector multiplication that allowed for multiplications of the new data structure with a vector. In fact this multiplication method performed faster for larger problems than the straightforward multiplication implemented in Matlab.

The geophysical problems are by nature ill-posed which means that when performing the inversion we are vulnerable to errors of any kind. The SVD analysis was used as a tool in able to find a good setup. The resulting setup consisted of using multi-level observation sets, squeezed cells in the $z$ direction, and not placing the observation and solution grid too far apart.

The multi-levels observations sets require some sort of preprocessing of data because geophysicist (usually) provide data sets in one level. To transform the single-level data set to multi-levels the scientists of the University of Naples provided us with the Fourier based upward continuation method. To understand the performance of this method we conducted several tests. These tests illustrated that under some conditions the performance of the method was good, but small changes in the conditions lead to devastating effects. However even under good circumstances we did not achieve depth resolution when inverting with upward continued data. For this reason we conducted an investigation of the behavior of the error introduced by the upward continuation. When using the upward continuation method the error was developed into a signal inseparable from the actual measured signal. Finally we reached the conclusion that the Fourier based upward continuation method was not suited for the type of problems of this thesis when using $T^3$ structure. This decision was not made without regret seeing how it means that we would then be unable to perform inversions using real data sets. For the remainder of the thesis we therefore focused on inversions using simulated data.

In the final part of the thesis the inversions were performed. We were not able to work with real observation data, but we were still interested in showing the potential of the package using simulated data. For this reason we set up a large-scale problem using the topography of Mount Vesuvious. This problem was reconstructed satisfactorily both with regard to the quality of the reconstruction and the incorporation of the topography. The main purpose of this thesis is to set up and reconstruct large-scale problems within a reasonable time consumption. First we set up a problem in which the coefficient matrix took up 1.3 MB, in comparison the uncompressed coefficient matrix took up approximately 1.2 GB of virtual memory. The inversion of this problem was visually very close to the exact solution. To illustrate the point that the package do not require a high performance computer to reconstruct a large-scale problem we set up an even larger problem on a 6 year old standard PC. For this problem the coefficient matrix was 8.9 MB whereas the corresponding full structured matrix would require 49 GB. The setup took 22 seconds and the CGLS solution was reached within 111 hours. This illustrated the fact that even an old PC can solve large-scale problems using the new implementation of the `GravMagTools` package. Furthermore this is done within a reasonable time consumption especially when considering the hardware used. We did not reconstruct bigger problems than this last setup, but we calculated that the theoretically largest coefficient matrix that we could setup and use in calculations corresponded to an uncompressed matrix of $5.2 * 10^5$ GB.

## 8.1 Future work

In order to be able to use the new implementation of the `GravMagTools` package for real inversions a new approach for achieving multi-level data sets is required. This can either be done by measuring data in multi-levels or by designing a physical model of the upward continuation.

In Chapter 7.4 we outlined the work for the implementation of a box solver routine to be able to reconstruct deeper. However the routine needs extensive development before it is operational

The object-oriented approach of the package makes future work easy as objects or modules can be replaces without further modification.

Appendix A

# Deducting of Equations in the Magnetic Surveying Problem

We will in this appendix deduce the equation for the discretized magnetic surveying problem and the constrains which have to be satisfied in order to obtain a $T^3$ structure.

The kernel is given by

$$K(\mathbf{r}, \mathbf{r}') = \frac{\mu_{\mathtt{per}}}{4\pi} \frac{\hat{\mathbf{j}} \cdot (3(\hat{\mathbf{i}} \cdot \hat{\mathbf{d}})\hat{\mathbf{d}} - \hat{\mathbf{i}})}{\|\mathbf{r} - \mathbf{r}'\|} \tag{A.1}$$

$\mathbf{r} = \left[x_i, y_j, z_k\right]^T$ is the location of a dipole magnetic source, $\mathbf{r}' = \left[x'_{i'}, y'_{j'}, z'_{k'}\right]^T$ is a observation point. $\hat{\mathbf{d}}$ is the unit vector with the direction from $\mathbf{r}$ towards $\mathbf{r}'$. $\hat{\mathbf{i}} = \left[i_x, i_y, i_z\right]^T$ is a unit vector with the direction of the magnetic field induced by the Earth. $\hat{\mathbf{j}} = \left[j_x, j_y, j_z\right]^T$ is the unit vector in the direction induced by the dipole in $\mathbf{r}$. $\mu_{\mathtt{per}}$ is the magnetic permeability.

In order to discretize the problem we divide $x$ into $n_x$ points, $y$ is divided into $n_y$ points, $z$ is divided into $n_z$ points, $x'$ is divided into $m_x$ points, $y'$ is divided into $m_y$ points, and $z$ is divided into $m_z$ points. The discritization methods we are using is the midpoint quadrature rule. The matrix of the dicretized problem $\mathbf{A}$ is given by

$$\mathbf{a}_{MN} = w_i w_j w_k K(x_i, y_j, z_k, x'_{i'}, y'_{j'}, z'_{k'})$$

where
$M = i' + (j'-1)m_x + (k'-1)m_x m_y$
$N = i + (j-1)n_x + (k-1)n_x n_y$
$i = 1, 2, ..., n_x$
$j = 1, 2, ..., n_y$
$k = 1, 2, ..., n_z$
$i' = 1, 2, ..., m_x$
$j' = 1, 2, ..., m_y$
$k' = 1, 2, ..., m_z$

The quadrature weights using the midpoint quadrature rule are

$$w_i = \frac{|x_{end} - x_{start}|}{n_x} \text{ and } w_j = \frac{|y_{end} - y_{start}|}{n_y} \text{ and } w_k = \frac{|z_{end} - z_{start}|}{n_z}$$

The quadrature points are

$$\begin{aligned} x_i &= x_{start} + ih_x - \tfrac{h_x}{2} && \text{where } h_x = \tfrac{|x_{end} - x_{start}|}{n_x} \\ y_j &= y_{start} + jh_y - \tfrac{h_y}{2} && \text{where } h_y = \tfrac{|y_{end} - y_{start}|}{n_y} \\ z_k &= z_{start} + kh_z - \tfrac{h_z}{2} && \text{where } h_z = \tfrac{|z_{end} - z_{start}|}{n_z} \end{aligned}$$

The collocation points are

$$\begin{aligned} x'_{i'} &= x'_{start} + i'h_{x'} - \tfrac{h_{x'}}{2} && \text{where } h_{x'} = \tfrac{|x'_{end} - x'_{start}|}{m_x} \\ y'_{j'} &= y'_{start} + j'h_{y'} - \tfrac{h_{y'}}{2} && \text{where } h_{y'} = \tfrac{|y'_{end} - y'_{start}|}{m_y} \\ z'_{k'} &= z'_{start} + k'h_{z'} - \tfrac{h_{z'}}{2} && \text{where } h_{z'} = \tfrac{|z'_{end} - z'_{start}|}{m_z} \end{aligned}$$

To obtain a expression for $K(x_i, y_j, z_k, x'_{i'}, y'_{j'}, z'_{k'})$ we insert the expression for the quadrature points and the collection points into Equation (A.1) and calculate the expression. First we calculate what $\hat{\mathbf{d}}$ becomes when inserting these points.

$$\hat{\mathbf{d}} = \begin{bmatrix} \frac{x_i - x'_{i'}}{\sqrt{(x_i - x'_{i'})^2 + (y_j - y'_{j'})^2 + (z_k - z'_{k'})^2}} \\ \frac{y_j - y'_{j'}}{\sqrt{(x_i - x'_{i'})^2 + (y_j - y'_{j'})^2 + (z_k - z'_{k'})^2}} \\ \frac{z_k - z'_{k'}}{\sqrt{(x_i - x'_{i'})^2 + (y_j - y'_{j'})^2 + (z_k - z'_{k'})^2}} \end{bmatrix}$$

We can now write the expression $\mathbf{a}$ in the following way

$$\mathbf{a}_{MN} = \mu_{\text{per}} w_i w_j w_k \frac{P^{MN}_{ijk}}{Q^{MN}_{ijk}} \tag{A.2}$$

$$
\begin{aligned}
P_{ijk}^{MN} \quad = \quad & \hat{j}_x(3(\hat{i}_x(x'_{start} - x_{start} + \frac{h_x}{2} - \frac{h_{x'}}{2} + (h_{x'}i' - h_x i)) + \hat{i}_y(y'_{start} - y_{start} \\
& +\frac{h_y}{2} - \frac{h_{y'}}{2} + (h_{y'}j' - h_y j)) + \hat{i}_z(z'_{start} - z_{start} + \frac{h_z}{2} - \frac{h_{z'}}{2} + (h_{z'}k' - h_z k))) \\
& (x'_{start} - x_{start} + \frac{h_x}{2} - \frac{h_{x'}}{2} + (h_{x'}i' - h_x i)) - \hat{i}_x((x'_{start} - x_{start} + \frac{h_x}{2} - \frac{h_{x'}}{2} \\
& +(h_{x'}i' - h_x i))^2 + (y'_{start} - y_{start} + \frac{h_y}{2} - \frac{h_{y'}}{2} + (h_{y'}j' - h_y j))^2 + (z'_{start} \\
& -z_{start} + \frac{h_z}{2} - \frac{h_{z'}}{2} + (h_{z'}k' - h_z k))^2)) + \hat{j}_y(3(\hat{i}_x(x'_{start} - x_{start} + \frac{h_x}{2} \\
& -\frac{h_{x'}}{2} + (h_{x'}i' - h_x i)) + \hat{i}_y(y'_{start} - y_{start} + \frac{h_y}{2} - \frac{h_{y'}}{2} + (h_{y'}j' - h_y j)) + \\
& \hat{i}_z(z'_{start} - z_{start} + \frac{h_z}{2} - \frac{h_{z'}}{2} + (h_{z'}k' - h_z k)))(y'_{start} - y_{start} + \frac{h_y}{2} - \frac{h_{y'}}{2} \\
& +(h_{y'}j' - h_y j)) - \hat{y}_x((x'_{start} - x_{start} + \frac{h_x}{2} - \frac{h_{x'}}{2} + (h_{x'}i' - h_x i))^2 + (y'_{start} \\
& -y_{start} + \frac{h_y}{2} - \frac{h_{y'}}{2} + (h_{y'}j' - h_y j))^2 + (z'_{start} - z_{start} + \frac{h_z}{2} - \frac{h_{z'}}{2} + (h_{z'}k' \\
& -h_z k))^2)) + \hat{j}_z(3(\hat{i}_x(x'_{start} - x_{start} + \frac{h_x}{2} - \frac{h_{x'}}{2} + (h_{x'}i' - h_x i)) + \hat{i}_y \\
& (y'_{start} - y_{start} + \frac{h_y}{2} - \frac{h_{y'}}{2} + (h_{y'}j' - h_y j)) + \hat{i}_z(z'_{start} - z_{start} + \frac{h_z}{2} - \\
& \frac{h_{z'}}{2} + (h_{z'}k' - h_z k)))(z'_{start} - z_{start} + \frac{h_z}{2} - \frac{h_{z'}}{2} + (h_{z'}k' - h_z k)) - \\
& \hat{i}_z((x'_{start} - x_{start} + \frac{h_x}{2} - \frac{h_{x'}}{2} + (h_{x'}i' - h_x i))^2 + (y'_{start} - y_{start} + \frac{h_y}{2} \\
& -\frac{h_{y'}}{2} + (h_{y'}j' - h_y j))^2 + (z'_{start} - z_{start} + \frac{h_z}{2} - \frac{h_{z'}}{2} + (h_{z'}k' - h_z k))^2))
\end{aligned}
$$

$$
Q_{ijk} = (\sqrt{(x_i - x_{i'})^2 + (y_j - y_{j'})^2 + (z_k - z_{k'})^2})^5
$$

To obtain a $T^3$ structure the elements in the matrix must only depends on the differences between the indices. By looking at Equation (A.2) is it possible to realize that the following expressions are the only ones containing indices

$$
\begin{gathered}
h_{x'}i' - h_x i \\
h_{y'}j' - h_y j \\
h_{z'}k' - h_z k
\end{gathered}
$$

We are now able to deduce the conditions which have to be met to obtain a $T^3$

structure

$$\begin{aligned} h_{x'}i' - h_x i &= i' - i \\ &\Updownarrow \\ \frac{|x'_{end} - x'_{start}|}{m_x} i' - \frac{|x_{end} - x_{start}|}{n_x} i &= i' - i \\ &\Updownarrow \\ \frac{n_x}{m_x} &= \frac{|x_{end} - x_{start}|}{|x'_{end} - x'_{start}|} \end{aligned}$$

In the same way we calculate the two other condition

$$\begin{aligned} \frac{n_y}{m_y} &= \frac{|y_{end} - y_{start}|}{|y'_{end} - y'_{start}|} \\ \frac{n_z}{m_z} &= \frac{|z_{end} - z_{start}|}{|z'_{end} - z'_{start}|} \end{aligned}$$

These are the same conditions we saw for the gravity surveying problem.

Appendix B

# Description of Cut Border

The linear interpolation method uses a triangle-based linear interpolation and this can cause NaNs[1] at the edges of the resulting dataset. For this reason we design a function that allows for a border to be chosen to be as wide as possible without including any NaNs.

In Figure B.1 an example of a data set containing a NaN (represented by a cross). The blue square in the figure represent the primary data set. This is the data set that will enter into the calculations after having performed an upward continuation. It is important to notice that we do not allow for NaN in the primary data set. The white part of the data set is the natural border that will be used in the upward continuation routine. The red dotted line indicates where we would place the cut in order to preserve the natural border as large as possible. The first thing the function does is to devide the total data set (primary set and the natural border) into different sections. In Figure B.2 we have illustrated the sections. The dark blue color represent the primary data set and the pink and the light blue are different types of the border. We then consider the pink areas that are all treated similarly and therefore we only illustrate one of these areas in Figure B.3. The crosses represent the NaNs. In these areas we measure the distances form each of the NaN to the primary data set. Then the cut is placed (in this case horizontally) at the smallest distance.

The light blue parts of the border in Figure B.2 are treated lastly and any NaN

[1]NaN is the IEEE arithmetic representation for Not-a-Number.

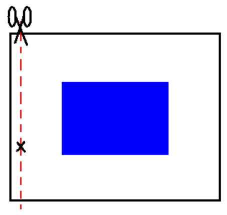


Figure B.1: Example of a data set containing a NaN.

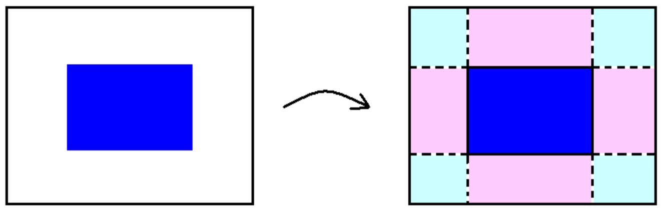


Figure B.2: The division of the total data set. The dark blue area is the primary data set and the pink and light blue are different types of the border.

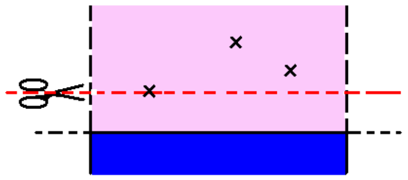


Figure B.3: Cut illustrated for the pink type of border

that has already been eliminated due to the cuts caused by any occurrence of NaN in the pink area are not considered in this step. If there are any NaNs in the light blue areas we consider two different scenarios:

One NaN:
This scenario is illustrated in Figure B.4. From the NaN we measure the distance to the primary observations. We then cut so that the natural border is as large as possible in all directions.

More than one NaN:
This is illustrated in Figure B.5. The function now needs to perform two steps. Firstly the NaN point closest to the nearest corner of the primary set is located and the border is cut so the natural border is preserved as large as possible in both directions. Secondly the smallest rectangle spanned by the corner of the primary data set and the NaN closest to the corner. Then a second cut is performed perpendicular to the first cut.

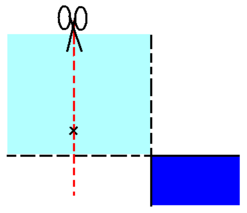


Figure B.4: Cut illustrated for one NaN in the light blue type of border

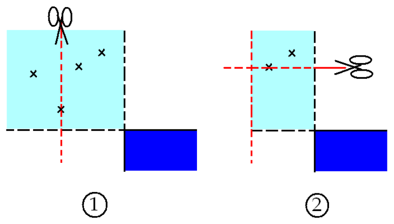


Figure B.5: Cut illustrated for more than one NaN in the light blue type of border

Appendix C

# RegularizerT3

The regularization object used in the new implementation is the same as the implementation using the full structure. To be able to use the object as is, the multiplication routine must be changed in order to make sure that $\mathbf{A} * \mathbf{m}$ and $\mathbf{L} * \mathbf{m}$ holds the same ordering

The basic idea of the new routine is illustrated in Figure C.1. When the multiplication method is used ($\mathbf{L} * \mathbf{m}$) the ordering of the solution object is changed so it corresponds to the ordering of the solution object using the full coefficient matrix. The multiplication is then preformed resulting in a vector that before being returned is changed back to the ordering of the new implementation.

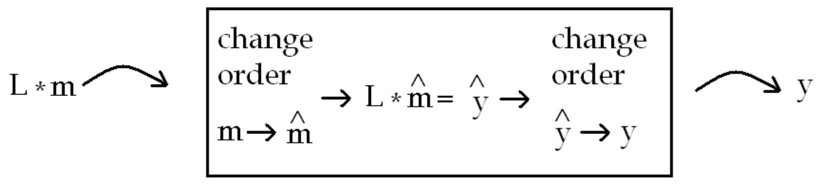


Figure C.1: The multiplication routine for the new regularization routine.

In the following we will derive that this is possible. We have

$$\mathbf{y} = \mathbf{L} * \mathbf{m}$$

we denote the regularizer object of the M$\mathcal{OO}$ReTools ordering $\hat{\mathbf{L}}$. The regularizer object of our implementation is simply a left and right transformation of the M$\mathcal{OO}$ReTools object, thus:

$$\begin{aligned}&\mathbf{L} = \boldsymbol{\pi}^T \hat{\mathbf{L}} \mathbf{P}\\&\mathbf{y} = \boldsymbol{\pi}^T \hat{\mathbf{L}} \mathbf{P} \mathbf{m} = \boldsymbol{\pi}^T \hat{\mathbf{L}} (\mathbf{P} \mathbf{m})\end{aligned}$$

where $\boldsymbol{\pi}^T$ is a left permutation matrix and $\mathbf{P}$ is a right permutation matrix.

In least squares formulation we have

$$\|\mathbf{L}\mathbf{m}\|_2 = \|\boldsymbol{\pi}^T \hat{\mathbf{L}} (\mathbf{P}\mathbf{m})\|_2$$

$\boldsymbol{\pi}$ is an orthogonal matrix ($\boldsymbol{\pi}^T \boldsymbol{\pi} = I$) this leads to

$$\|\boldsymbol{\pi}^T \hat{\mathbf{L}} (\mathbf{P}\mathbf{m})\|_2 = \|\hat{\mathbf{L}} (\mathbf{P}\mathbf{m})\|_2$$

From this we can conclude that we can use the regularizer object from M$\mathcal{OO}$ReTools, we simply have to perform a permutation of the solution object before and after the multiplication is performed.

Appendix D

# Routines of the New Implementation of the GravMagTools package

In this appendix we will list the routines of the new implementation. For further descriptions we have implemented a help topic for each of the routines. These can be activated using the commando

`help function name` (regular functions)
`help private/function name` (private functions)

| **Initial operations** | | |
|---|---|---|
| | Function | Description |
| path | ./ | |
| | startup | Sets up the necessary paths required by the GravMagTools package - regardless of which OS it is run on |

| **Objects** | | |
|---|---|---|
| | Function | Description |
| path | ./@GMT3Operator | |
| | GMT3Operator | Constructor of the GMT3Operator object |
| | GMInfo | Lists information contained in GMT3Operator |
| | GetDimensions | Display dimension of GMT3Operator object |
| | GetLegalPoints | Display legal points of GMT3Operator object |
| | GetWidths | Display widths of GMT3Operator object (dx, dy, dz) |
| | display | Display information on GMT3Operator object |
| | show | Vizualization of the coefficient matrix |
| | sub_applytovector | Supports multiplication of a T3 element with Vector3D |
| | sub_getmatrix | Returns full structure of the coefficient matrix |
| | sub_size | Returns the size of GMT3Operator |
| | subsref | Extracts any field of object GMSolution object |
| path | ./@GMT3Operator/private | |
| | T3_mul | preforms FFT vector-matrix multiplication of T3 matrix and a vector |
| | createA | create the coefficient matrix given the EIG structure |
| | gravityEIG | create the EIG structure for a gravity setup |
| | magneticEIG | create the EIG structure for a magnetic setup |
| | mul_BTTB | preforms FFT vector-matrix multiplication of bttb matrix and a number (mx*my) of vectors of size nx*ny |
| | create_EIGtranspose | create the transposed representation of the coefficient matrix |
| path | ./@GMData | |
| | GMData | Data object (modified - fields added) |
| | show | Visualization of data object (modified) |
| | subsref | Extracts any field of object (modified) |
| | GMInfo | Lists information contained in object (modified) |
| path | ./@GMSolution | |
| | GMSolution | Solution object (modified - fields added) |
| | show | Visualization of data object (modified) |
| | subsref | Extracts any field of object (modified |
| | GMInfo | Lists information contained in object (modified) |
| | SetZerosT3 | Insert zeroes at grid points that lies above topography. |
| path | ./@GMT3Regularizer | |
| | sub_applytovector | Multiplication routine for regularizer object (This object has been copied from GMRegularizer object - only the multiplication routine has been changed) |

| **Setups** | | |
|---|---|---|
| | Function | Description |
| path | ./setupsT3 | |
| | GravSetupT3 | Setup gravity problem (forward/inverse/solution only) |
| | MagSetupT3 | Magnetic gravity problem (forward/inverse/solution only) |
| path | ./setupsT3/private | |
| | CalBorder | Calculates the largest possible border around data without including NaN |
| | ConvertGridT3 | Converts the coordinates in the solution grid, struct which corresponds to the positions of the dipoles. These are centered in the cells. |
| | InterpT3 | Preform the linear interpolation |
| | InterpT3Test | Tests if the data needs to be extrapolated |
| | ReadDirectionsT3 | Reads from data file the assumed directions for the solution in the struct 'solution' |
| | ReadObservationGridT3 | ReadObservationGridT3 reads from data file specifying the observation grid |
| | ReadObservationsT3 | Reads the observations from a file |
| | ReadOptionsT3 | Reads from datafile in which the domain size and type of upward continuation is stated |
| | ReadSolutionGridT3 | Reads the solutiongrid from a datafile. |
| | ReadTopographyT3 | Reads topography data from file. Corrects the coordinates of observations since these are defined to be the height above topography. It also locates solution points above the topography. |
| | ReshapeT3 | Reshapes a vector of T3 form to a matrix (of type GMMatrix ordering) |
| | T3StructurTest | Test if the solution- and observation grid in the x and y direction produces T3 structure. |
| | bordowav | Places a border around the observation level |
| | cont2D | Performs the upward continuation |

**Tools**

| | Function | Description |
|---|---|---|
| path | ./Tools | |
| | GenGridT3 | Generates a file with T3 grid |
| | GenSolution | Generates a file with all the data needed to describe a solution |
| | WriteDataToObsFile | Generates the observation file in the right format from a matrix with observations/data |
| | GM_2_GMT3 | Permutate a vector or matrix of GMMatrix ordering to $T^3$ ordering |
| | GMT3_2_GM | Permutate a vector or matrix of $T^3$ ordering to GMMatrix ordering |

**Visualization routines**

| | Function | Description |
|---|---|---|
| path | ./VisualizeT3 | |
| | PlotLegalPointsT3 | Visualization of legal grid points an SVD and produces a Picard plot |
| | sliceplotT3 | Visualization of solution vector using sliceplot |
| | draw_line | Visualize the edge of the topography. Used by sliceplotT3 |

# Bibliography

[1] Lars Eldén, Linde Wittmeyer-Koch, Hans Bruun Nielsen, **Introduction to Numerical Computation - analysis and MATLAB illustrations**, Studentlitteratur, Lund, 2004, pp 375

[2] M. S. Zhdanov, **Geophysical Inverse Theory and Regularization Problems**, Elsevier 2002, Salt Lake City, 2002, pp 609

[3] Maurizio Fedi, Per Christian Hansen, and Valeria Paoletti, **Analysis of depth resolution in potential-field inversion**, Geophysics, vol. 70, A1-A11, 2005

[4] Michael Jacobsen, **Modular Regularization Algorithms**, IMM-PhD-2004-140, IMM, 2004, pp 205. Software can be acquired from http://www2.imm.dtu.dk/∼tkj/MOOReTools/index.shtml

[5] Per Christian Hansen, Jesper Pedersen, Maurizio Fedi, and Valeria Paoletti **GravMag Tool: an object oriented Matlab package for 3-D potential-field inversion**, IMM, Work in progress

[6] Per Christian Hansen, **Regularization Tools: A Matlab Package for Analysis and Solution of Discrete Ill-Posed Problems**, Numerical Algorithms 6: pp 1-35, 1994. Software can be aquired from http://www2.imm.dtu.dk/∼pch/Regutools/index.html

[7] Per Christian Hansen, **Deconvolution and regularization with Toeplitz matrices**, Numerical Algorithms 29: 323-378, 2002

[8] Per Christian Hansen, **Discrete Inverse Problems, Insight and Algorithms**, manuscript in progress, IMM, November 2005

[9] Richard J. Blakely **Potential Theory in Gravity & Magnetic Applications**, Cambridge University Press, Cambridge, 1996, 441 pp.

[10] Søren Nymand Lophaven, Hans Bruun Nielsen, Jacob Søndergaard, **DACE - A Matlab Kriging Toolbox, Version 2.0**, IMM-REP-2002-12, IMM, 2002. Software can be acquired from http://www2.imm.dtu.dk/∼hbn/dace/.

[11] Toke Koldborg Jensen, **Stabilization Algorithms for Large-Scale Problems**, IMM-PhD-2006-163, IMM, 2006, pp 243

[12] William C. Dean, **Frequency Analysis for Gravity and Magnetic Interpretation**, Geophysics, vol XXII, pp 97-127, 1958)