Observatório Nacional

Divisão de Programas
de Pós-Graduação

CAMADA EQUIVALENTE CONVOLUCIONAL PARA PROCESSAMENTO DE DADOS POTENCIAIS

Diego Takahashi Tomazella

Tese apresentada ao Programa de Pós-graduação em Geofísica do Observatório Nacional, como parte dos requisitos necessários à obtenção do título de Doutor em Geofísica.

Orientador(a):	Dr. Vanderlei Coelho Oliveira Jr.
Co-orientador(a):	Dra. Valéria C. F. Barbosa

Rio de Janeiro
Setembro de 2021

Observatório Nacional

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÕES

PÁTRIA AMADA BRASIL GOVERNO FEDERAL

DIVISÃO DE PROGRAMAS DE PÓS-GRADUAÇÃO

Ata de Defesa

Doutorado em Geofísica. Processo 769/2017. Candidato **Diego Takahashi Tomazella.** No vigésimo sétimo dia do mês de setembro do ano de dois mil e vinte e um, às 14h10, na modalidade remota. Integraram a Banca Examinadora os Doutores Vanderlei Coelho de Oliveira Junior – ON, orientador do candidato e presidente da banca; Valeria Cristina Ferreira Barbosa - ON, coorientadora do candidato; Cosme Ferreira da Ponte Neto - ON, Joerg Dietrich Wilhelm Schleicher -UNICAMP, Yara Regina Marangoni - IAG/USP, Cícero Roberto Teixeira Regis -UFPA, os quais foram indicados pela Comissão de Pós-Graduação em Geofísica. O presidente iniciou a sessão pública, tendo o candidato explanado a respeito do seu trabalho intitulado: **"CAMADA EQUIVALENTE CONVOLUCIONAL PARA PROCESSAMENTO DE DADOS POTENCIAIS"**, sobre a qual foi sucessivamente arguido pelos membros da banca. Após a arguição o presidente suspendeu a sessão pública para que em sessão secreta a banca expressasse o seu julgamento sobre o trabalho, o qual foi considerado **APROVADO**. O Presidente reabriu os trabalhos da sessão pública anunciando a decisão da Banca Examinadora. Eu, Giane do Carmo Boldrim, lavrei o presente termo que assino juntamente com os membros da Banca e o candidato.

Rio de Janeiro, 27 de setembro de 2021.

a) [illegible]
Vanderlei Coelho de Oliveira Junior

b) [illegible]
Valeria Cristina Ferreira Barbosa

c) [illegible]
Cosme Ferreira da Ponte Neto

d) [illegible]
Joerg Dietrich Wilhelm Schleicher

e) [illegible]
Yara Regina Marangoni

f) [illegible]
Cícero Roberto Teixeira Regis

g) Diego Takahashi Tomazella
Diego Takahashi Tomazella

h) [illegible]
Giane do Carmo Boldrim

Resumo da Tese apresentada ao Programa de Pós-Graduação em Geofísica do Observatório Nacional como parte dos requisitos necessários para a obtenção do título de Doutor em Geofísica.

CAMADA EQUIVALENTE CONVOLUCIONAL PARA PROCESSAMENTO DE DADOS POTENCIAIS

Diego Takahashi Tomazella

Setembro/2021

Neste trabalho foi desenvolvida uma eficiente e rápida técnica de camada equivalente para o processamento de dados de campos potenciais usando um método de convolução discreta que modifica o cálculo do problema direto dos métodos iterativos baseado em um vínculo de excesso de massa para o caso gravimétrico e o algoritmo do gradiente conjugado por mínimos quadrados para o caso magnético. Aproveitando as estruturas *block-Toeplitz Toeplitz-block* (BTTB) da matriz de sensibilidade, que surge quando *grids* de observações e de fontes equivalentes (pontos de massa ou dipolos) são regulares, desenvolvemos um algoritmo que reduz drasticamente o número de cálculos de pontos flutuantes (*flops*) e de memória RAM necessária para estimar a distribuição de propriedade física 2D sobre a camada equivalente. A estrutura da matriz BTTB pode ser escrita usando somente a primeira coluna da matriz de sensibilidade, que pode ser transformada em uma matriz *block-circulant circulant-block* (BCCB). Similarmente, somente a primeira coluna da matriz BCCB é necessária para reconstrui-la. Usando a primeira coluna da BCCB também é possível calcular seus auto-valores por uma transformada de Fourier 2D (2D FFT), que pode ser usada para calcular rapidamente o problema direto da camada equivalente. Como resultado, este método pode ser usado para processar grandes conjuntos de dados de forma eficiente. Testes com dados sintéticos mostram que o método estima as fontes equivalentes de forma satisfatória para técnicas de processamento, como por exemplo, a continuação para cima de dados gravimétricos e magnéticos. Os resultados mostram efeitos de borda e de ruído muito reduzidos comparados ao método tradicional no domínio de Fourier. Para o caso gravimétrico, os testes sintéticos mostram que para processar $N = 1\,000\,000$ de observações, este método precisou de $\approx 30,9$ segundos, enquanto que o método iterativo com vínculo de massa levou $\approx 46,8$ segundos com apenas $N = 22\,500$. Um teste com o dado real da Província de

Carajás, Brasil, mostra o baixo custo computacional deste método para processar grandes volumes de dados, usando $N = 250\,000$ observações. Testes sintéticos com dados magnéticos mostram uma diminuição da ordem de $\approx 10^4$ em *flops* e ≈ 25 vezes em tempo computacional com um *grid* de tamanho médio de 100×50 se comparado o método clássico da solução de sistemas lineares das equações normais por mínimos quadrados usando o método da decomposição de Cholesky. Resultados ainda melhores são obtidos usando milhões de dados, mostrando um decréscimo exponencial no uso de memória RAM e de custo computacional, permitindo o uso deste método em computadores pessoais. Os resultados mostram, comparado ao método de Fourier, que o processamento magnético requer tempo computacional similar, mas produz menores efeitos de borda sem usar nenhum tipo de *padding* e também se mostrando muito mais robusta para lidar com dados irregulares ou superfícies onduladas. Um teste com $N = 1\,310\,000$ dados irregularmente espaçados da Província de Carajás, Brasil, confirma com sucesso este método levando $\approx 385,56$ segundos para estimar a distribuição de propriedade física e $\approx 2,64$ segundos para calcular a continuação para cima.

Abstract of the Thesis presented to the National Observatory's Graduate Program in Geophysics as a partial fulfillment of the requirements for the degree of Doctor in Geophysics.

CONVOLUTIONAL EQUIVALENT LAYER FOR POTENTIAL DATA PROCESSING

Diego Takahashi Tomazella

September/2021

We have developed an efficient and very fast equivalent-layer technique for gravity and magnetic data processing by modifying the forward problem calculation of an iterative method grounded on excess mass constraint that does not require the solution of linear systems and of the conjugate gradient least squares algorithm, respectively, using a discrete convolutional method. Taking advantage of the Block-Toeplitz Toeplitz-block (BTTB) structure of the sensitivity matrix, that raises when regular grids of observation points and equivalent sources (point masses or dipoles) are used to set up a fictitious equivalent layer, we have developed an algorithm which greatly reduces the number of floating-point operations (flops) and computer memory necessary to estimate a 2D physical property distribution over the equivalent layer. The structure of the BTTB matrix can be written by using only the elements of the first column of the sensitivity matrix, which in turn can be transformed into a block-circulant circulant-block (BCCB) matrix. Likewise, only the first column of the BCCB matrix is needed to reconstruct the full sensitivity matrix completely. Also, from the first column of BCCB matrix, its eigenvalues can be calculated using the 2D Fast Fourier Transform (2D FFT), which can be used to readily compute the matrix-vector product of the forward modeling in the fast equivalent-layer technique. As a result, our method is efficient to process very large datasets. Tests with synthetic data demonstrate the ability of our method to satisfactorily use the estimated equivalent sources for data processing, for example, upward-continuing the gravity and magnetic data. Our results show very small border effects and noise amplification compared to those produced by the classical approach in the Fourier domain. For the gravity case, our synthetic results show that while the running time of our method is ≈ 30.9 seconds for processing $N = 1,000,000$ observations, the iterative method grounded on excess mass constrain spent ≈ 46.8 seconds with $N = 22,500$.

A test with field data from Carajás Province, Brazil, illustrates the low computational cost of our method to process a large data set composed of $N = 250,000$ observations. Synthetic tests for magnetic data with a mid-size 100×50 grid of total-field anomaly data show a decrease of $\approx 10^4$ in floating-point operations and $\approx 25\times$ in computation runtime of our method compared to the classical approach of solving the least-squares normal equations via Cholesky decomposition. Faster results are obtained for millions of data, showing drastic decreases in computer memory usage and runtime, allowing to perform magnetic data processing of large data sets on regular desktop computers. Our results also show that, compared to the classical Fourier approach, the magnetic data processing with our method requires similar computation time, but produces significantly smaller border effects without using any padding scheme and also is more robust to deal with data on irregularly spaced points or on undulating observation surfaces. A test with $1,310,000$ irregularly spaced field data over the Carajás Province, Brazil, confirms the efficiency of our method by taking ≈ 385.56 seconds to estimate the physical-property distribution over the equivalent layer and ≈ 2.64 seconds to compute the upward-continuation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The equivalent layer is a well-known technique for processing potential-field data in applied geophysics since the 60's (BOTT, 1967; DAMPNEY, 1969; DANES, 1961). It comes from potential theory as a mathematical solution of the Laplace's equation, in the region above the sources, by using the Dirichlet boundary condition (KELLOGG, 1929). This theory states that any potential-field data produced by an arbitrary 3D physical-property distribution can be exactly reproduced by a fictitious layer located at any depth and having a continuous 2D physical-property distribution. In practical situations, the layer is approximated by a finite set of sources (e.g., point masses or dipoles) and their physical properties are estimated by solving a linear system of equations that yields an acceptable potential-field data fit. These fictitious sources are called equivalent sources.

Many previous works have used the equivalent layer to perform different potential-field data transformations such as gridding (e.g., CORDELL, 1992; DAMPNEY, 1969; MENDONÇA and SILVA, 1994), upward- and downward-continuation (e.g., EMILIA, 1973; HANSEN and MIYAZAKI, 1984; LI and OLDENBURG, 2010), reduction to the pole (e.g., GUSPÍ and NOVARA, 2009; LEÃO and SILVA, 1989; LI *et al.*, 2014; OLIVEIRA JR. *et al.*, 2013; SILVA, 1986), combining multiple data sets (e.g., BOGGS and DRANSFIELD, 2004), gradient data processing (e.g., BARNES and LUMLEY, 2011) first and second vertical derivatives fields (EMILIA, 1973) and total magnetic induction vector components calculation (SUN *et al.*, 2019).

Although the use of the equivalent-layer technique increased over the last decades, one of the biggest problems is still its high computational cost for processing large-data sets. This problem propelled several studies to improve the computational efficiency of the equivalent layer technique. LEÃO and SILVA (1989) developed a fast method for processing a regular grid of potential-field data. The method consists in estimating an equivalent layer which exactly reproduces the potential-field data within a small data window. The data window is shifted over the whole gridded data in a procedure similar to a discrete convolution. The equivalent layer extends beyond the moving-data window and is located at a depth between two and six times the grid spacing of the observations. For each data window, the equivalent layer is estimated by solving an underdetermined linear system. After estimating an equivalent layer, the transformed-potential field is computed only at the center of the moving-data window. The use of a small moving-data window greatly reduces the total number of floating-point operations (*flops*) and computer memory storage. The computational efficiency of this method relies on the strategy of constructing the equivalent layer by successively solving small linear systems instead of solving just one large linear system for the entire equivalent layer. MENDONÇA and SILVA (1994) also followed the strategy of solving successive small linear systems for constructing an equivalent layer. Their method is based on the equivalent-data concept, which consists in determining a subset of all potential-field data (named equivalent-data set), such that the interpolating surface that fits the chosen subset also automatically fits all remaining data. The equivalent-data set is obtained by iteratively introducing the potential-field observation with the greatest residual in the preceding iteration. By applying to the interpolation problem, the method is optimized by approximating dot products by the discrete form of an analytic integration that can be evaluated with less computational effort. According to the authors, the equivalent-data set is usually smaller than the total number of potential-field observations, leading to computational savings. The authors also pointed out that the computational efficiency of the method depends on the number of equivalent data. If the potential-field anomaly is nonsmooth, the number of equivalent data can be large and the method will be less efficient than the classical approach.

By following a different strategy, LI and OLDENBURG (2010) developed a rapid method that transforms the dense sensitivity matrix associated with the linear system into a sparse one by using a wavelet technique. After obtaining a sparse representation of the sensitivity matrix, those authors estimate the physical-property distribution within the equivalent layer by using an overdetermined formulation. Those authors pointed out that, given the sparse representation, their method reduces the computational time required for solving the linear system by as many as two orders of magnitude if compared with the same formulation using a dense matrix. BARNES and LUMLEY (2011) followed a similar strategy and transformed the dense sensitivity matrix into a sparse one. However, differently from LI and OLDENBURG (2010), their method operates in the space domain by grouping equivalent sources far from an observation point into blocks with average physical property. This procedure aims at reducing the memory storage and achieving computational efficiency by solving the transformed linear system with a weighted-least-squares conjugate-gradient algorithm. Notice that, instead of constructing the equivalent layer by solving successive small linear systems, these last two methods first transform the large linear system into a sparse one and then take advantage of this sparseness.

OLIVEIRA JR. *et al.* (2013) developed a fast method based on the reparameterization of the physical-property distribution within the equivalent layer. Those authors divided the equivalent layer into a regular grid of equivalent-source windows inside which the physical-property distribution is described by bivariate polynomial functions. By using this polynomial representation, the inverse problem for estimating the equivalent layer is posed in the space of the total number of polynomial coefficients within all equivalent-source windows instead of in the space of the total number of equivalent sources. According to OLIVEIRA JR. *et al.* (2013), the computational efficiency of their method relies on the fact that the total number of polynomial coefficients needed to describe the physical-property distribution within the equivalent layer is generally much smaller than the number of equivalent sources, leading to a very smaller linear system to be solved. Those authors could verify that the total number of *flops* needed for building and solving the linear inverse problem of estimating the total number of polynomial coefficients can be reduced by as many as three and four orders of magnitude, respectively, if compared with the same inverse problem of estimating the physical property of each equivalent source via Cholesky decomposition.

There is another class of methods that iteratively estimates the physical-property distribution within the equivalent layer without solving linear systems. The method presented by CORDELL (1992), and later generalized by GUSPÍ and NOVARA (2009), updates the physical property of the sources, which are located below each potential-field data, using a procedure that removes the maximum residual between the observed and predicted data. XIA and SPROWL (1991) and XIA *et al.* (1993) developed fast iterative schemes for updating the physical-property distribution within the equivalent layer in the wavenumber and space domains, respectively. Grounded on excess mass constraint, SIQUEIRA *et al.* (2017) developed an iterative scheme starting with a mass distribution within the equivalent layer that is proportional to observed gravity data. Then, their method iteratively adds mass corrections that are proportional to the gravity residuals. The total number of *flops* required by these iterative methods for estimating the physical-property distribution within the equivalent layer depends on the total number of iterations, however this number is generally much smaller than the total number of *flops* required to solve a large-scaled linear system. Generally, the most computational expensive step in each iteration of these methods is the forward problem of calculating the potential-field data produced by the equivalent layer.

In the present work, we show that the sensitivity matrix associated with a planar equivalent layer of point masses/dipoles has a very well-defined structure called

Block-Toeplitz Toeplitz-Block (BTTB) for the case in which (i) the observed gravity or magnetic data is located on a regularly spaced grid at constant height and (ii) there is one equivalent source directly beneath each observation point. This technique have been successfully used in potential-field methods for 3D gravity inversion (ZHANG and WONG, 2015), downward-continuation (ZHANG *et al.*, 2016) and 3D magnetic modeling (QIANG *et al.*, 2019). More recently in HOGUE *et al.* (2020) the authors provided an overview on modeling the gravity and magnetic kernels using the BTTB structures and RENAUT *et al.* (2020) used for inversion of both gravity and magnetic data to recover sparse subsurface structures. By using this property, we propose an efficient algorithm based on FFT convolution (e.g., VAN LOAN, 1992, p. 207) for computing the forward problem at each iteration of the fast equivalent-layer technique proposed by SIQUEIRA *et al.* (2017) for the gravity case and also for computing the forward problem at each iteration of the conjugate gradient least squares algorithm for the magntic case.

In Part I, our method uses the gravitational effect produced by a single point mass to compute the effect produced by the whole equivalent layer, which results in a drastic reduction not only in the number of flops, but also in the RAM memory usage of the fast equivalent-layer technique. Tests with synthetic and field data illustrate the good performance of our method in processing large gravity data sets.

In Part II, we achieve very fast solutions using a conjugate gradient algorithm combined with the fast Fourier transform. We present a novel method of exploring the symmetric structures of the second order derivatives of the inverse of the distance contained in the magnetic kernel, to keep the memory RAM usage to the minimal by using only one equivalent source to carry the calculations of the forward problem. We also show tests of the magnetic convolutional equivalent layer when irregular grids are used. The convergence of the conjugate gradient maintains an acceptable level even using irregular grids. Our results show a good performance of our method in producing fast and robust solutions for processing large amounts of magnetic data using the equivalent layer technique.

Chapter 2

Fundamentals

2.1 Classical equivalent layer

2.1.1 Classical equivalent layer for gravity data processing

Let d_i^o be the observed gravity data at the point (x_i, y_i, z_i), $i = 1, ..., N$, of a local Cartesian system with x-axis pointing to north, the y-axis pointing to east and the z-axis pointing downward from the $N \times 1$ data vectopr $\mathbf{d}^o$. Let us consider an equivalent layer composed by a set of N point masses (equivalent sources) over a layer located at depth z_c ($z_c > z_i$) and whose x- and y- coordinates of each point mass coincides with the corresponding coordinates of the observation directly above. There is a linear relationship that maps the unknown mass distribution onto the gravity data given by

$$\mathbf{d}(\mathbf{p}_{\boldsymbol{g}}) = \mathbf{A}_{\boldsymbol{g}}\mathbf{p}_{\boldsymbol{g}} \ , \tag{2.1}$$

where $\mathbf{d}(\mathbf{p}_{\boldsymbol{g}})$ is an $N \times 1$ vector whose ith element is the predicted gravity data at the ith point (x_i,y_i,z_i), $\mathbf{p}$ is the unknown $N \times 1$ parameter vector whose jth element p_j is the mass of the jth equivalent source (point mass) at the jth Cartesian coordinates (x_j,y_j,z_c) and $\mathbf{A}_{\boldsymbol{g}}$ is an $N \times N$ sensitivity matrix whose ijth element is given by

$$a_{ij}^g = \frac{c_g \, G \, (z_c - z_i)}{[(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_c)^2]^{\frac{3}{2}}} \ , \tag{2.2}$$

where G is the Newton's gravitational constant and $c_g = 10^5$ transforms from m/s^2 to mGal. Notice that equation 2.2 is the product of these constants with the vertical derivative of the function inverse of the distance

$$\frac{1}{r_{ij}} = \frac{1}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_c)^2}} \tag{2.3}$$

Also, notice that the sensitivity matrix depends on the ith coordinate of the observation and the jth coordinate of the equivalent source. For convenience, we designate these coordinates as *matrix coordinates* and the indices i and j as *matrix indices.*

To estimate the unknown mass distibution $\mathbf{p_g}$ the Cholesky factorization method will introduced in section 2.1.3.

2.1.2 Classical equivalent layer for magnetic data processing

Let $\mathbf{d}^o$ be a $N \times 1$ observed data vector where, $d_i^o\ (x_i, y_i, z_i)$, $i = 1, \dots, N$, is the total-field anomaly produced by arbitrarily magnetized sources at the ith position, aranged in a right-handed Cartesian coordinate system with x-, y- and z-axis pointing to north, east and down, respectively. We consider that the total-field anomaly data d_i^o represent the discrete values of a harmonic function. Besides, we consider that the main geomagnetic field direction at the study area can be defined by the unit vector

$$\hat{\mathbf{F}} = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} \cos(I_0)\,\cos(D_0) \\ \cos(I_0)\,\sin(D_0) \\ \sin(I_0) \end{bmatrix} , \tag{2.4}$$

with constant inclination I_0 and declination D_0. In this case, d_i^o can be approximated by the predicted total-field anomaly (BLAKELY, 1996)

$$\Delta T_i = \sum_{j=1}^{M} p_j a_{ij}^m , \tag{2.5}$$

which describes the magnetic induction exerted, at the observation point (x_i, y_i, z_i), by a discrete layer of M dipoles (equivalent sources) defined on the horizontal plane $z = z_c$, where p_j is the magnetic moment intensity (in Am^2) of the jth dipole, that has unit volume and is located at the point (x_j, y_j, z_c). In equation 2.5, a_{ij} is the harmonic function

$$a_{ij}^m = c_m \frac{\mu_0}{4\pi} \hat{\mathbf{F}}^\top \mathbf{H}_{ij}\, \hat{\mathbf{u}} , \tag{2.6}$$

the unit vector

$$\hat{\mathbf{u}} = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} = \begin{bmatrix} \cos(I)\,\cos(D) \\ \cos(I)\,\sin(D) \\ \sin(I) \end{bmatrix} , \tag{2.7}$$

defines the magnetization direction of all dipoles, with constant inclination I and declination D, $\mu_0 = 4\pi\, 10^{-7}$ H/m is the magnetic constant, $c_m = 10^9$ is a factor that transforms the magnetic induction from Tesla (T) to nanotesla (nT) and $\mathbf{H}_{ij}$

is a 3×3 matrix

$$\mathbf{H}_{ij} = \begin{bmatrix} h_{ij}^{xx} & h_{ij}^{xy} & h_{ij}^{xz} \\ h_{ij}^{xy} & h_{ij}^{yy} & h_{ij}^{yz} \\ h_{ij}^{xz} & h_{ij}^{yz} & h_{ij}^{zz} \end{bmatrix} , \tag{2.8}$$

with elements defined in terms of matrix coordinates and matrix indices according to

$$h_{ij}^{\alpha\beta} = \begin{cases} \frac{3(\alpha_i - \alpha_j)^2}{r_{ij}^5} - \frac{1}{r_{ij}^3} , & \alpha = \beta \\ \frac{3(\alpha_i - \alpha_j)(\beta_i - \beta_j)}{r_{ij}^5} , & \alpha \neq \beta \end{cases} , \quad \alpha, \beta = x, y, z , \tag{2.9}$$

which are the second derivatives of the inverse distance function (equation 2.3) with respect to the coordinates of the observation point (x_i, y_i, z_i).

Equation 2.5 can be rewritten in matrix notation as follows:

$$\mathbf{d}(\mathbf{p}_{\boldsymbol{m}}) = \mathbf{A}_{\boldsymbol{m}} \mathbf{p}_{\boldsymbol{m}} , \tag{2.10}$$

where $\mathbf{d}(\mathbf{p}_{\boldsymbol{m}})$ is the $N \times 1$ predicted data vector with ith element defined as the predicted total-field anomaly ΔT_i (equation 2.5), $\mathbf{p_m}$ is the $M \times 1$ parameter vector whose jth element is the magnetic moment intensity p_j of the jth dipole and $\mathbf{A}_{\boldsymbol{m}}$ is the $N \times M$ sensitivity matrix with element ij defined by the harmonic function a_{ij} (equation 2.6).

2.1.3 Cholesky factorization

In the classical equivalent-layer technique, the common approach for estimating the parameter vector from the observed gravity or the total-field anomaly data $\mathbf{d}^o$ is by solving the least-squares normal equations

$$\tilde{\mathbf{p}} = \left(\mathbf{A}^\top \mathbf{A}\right)^{-1} \mathbf{A}^\top \mathbf{d}^o . \tag{2.11}$$

This equation can be rewritten to

$$\mathbf{A}^\top \mathbf{A} \, \mathbf{p} = \mathbf{A}^\top \mathbf{d}^o . \tag{2.12}$$

Equation 2.12 is usually solved by first computing the Cholesky factor $\mathbf{G}$ of matrix $\mathbf{A}^\top \mathbf{A}$ and then using it to solve the linear systems (GOLUB and LOAN, 2013, p. 262):

$$\begin{aligned} \mathbf{G}\mathbf{w} &= \mathbf{A}^\top \mathbf{d}^o \\ \mathbf{G}^\top \tilde{\mathbf{p}} &= \mathbf{w} \end{aligned} , \tag{2.13}$$

where $\mathbf{w}$ is a temporary variable. This approach to estimating the parameter vector will be referenced throughout this work as the *classical method.* The computational cost associated with the classical method can be very high when dealing with large datasets. In the following subsections, we will show how to explore the structures of the sensitivity matrices $\mathbf{A}_{\boldsymbol{g}}$ and $\mathbf{A}_{\boldsymbol{m}}$ to efficiently solve the least-squares normal equations (equation 2.12).

2.2 Fast equivalent-layer methods

2.2.1 Fast equivalent-layer technique for gravity data processing

SIQUEIRA *et al.* (2017) developed an iterative least-squares method to estimate the mass distribution over the equivalent layer based on the excess of mass and the positive correlation between the observed gravity data and the masses on the equivalent layer. They showed that the fast equivalent-layer technique has a better computational efficiency than the classical equivalent layer approach (equation 2.11) if the dataset is greater than at least 200 observation points, even using a large number of iterations.

Considering one equivalent source (point mass) directly beneath each observation point, the iteration of the SIQUEIRA *et al.*'s (2017) method starts by an initial approximation of mass distribution given by

$$\tilde{\mathbf{p}}^0 = \tilde{\mathbf{A}}_{\boldsymbol{g}}^{-1}\mathbf{d}^o \, , \tag{2.14}$$

where $\tilde{\mathbf{A}}_{\boldsymbol{g}}^{-1}$ is an $N \times N$ diagonal matrix with elements

$$\tilde{a}_{ii}^{-1} = \frac{\Delta s_i}{(2\pi \, G \, c_g)} \, , \tag{2.15}$$

where Δs_i is the ith element of surface area located at the ith horizontal coordinates x_i and y_i of the ith observation. At the kth iteration, the masses of the equivalent sources are updated by

$$\tilde{\mathbf{p}}^{k+1} = \tilde{\mathbf{p}}^k + \boldsymbol{\Delta}\tilde{\mathbf{p}}^k \, , \tag{2.16}$$

where the mass correction is given by

$$\boldsymbol{\Delta}\tilde{\mathbf{p}}^{k+1} = \tilde{\mathbf{A}}_{\boldsymbol{g}}^{-1}(\mathbf{d}^o - \mathbf{A}_{\boldsymbol{g}}\tilde{\mathbf{p}}^k) \, . \tag{2.17}$$

At the kth iteration of SIQUEIRA *et al.*'s (2017) method, the matrix-vector product $\mathbf{A}_{\boldsymbol{g}}\tilde{\mathbf{p}}^k = \mathbf{d}(\tilde{\mathbf{p}}^k)$ must be calculated to get a new residual $\mathbf{d^o} - \mathbf{A}_{\boldsymbol{g}}\tilde{\mathbf{p}}^k$, which represents a bottleneck. Considering the limitation of 16 Gb of computer memory in our system, we could run the SIQUEIRA *et al.*'s (2017) method only up to $22,500$ observation points; Otherwise, it is costly and can be prohibitive in terms of computer memory to maintain such operation.

2.2.2 Conjugate Gradient Least Squares (CGLS) method for magnetic data processing

The computational cost associated with the classical method to estimate the parameter vector $\tilde{\mathbf{p}}$ by solving the linear system 2.12 can be very high or even prohibitive when dealing with large data sets. In these cases, a well-known alternative is solving the normal equations (equation 2.12) iteratively by using the *standard Conjugate Gradient Least Squares (CGLS) method*:

Algorithm 1 Standard CGLS pseudocode (ASTER *et al.*, 2019, p. 166).

```
Input: $\mathbf{A_m}$ and $\mathbf{d}^o$.
Output: Estimated parameter vector $\tilde{\mathbf{p}}$.
Set $it = 0$, $\tilde{\mathbf{p}}_{(it)} = \mathbf{0}$, $\mathbf{c}_{(it-1)} = \mathbf{0}$, $\beta_{(it)} = 0$, $\mathbf{s}_{(it)} = \mathbf{d}^o$ and $\mathbf{r}_{(it)} = \mathbf{A_m}^\top \mathbf{s}_{(it)}$.
1 - If $it > 0$, $\beta_{(it)} = \dfrac{\|\mathbf{r}_{(it)}\|_2^2}{\|\mathbf{r}_{(it-1)}\|_2^2}$
2 - $\mathbf{c}_{(it)} = \mathbf{r}_{(it)} + \beta_{(it)}\,\mathbf{c}_{(it-1)}$
3 - $\alpha_{(it)} = \dfrac{\|\mathbf{r}_{(it)}\|_2^2}{\|\mathbf{A_m}\,\mathbf{c}_{(it)}\|_2^2}$
4 - $\tilde{\mathbf{p}}_{(it+1)} = \tilde{\mathbf{p}}_{(it)} + \alpha_{(it)}\,\mathbf{c}_{(it)}$
5 - $\mathbf{s}_{(it+1)} = \mathbf{s}_{(it)} - \alpha_{(it)}\,\mathbf{A_m}\,\mathbf{c}_{(it)}$
6 - $\mathbf{r}_{(it+1)} = \mathbf{A_m}^\top\,\mathbf{s}_{(it+1)}$
7 - $it = it + 1$
8 - Repeat previous steps until convergence (stops if $\delta = \dfrac{|\mathbf{r}_{(it+1)} - \mathbf{r}_{(it)}|}{N} \leq 10^{-3}$).
```

Setting a convergence criterion δ (Algorithm 1) based on the minimum tolerance of the residuals is a good option to carry out this algorithm efficiently and still obtaining very good results. Another possibility is to set an invariance limit to the normalized Euclidean norm of residuals between iterations, which would increase algorithm runtime, but with smaller residuals. We chose the latter option, as we could achieve better results.

Chapter 3

Methodology

3.1 Regular grids

3.1.1 x- and y-oriented grids

Consider that the observed data are located on an $N_x \times N_y$ regular grid of points regularly spaced by Δx and Δy along the x- and y-directions, respectively, on a horizontal plane defined by the constant vertical coordinate $z_0 < z_c$. As a consequence, a given pair of matrix coordinates (x_i, y_i), defined by the matrix index i, $i = 1, \ldots, N = N_x N_y$, is equivalent to a pair of coordinates (x_k, y_l) given by:

$$x_i \equiv x_k = x_1 + [k(i) - 1] \, \Delta x \, , \tag{3.1}$$

and

$$y_i \equiv y_l = y_1 + [l(i) - 1] \, \Delta y \, , \tag{3.2}$$

where $k(i)$ and $l(i)$ are integer functions of the matrix index i. These equations can also be used to define the matrix coordinates x_j and y_j associated with the jth equivalent source, $j = 1, \ldots, N = N_x N_y$. In this case, the integer functions are evaluated by using the index j instead of i. For convenience, we designate x_k and y_l as *grid coordinates* and the indices k and l as *grid indices*, which are computed with the integer functions.

The integer functions assume different forms depending on the orientation of the regular grid of data. Consider the case in which the grid is oriented along the x-axis (Figure 3.1 left panel). For convenience, we designate these grids as *x-oriented grids*. For them, we have the following integer functions:

$$i(k, l) = (l - 1) \, N_x + k \quad , \tag{3.3}$$

$$l(i) = \left\lceil \frac{i}{N_x} \right\rceil \tag{3.4}$$

and

$$k(i) = i - \left\lceil \frac{i}{N_x} \right\rceil N_x + N_x \quad , \tag{3.5}$$

where $\lceil \cdot \rceil$ denotes the ceiling function (GRAHAM *et al.*, 1994, p. 67). These integer functions are defined in terms of the matrix index i, but they can be defined in the same way by using the index j. Figure 3.1 left panel illustrates an x-oriented grid defined by $N_x = 3$ and $N_y = 2$. In this example, the matrix coordinates x_5 and y_5, defined by the matrix index $i = 5$ (or $j = 5$), are equivalent to the grid coordinates x_2 and y_2, which are defined by the grid indices $k = 2$ and $l = 2$, respectively. These indices are computed with equations 3.4 and 3.5, by using the matrix index $i = 5$ (or $j = 5$).

Now, consider the case in which the regular grid of data is oriented along the y-axis (Figure 3.1 right panel). For convenience, we call them *y-oriented grids*. Similarly to x-oriented grids, we have the following integer functions associated with y-oriented grids:

$$i(k, l) = (k - 1)\, N_y + l \quad , \tag{3.6}$$

$$k(i) = \left\lceil \frac{i}{N_y} \right\rceil \tag{3.7}$$

and

$$l(i) = i - \left\lceil \frac{i}{N_y} \right\rceil N_y + N_y \quad . \tag{3.8}$$

Figure 3.1 right panel illustrates an y-oriented grid defined by $N_x = 3$ and $N_y = 2$. In this example, the matrix coordinates x_5 and y_5, defined by the matrix index $i = 5$ (or $j = 5$), are equivalent to the grid coordinates x_3 and y_1, which are defined by the grid indices $k = 3$ and $l = 1$, respectively. Differently from the example shown in Figure 3.1 left panel, the grid indices of the present example are computed with equations 3.7 and 3.8, by using the matrix index $i = 5$ (or $j = 5$).

Using equations 3.4 or 3.8 and 3.5 or 3.7 it is also possible to define a difference between the grid indices:

$$\Delta l_{ij} = l(i) - l(j) \quad , \tag{3.9}$$

$$\Delta k_{ij} = k(i) - k(j) \quad . \tag{3.10}$$

In the following sections we will use this grid indices differences to represent the sensitivity matrices both of gravity and magntic cases.

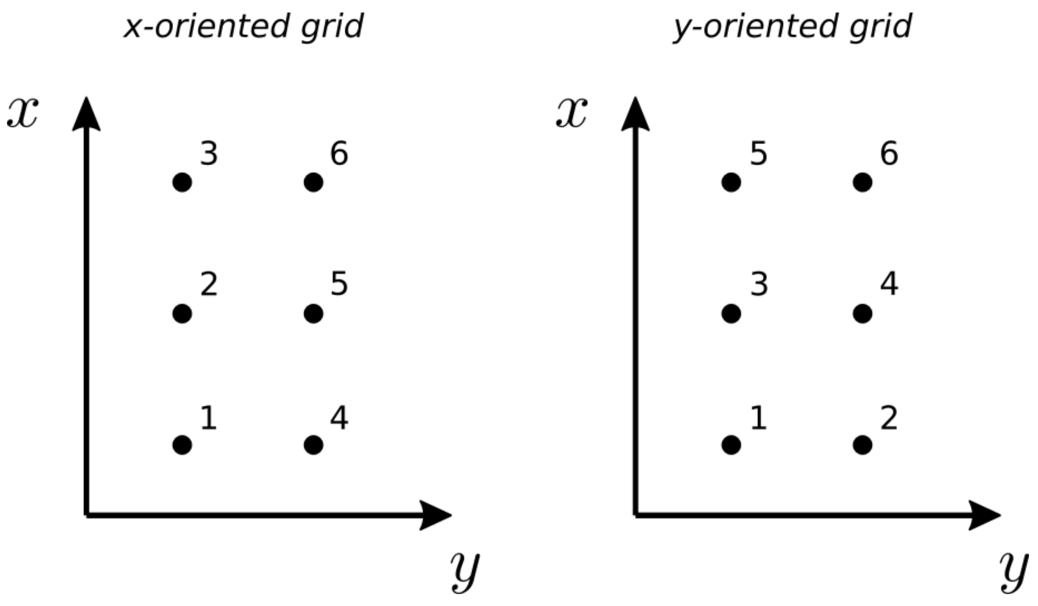

Figure 3.1: Schematic representation of an $N_x \times N_y$ regular grid of points (black dots) with $N_x = 3$ and $N_y = 2$, where each point has an associated index. This index may represent i or j, that are associated with observation points (x_i, y_i, z_0) and equivalent sources (x_j, y_j, z_c). Left panel shows an example of x-oriented grid, with indices varying along x-axis, while right panel shows an example of y-oriented grid, with indices varying along y-axis.

3.1.2 Elements of gravity sensitivity matrix for regular grids

To access the structure of the sensitivity matrix $\mathbf{A_g}$ (equation 2.1), let us first rewrite its elements a_{ij} (equation 2.2) by using equations 3.1 and 3.2, i.e.

$$a_{ij} = \frac{c_g \, G \, \Delta z}{\left[(\Delta k_{ij} \, \Delta x)^2 + (\Delta l_{ij} \, \Delta y)^2 + (\Delta z)^2\right]^{\frac{3}{2}}} \, , \tag{3.11}$$

where $\Delta_z = z_c - z_0$,

$$\Delta k_{ij} = \frac{x_i - x_j}{\Delta_x} = k(i) - k(j) \, , \tag{3.12}$$

$$\Delta l_{ij} = \frac{y_i - y_j}{\Delta_y} = l(i) - l(j) \tag{3.13}$$

and

$$\frac{1}{r_{ij}} = \frac{1}{\sqrt{(\Delta k_{ij} \, \Delta_x)^2 + (\Delta l_{ij} \, \Delta_y)^2 + \Delta_z^2}} \, . \tag{3.14}$$

Note that the integer functions $k(i)$, $k(j)$, $l(i)$ and $l(j)$ (equations 3.5–3.8) defining Δk_{ij} (equation 3.12), Δl_{ij} (equation 3.13) and $\frac{1}{r_{ij}}$ (equation 3.14) assume different forms depending on the grid orientation. Despite of that, it can be shown that

$$\Delta k_{ij} = -\Delta k_{ji} \, , \tag{3.15}$$

$$\Delta l_{ij} = -\Delta l_{ji} \tag{3.16}$$

and

$$\frac{1}{r_{ij}} = \frac{1}{r_{ji}} \tag{3.17}$$

for any grid orientation. Notice that the structure of matrix $\mathbf{A}$ (equation 2.1), for the case in which its elements are given by a_{ij} (equation 3.11), is defined by the coefficients Δk_{ij} and Δl_{ij}.

3.1.3 Elements of magnetic sensitivity matrix for regular grids

To access the structure of the sensitivity matrix $\mathbf{A_m}$ (equation 2.10), let us first rewrite its elements a_{ij} (equation 2.6) in the following way:

$$a_{ij} = a_{ij}^{xx} + a_{ij}^{xy} + a_{ij}^{xz} + a_{ij}^{yy} + a_{ij}^{yz} + a_{ij}^{zz} \ , \tag{3.18}$$

where

$$a_{ij}^{\alpha\beta} = \begin{cases} c_m \, \frac{\mu_0}{4\pi} \left(F_\alpha u_\beta \right) h_{ij}^{\alpha\beta} & , \quad \alpha = \beta \\ c_m \, \frac{\mu_0}{4\pi} \left(F_\alpha u_\beta + F_\beta u_\alpha \right) h_{ij}^{\alpha\beta} & , \quad \alpha \neq \beta \end{cases} , \quad \alpha, \beta = x, y, z \ , \tag{3.19}$$

are defined by the elements of $\hat{\mathbf{F}}$ (equation 2.4), $\hat{\mathbf{u}}$ (equation 2.7) and $\mathbf{H}_{ij}$ (equations 2.8 and 2.9). Then, we can rewrite the sensitivity matrix $\mathbf{A_m}$ (equation 2.10) as:

$$\mathbf{A_m} = \mathbf{A_{xx}} + \mathbf{A_{xy}} + \mathbf{A_{xz}} + \mathbf{A_{yy}} + \mathbf{A_{yz}} + \mathbf{A_{zz}} \ , \tag{3.20}$$

where $\mathbf{A}_{\boldsymbol{\alpha\beta}}$ are $N \times M$ matrices with elements ij defined by $a_{ij}^{\alpha\beta}$ (equation 3.19).

Now we can define the structure of $\mathbf{A_m}$ in terms of its components $\mathbf{A}_{\boldsymbol{\alpha\beta}}$ (equation 3.20). To do this, we consider the particular case in which the observed total-field anomaly is located on an $N_x \times N_y$ regular grid of points spaced by Δ_x and Δ_y along the x- and y-directions, respectively, on a constant vertical coordinate z_0. We also consider that the equivalent layer is formed by one dipole right below each observation point, at a constant coordinate z_c. In this case, the number of equivalent sources M is equal to the number of data N and, consequently, matrices $\mathbf{A_m}$ and $\mathbf{A}_{\boldsymbol{\alpha\beta}}$ become square ($N \times N$).

By using equations 3.1–3.8 to define the coordinates x_i and y_i of the observation points and x_j and y_j of the equivalent sources, we can rewrite the elements $h_{ij}^{\alpha\beta}$

(equation 2.9) of matrix $\mathbf{H}_{ij}$ (equation 2.8) as follows:

$$h_{ij}^{xx} = \frac{3\left(\Delta k_{ij}\, \Delta_x\right)^2}{r_{ij}^5} - \frac{1}{r_{ij}^3} \, , \tag{3.21}$$

$$h_{ij}^{yy} = \frac{3\left(\Delta l_{ij}\, \Delta_y\right)^2}{r_{ij}^5} - \frac{1}{r_{ij}^3} \, , \tag{3.22}$$

$$h_{ij}^{zz} = \frac{3\Delta_z^2}{r_{ij}^5} - \frac{1}{r_{ij}^3} \, , \tag{3.23}$$

$$h_{ij}^{xy} = \frac{3\left(\Delta k_{ij}\, \Delta_x\right)\left(\Delta l_{ij}\, \Delta_y\right)}{r_{ij}^5} \, , \tag{3.24}$$

$$h_{ij}^{xz} = \frac{3\left(\Delta k_{ij}\, \Delta_x\right)\Delta_z}{r_{ij}^5} \tag{3.25}$$

and

$$h_{ij}^{yz} = \frac{3\left(\Delta l_{ij}\, \Delta_y\right)\Delta_z}{r_{ij}^5} \, . \tag{3.26}$$

3.2 General structure of sensitivity matrices

For x-oriented grids, the coefficients Δk_{ij} and Δl_{ij} are computed by using equations 3.5 and 3.4, respectively. In this case, $\mathbf{A}_{\boldsymbol{g}}$, $\mathbf{A}_{\boldsymbol{m}}$ or $\mathbf{A}_{\boldsymbol{\alpha\beta}}$ (equations 2.1, 2.10 or 3.20) are composed of $N_y \times N_y$ blocks, where each block is formed by $N_x \times N_x$ elements. For y-oriented grids, the coefficients Δk_{ij} and Δl_{ij} are computed by using equations 3.7 and 3.8, respectively. In this case, $\mathbf{A}_{\boldsymbol{g}}$, $\mathbf{A}_{\boldsymbol{m}}$ or $\mathbf{A}_{\boldsymbol{\alpha\beta}}$ (equations 2.1, 2.10 or 3.20) are composed of $N_x \times N_x$ blocks, where each block is formed by $N_y \times N_y$ elements. In all cases, the matrices are Toeplitz blockwise, i.e., the blocks lying at the same block diagonal are equal to each other and each block are Toeplitz matrices themselves. Matrices with this well-defined pattern are called Doubly Block Toeplitz (JAIN, 1989, p. 28) or Block-Toeplitz Toeplitz-Block (BTTB), for example. We opted for using the second term.

This well-defined pattern is better represented by using the *block indices* q and p. For *x-oriented grids* (Figure 3.1 left panel), $Q = N_y$, $P = N_x$ and the block indices q and p are given by:

$$q \equiv q(i,j) = \Delta l_{ij} \tag{3.27}$$

and

$$p \equiv p(i,j) = \Delta k_{ij} \, , \tag{3.28}$$

where Δk_{ij} and Δl_{ij} (equations 3.15 and 3.16) are defined by integer functions $k(i)$, $k(j)$, $l(i)$ and $l(j)$ given by equations 3.5 and 3.4. For *y-oriented grids* (Figure 3.1

right panel), $Q = N_x$, $P = N_y$ and the block indices q and p are given by:

$$q \equiv q(i,j) = \Delta k_{ij} \tag{3.29}$$

and

$$p \equiv p(i,j) = \Delta l_{ij} \ , \tag{3.30}$$

where Δk_{ij} and Δl_{ij} (equations 3.12 and 3.13) are defined by integer functions $k(i)$, $k(j)$, $l(i)$ and $l(j)$ given by equations 3.7 and 3.8. Equations 3.27–3.30 show that q varies from $-Q+1$ to $Q-1$ and p from $-P+1$ to $P-1$, regardless of the grid orientation.

Let us consider the small regular grid of $N_x = 3$ and $N_y = 2$ points shown by Figure 3.1. This grid may represent observation points (x_i, y_i, z_0) with constant vertical coordinate z_0 or equivalent sources (x_j, y_j, z_c) with constant vertical coordinate $z_c > z_0$. In both cases, the horizontal coordinates are defined by equations 3.1 and 3.2. Given an index i, associated with an observation point, and an index j, associated with an equivalent source, we can compute Δk_{ij} (equation 3.12), Δl_{ij} (equation 3.13) and $\frac{1}{r_{ij}}$ (equation 3.14). The matrices $\mathbf{\Delta K}$ and $\mathbf{\Delta L}$ having elements ij defined by Δk_{ij} and Δl_{ij}, respectively, assume different forms, depending on the grid orientation. For x-oriented grids (Figure 3.1 left panel), they are given by:

$$\mathbf{\Delta K} = \begin{bmatrix} 0 & -1 & -2 & 0 & -1 & -2 \\ 1 & 0 & -1 & 1 & 0 & -1 \\ 2 & 1 & 0 & 2 & 1 & 0 \\ 0 & -1 & -2 & 0 & -1 & -2 \\ 1 & 0 & -1 & 1 & 0 & -1 \\ 2 & 1 & 0 & 2 & 1 & 0 \end{bmatrix} \tag{3.31}$$

and

$$\mathbf{\Delta L} = \begin{bmatrix} 0 & 0 & 0 & -1 & -1 & -1 \\ 0 & 0 & 0 & -1 & -1 & -1 \\ 0 & 0 & 0 & -1 & -1 & -1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \ . \tag{3.32}$$

For y-oriented grids (Figure 3.1 right panel), they are given by:

$$\Delta\mathbf{K} = \begin{bmatrix} 0 & 0 & -1 & -1 & -2 & -2 \\ 0 & 0 & -1 & -1 & -2 & -2 \\ 1 & 1 & 0 & 0 & -1 & -1 \\ 1 & 1 & 0 & 0 & -1 & -1 \\ 2 & 2 & 1 & 1 & 0 & 0 \\ 2 & 2 & 1 & 1 & 0 & 0 \end{bmatrix} \tag{3.33}$$

and

$$\Delta\mathbf{L} = \begin{bmatrix} 0 & -1 & 0 & -1 & 0 & -1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & -1 & 0 & -1 & 0 & -1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & -1 & 0 & -1 & 0 & -1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} . \tag{3.34}$$

For example, consider the matrix coordinates x_5, y_5 for the observation point and x_3, y_3 for the source in a x-oriented grid, these are defined by the matrix index $i = 5$ and $j = 3$. Using equations 3.5 and 3.4 the grid indices $k(i) = 2$, $l(i) = 2$ and $k(j) = 3$, $l(j) = 1$ can be calculated. Equations 3.12 and 3.13 define $\Delta k_{ij} = -1$ and $\Delta l_{ij} = 1$, which represents the the value in the fifth row and third column of matrices $\Delta\mathbf{K}$ and $\Delta\mathbf{L}$ (equations 3.31 and 3.32), respectively. Using the matrix coordinates x_4 (y_4) for the observation point and x_2 (y_2) for the source, lead us to the same values for Δk_{ij} and Δl_{ij} (fourth row and second column of equations 3.31 and 3.32). These examples (equations 3.31–3.34) show that different combinations of indices i and j result in integer functions Δk_{ij} and Δl_{ij} (equations 3.12 and 3.13) having the same numerical value. In these cases, not only the numerical values of the corresponding elements $a_{ij}^{\alpha\beta}$ (equation 3.19), but also their associated block indices q and p (equations 3.27–3.30) are the same. The contrary is also true: elements $a_{ij}^{\alpha\beta}$ having different associated block indices q and p also have different numerical values. Because of that, using the alternative notation $a_{qp}^{\alpha\beta}$ to define the elements $a_{ij}^{\alpha\beta}$ in terms of its associated block indices q and p is a good approach to investigating the structure of a given matrix component $\mathbf{A}_{\boldsymbol{\alpha\beta}}$ (equation 3.20). This approach allows identifying elements $a_{ij}^{\alpha\beta}$ having the same numerical value only by inspecting their associated block indices.

Note that, for x-oriented grids, matrices $\Delta\mathbf{K}$ (equation 3.31) and $\Delta\mathbf{L}$ (equation 3.32) define the block indices p (equation 3.28) and q (equation 3.27), respectively. In this case, they are composed of $Q \times Q$ blocks with $P \times P$ elements each, where $Q = N_y$ and $P = N_x$. For y-oriented grids, matrices $\Delta\mathbf{K}$ (equation 3.33) and $\Delta\mathbf{L}$

(equation 3.34) define the block indices q (equation 3.29) and p (equation 3.30), respectively. In this case, they are also composed of $Q \times Q$ blocks with $P \times P$ elements each, but now $Q = N_x$ and $P = N_y$. The examples shown by equations 3.31–3.34 also illustrate that, regardless of grid orientation, (i) the block index q is constant inside each block; (ii) blocks disposed along the same block diagonal are equal to each other; (iii) the block index p is constant on each diagonal of a given block; (iv) elements of a given block located on the same diagonal are also equal do each other. The results obtained with the small grid shown in Figure 3.1 can be easily generalized for larger grids. Based on the well-defined structure of block indices, we can define matrices $\mathbf{A}_{\boldsymbol{g}}$, $\mathbf{A}_{\boldsymbol{m}}$ or $\mathbf{A}_{\boldsymbol{\alpha\beta}}$ in a general form

$$\mathbf{A}_{\boldsymbol{g}},\, \mathbf{A}_{\boldsymbol{m}},\, \mathbf{A}_{\boldsymbol{\alpha\beta}} \equiv \mathbf{A} = \begin{bmatrix} \mathbf{A}^{0} & \mathbf{A}^{-1} & \cdots & \mathbf{A}^{-Q+1} \\ \mathbf{A}^{1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{A}^{-1} \\ \mathbf{A}^{Q-1} & \cdots & \mathbf{A}^{1} & \mathbf{A}^{0} \end{bmatrix}_{N \times N} , \tag{3.35}$$

with blocks $\mathbf{A}^{q}_{\boldsymbol{g}}$, $\mathbf{A}^{q}_{\boldsymbol{m}}$ or $\mathbf{A}^{q}_{\boldsymbol{\alpha\beta}}$, $q = -Q+1, \ldots, Q-1$, given by

$$\mathbf{A}^{q}_{\boldsymbol{g}},\, \mathbf{A}^{q}_{\boldsymbol{m}},\, \mathbf{A}^{q}_{\boldsymbol{\alpha\beta}} \equiv \mathbf{A}^{q} = \begin{bmatrix} a_{q0} & a_{q(-1)} & \cdots & a_{q(-P+1)} \\ a_{q1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{q(-1)} \\ a_{q(P-1)} & \cdots & a_{q1} & a_{q0} \end{bmatrix}_{P \times P} , \tag{3.36}$$

formed by elements a^{g}_{qp}, a^{m}_{qp} or $a^{\alpha\beta}_{qp}$, $p = -P+1, \ldots, P-1$.

3.3 Detailed structure of the sensitivity matrices

3.3.1 Detailed structure of the gravity sensitivity matrix

From equations 3.35 and 3.36 we can define the structure of matrix $\mathbf{A}_g$ (equation 2.1). Considering the elements a^{g}_{ij} (equation 2.2), defined by the first vertical derivative of equation 2.3, it is possible to verify from equations 3.15 and 3.17 that $a^{g}_{ij} = a^{g}_{ji}$. As a consequence

$$\mathbf{A}_{\boldsymbol{g}} = (\mathbf{A}_{\boldsymbol{g}})^{\top} , \tag{3.37}$$

for both x- and y-oriented grids. Using a x-oriented grid (Figure 3.1 left panel), the block indices q and p are defined by equations 3.27 and 3.28 and a^{g}_{qp} can be rewritten as follows

$$a^{g}_{qp} = \frac{c_g \, G \, \Delta_z}{[(p\,\Delta_x)^2 + (q\,\Delta_y)^2 + \Delta_z^2]^{\frac{3}{2}}} , \tag{3.38}$$

where $\Delta_z = z_c - z_i$.

For y-oriented grids (Figure 3.1 right panel), the block indices q and p are defined by equations 3.29 and 3.30 thus, a^g_{qp} becomes

$$a^g_{qp} = \frac{c_g \, G \, \Delta_z}{[(q\,\Delta_x)^2 + (p\,\Delta_y)^2 + \Delta_z^2]^{\frac{3}{2}}} \ . \tag{3.39}$$

Equations 3.38 and 3.39 show us that

$$\mathbf{A}_{\boldsymbol{g}}^{q} = \mathbf{A}_{\boldsymbol{g}}^{(-q)} \ , \tag{3.40}$$

which demonstrates the symmetry by blocks of matrix $\mathbf{A}_g$ and

$$\mathbf{A}_{\boldsymbol{g}}^{q} = \left(\mathbf{A}_{\boldsymbol{g}}^{q}\right)^{\top} \ , \tag{3.41}$$

that demonstrates the symmetry inside the blocks of matrix $\mathbf{A}_g$. Therefore, $\mathbf{A}_g$ has a strucuture of a *symmetric-Block-Toeplitz symmetric-Toeplitz-Block* matrix for x- and y-orientation grids. Using the symmetry presented in equations 3.40 and 3.41, the gravity sensitiviry matrix from equations 3.35 and 3.36 can be rewritten as

$$\mathbf{A}_{\boldsymbol{g}} = \begin{bmatrix} \mathbf{A}^0 & \mathbf{A}^1 & \cdots & \mathbf{A}^{Q-1} \\ \mathbf{A}^1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{A}^1 \\ \mathbf{A}^{Q-1} & \cdots & \mathbf{A}^1 & \mathbf{A}^0 \end{bmatrix}_{N \times N} \ , \tag{3.42}$$

with blocks $\mathbf{A}_{\boldsymbol{g}}^{q}$, $q = -Q+1, \ldots, Q-1$, given by

$$\mathbf{A}_{\boldsymbol{g}}^{q} = \begin{bmatrix} a_{q0} & a_{q(1)} & \cdots & a_{q(P-1)} \\ a_{q1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{q(1)} \\ a_{q(P-1)} & \cdots & a_{q1} & a_{q0} \end{bmatrix}_{P \times P} \ . \tag{3.43}$$

Figures 3.2 and 3.3 shows a example of this type of BTTB matrix structure for a 3×2 x- and y-oriented grids, respectively.

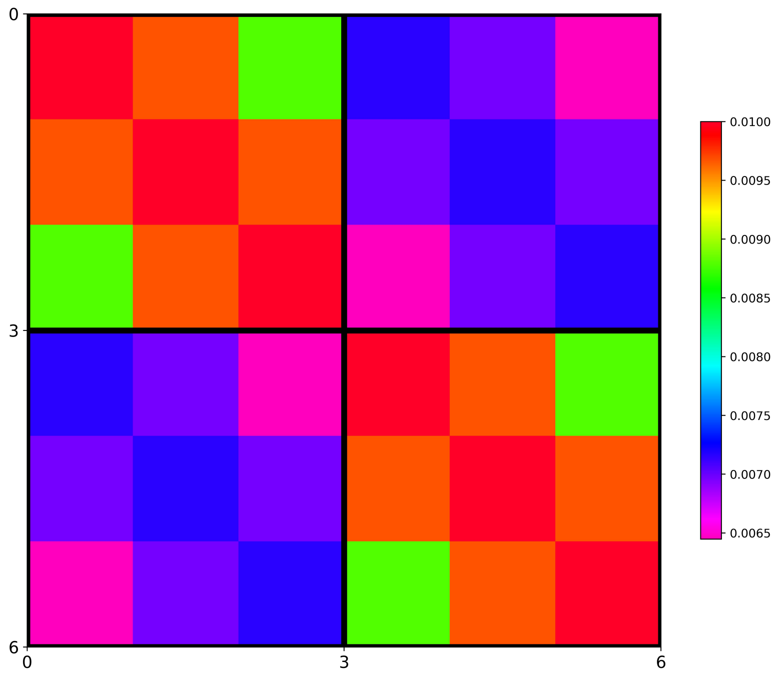

Figure 3.2: Example of a *symmetric-Block-Toeplitz symmetric-Toeplitz-Block* matrix for 3×2 x-oriented grid. This matrix represents the structure of the gravity sensitivity matrix $\mathbf{A}_g$.

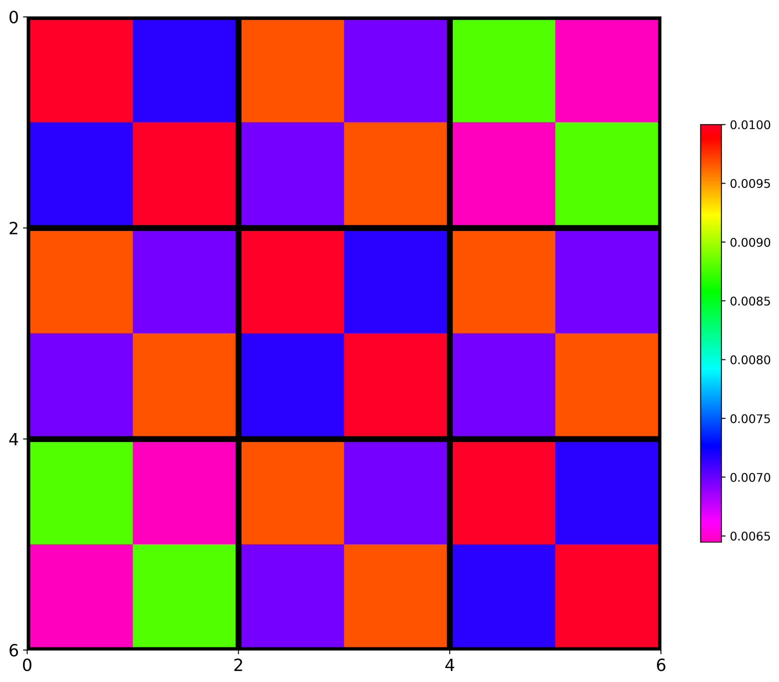

Figure 3.3: Example of a *symmetric-Block-Toeplitz symmetric-Toeplitz-Block* matrix for 3×2 y-oriented grid. This matrix represents the structure of the gravity sensitivity matrix $\mathbf{A}_g$.

3.3.2 Detailed structure of matrices $\mathbf{A_{xx}}$, $\mathbf{A_{yy}}$ and $\mathbf{A_{zz}}$

The direct second derivatives matrices $\mathbf{A_{xx}}$, $\mathbf{A_{yy}}$ and $\mathbf{A_{zz}}$ have the same strucuture of the gravity sensitivity matrix. Equations 3.35 and 3.36 also define the general BTTB structure of all matrix components $\mathbf{A}_{\boldsymbol{\alpha\beta}}$, but there are some differences between them. Let us consider the matrix component $\mathbf{A}_{\boldsymbol{xx}}$, with elements a^{xx}_{ij} (equation 3.19) defined by the second derivative h^{xx}_{ij} (equation 3.21). It can be easily verified from equations 3.15 and 3.17 that $h^{xx}_{ij} = h^{xx}_{ji}$. As a consequence, $a^{xx}_{ij} = a^{xx}_{ji}$, which means that

$$\mathbf{A}_{\boldsymbol{xx}} = (\mathbf{A}_{\boldsymbol{xx}})^{\top} \tag{3.44}$$

for any grid orientation. Now, let us investigate the elements a^{xx}_{qp} forming the blocks $\mathbf{A}^{q}_{\boldsymbol{xx}}$. For x-oriented grids (Figure 3.1 left panel), the block indices q and p are defined by equations 3.27 and 3.28 and a^{xx}_{qp} can be rewritten as follows:

$$a^{xx}_{qp} = c_m \, \frac{\mu_0}{4\pi} \, (F_x u_x) \, \frac{3\,(p\,\Delta_x)^2}{r^5_{qp}} - \frac{1}{r^3_{qp}} \, , \tag{3.45}$$

where

$$\frac{1}{r_{qp}} = \frac{1}{\sqrt{(p\,\Delta_x)^2 + (q\,\Delta_y)^2 + \Delta_z^2}} \,. \tag{3.46}$$

For y-oriented grids (Figure 3.1 right panel), the block indices q and p are defined by equations 3.29 and 3.30 and a_{qp}^{xx} can be rewritten as follows:

$$a_{qp}^{xx} = c_m \, \frac{\mu_0}{4\pi} \, (F_x u_x) \, \frac{3\,(q\,\Delta_x)^2}{r_{qp}^5} - \frac{1}{r_{qp}^3} \,, \tag{3.47}$$

where

$$\frac{1}{r_{qp}} = \frac{1}{\sqrt{(q\,\Delta_x)^2 + (p\,\Delta_y)^2 + \Delta_z^2}} \,. \tag{3.48}$$

From equations 3.45–3.48, we can easily verify that

$$\mathbf{A}_{\boldsymbol{xx}}^{q} = \mathbf{A}_{\boldsymbol{xx}}^{(-q)} \tag{3.49}$$

and

$$\mathbf{A}_{\boldsymbol{xx}}^{q} = (\mathbf{A}_{\boldsymbol{xx}}^{q})^{\top} \,. \tag{3.50}$$

Note that these symmetries are valid for any grid orientation. From this results we conclude the matrix component $\mathbf{A}_{\boldsymbol{xx}}$ is *symmetric-Block-Toeplitz symmetric-Toeplitz-Block* for any grid orientation. The same reasoning can be used to show that matrices $\mathbf{A}_{\boldsymbol{yy}}$ and $\mathbf{A}_{\boldsymbol{zz}}$ also have this symmetric structure. Figure 3.4 panels a), d) and f) show examples of this type of BTTB matrices structures when 3×2 x-oriented grids are used for $\mathbf{A_{xx}}$, $\mathbf{A_{yy}}$ and $\mathbf{A_{zz}}$, respectively. Figure 3.5 panels a), d) and f) show examples of this type of BTTB matrices structures when 3×2 y-oriented grids are used for $\mathbf{A_{xx}}$, $\mathbf{A_{yy}}$ and $\mathbf{A_{zz}}$, respectively.

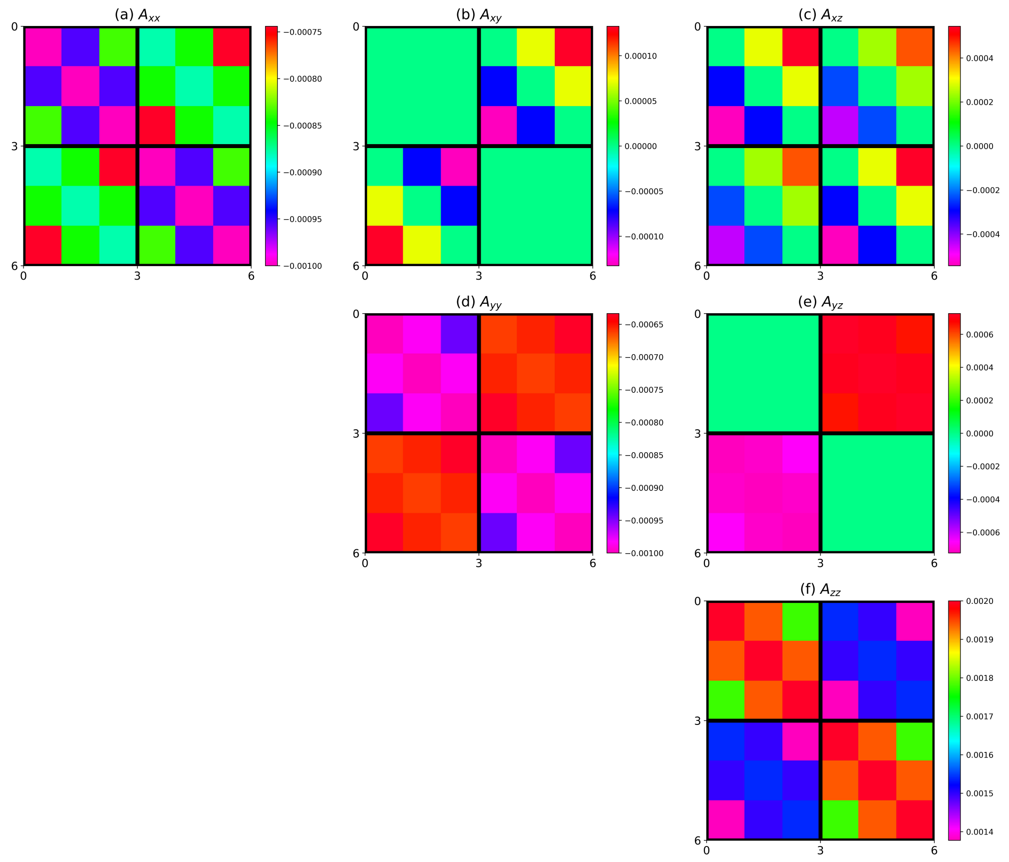

Figure 3.4: This figure shows examples of all the BTTB structures possible for $\mathbf{A}_{\alpha\beta}$ matrix when 3×2 x-oriented grids are used. Panel a) Example of a *symmetric-Block-Toeplitz symmetric-Toeplitz-Block* structure of the $\mathbf{A_{xx}}$ matrix. Panel b) Example of a *skew symmetric-Block-Toeplitz skew symmetric-Toeplitz-Block* structure of the $\mathbf{A_{xy}}$ matrix. Panel c) Example of a *symmetric-Block-Toeplitz skew symmetric-Toeplitz-Block* structure of the $\mathbf{A_{xz}}$ matrix for x-oriented grids. Panel d) Example of a *symmetric-Block-Toeplitz symmetric-Toeplitz-Block* structure of the $\mathbf{A_{yy}}$ matrix. Panel e) Example of a *skew symmetric-Block-Toeplitz symmetric-Toeplitz-Block* structure of the $\mathbf{A_{yz}}$ matrix for x-oriented grids. Panel f) Example of a *symmetric-Block-Toeplitz symmetric-Toeplitz-Block* structure of the $\mathbf{A_{zz}}$ matrix.

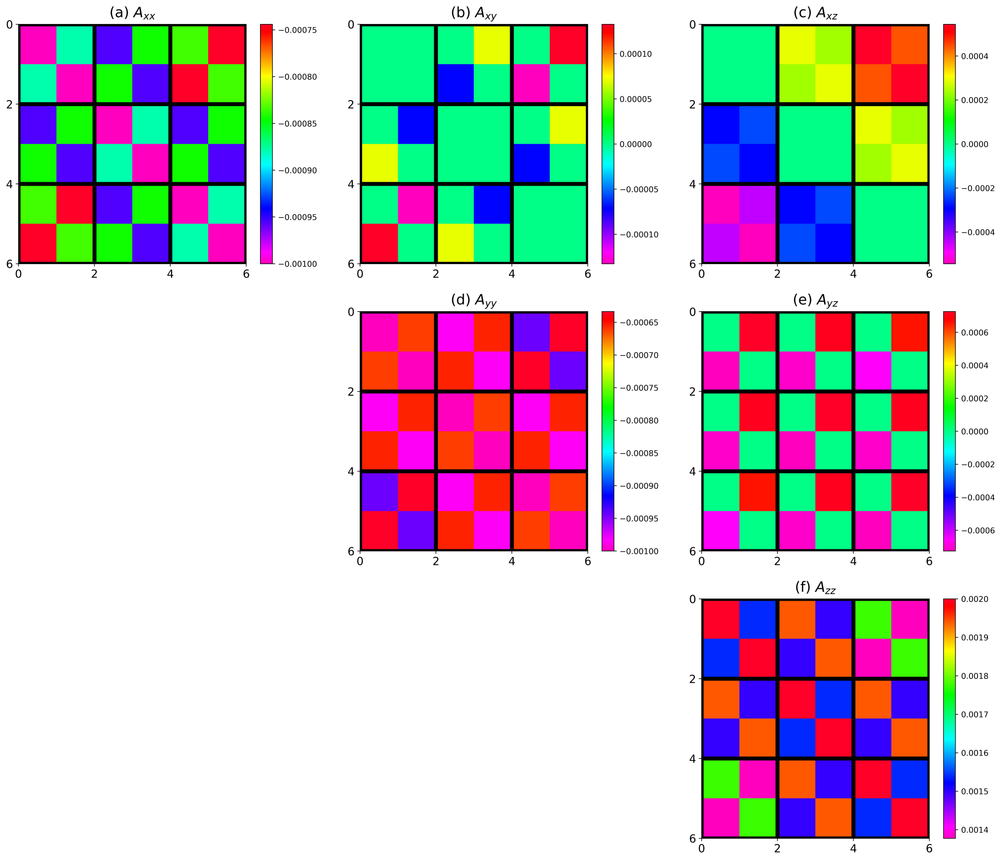

Figure 3.5: This figure shows examples of all the BTTB structures possible for $\mathbf{A}_{\alpha\beta}$ matrix when 3×2 y-oriented grids are used. Panel a) Example of a *symmetric-Block-Toeplitz symmetric-Toeplitz-Block* structure of the $\mathbf{A_{xx}}$ matrix. Panel b) Example of a *skew symmetric-Block-Toeplitz skew symmetric-Toeplitz-Block* structure of the $\mathbf{A_{xy}}$ matrix. Panel c) Example of a *skew symmetric-Block-Toeplitz symmetric-Toeplitz-Block* structure of the $\mathbf{A_{xz}}$ matrix for y-oriented grids. Panel d) Example of a *symmetric-Block-Toeplitz symmetric-Toeplitz-Block* structure of the $\mathbf{A_{yy}}$ matrix. Panel e) Example of a *symmetric-Block-Toeplitz skew symmetric-Toeplitz-Block* structure of the $\mathbf{A_{yz}}$ matrix for y-oriented grids. Panel f) Example of a *symmetric-Block-Toeplitz symmetric-Toeplitz-Block* structure of the $\mathbf{A_{zz}}$ matrix.

3.3.3 Detailed structure of matrix $\mathbf{A_{xy}}$

Let $\mathbf{A}_{\boldsymbol{xy}}$ be a matrix component with elements a^{xy}_{ij} (equation 3.19) defined by the second derivative h^{xy}_{ij} (equation 3.24). It can be easily verified from equations 3.15–3.17 that $h^{xy}_{ij} = h^{xy}_{ji}$. As a consequence, $a^{xy}_{ij} = a^{xy}_{ji}$, which means that

$$\mathbf{A}_{\boldsymbol{xy}} = (\mathbf{A}_{\boldsymbol{xy}})^{\top} \tag{3.51}$$

for any grid orientation. For x-oriented grids (Figure 3.1 left panel), the block indices q and p are defined by equations 3.27 and 3.28 and a^{xy}_{qp} can be rewritten as follows:

$$a^{xy}_{qp} = c_m \, \frac{\mu_0}{4\pi} \left(F_x u_y + F_y u_x\right) \frac{3\,(p\,\Delta_x)\,(q\,\Delta_y)}{r^5_{qp}} \, , \tag{3.52}$$

with $\frac{1}{r_{qp}}$ defined by equation 3.46. For y-oriented grids (Figure 3.1 right panel), the block indices q and p are defined by equations 3.29 and 3.30 and a^{xy}_{qp} can be rewritten as follows:

$$a^{xy}_{qp} = c_m \, \frac{\mu_0}{4\pi} \left(F_x u_y + F_y u_x\right) \frac{3\,(q\,\Delta_x)\,(p\,\Delta_y)}{r^5_{qp}} \, , \tag{3.53}$$

with $\frac{1}{r_{qp}}$ defined by equation 3.48. From equations 3.46, 3.48, 3.52 and 3.53, we can show that

$$\mathbf{A}^{q}_{\boldsymbol{xy}} = -\mathbf{A}^{(-q)}_{\boldsymbol{xy}} \tag{3.54}$$

and

$$\mathbf{A}^{q}_{\boldsymbol{xy}} = -\left(\mathbf{A}^{q}_{\boldsymbol{xy}}\right)^{\top} \, . \tag{3.55}$$

Note that these symmetries are valid for any grid orientation. From this results we conclude the matrix component $\mathbf{A}_{\boldsymbol{xy}}$ is *skew symmetric-Block-Toeplitz skew symmetric-Toeplitz-Block* for any grid orientation. Figure 3.4 panel b) shows a example of this type of BTTB matrix structure for 3×2 x-oriented grids. Figure 3.5 panel b) shows a example for 3×2 y-oriented grids.

3.3.4 Detailed structure of matrices $\mathbf{A_{xz}}$ and $\mathbf{A_{yz}}$

Let $\mathbf{A}_{\boldsymbol{xz}}$ be a matrix component with elements a^{xz}_{ij} (equation 3.19) defined by the second derivative h^{xz}_{ij} (equation 3.25). It can be easily verified from equations 3.15–3.17 that $h^{xz}_{ij} = -h^{xz}_{ji}$. As a consequence, $a^{xz}_{ij} = -a^{xz}_{ji}$, which means that

$$\mathbf{A}_{\boldsymbol{xz}} = -\left(\mathbf{A}_{\boldsymbol{xz}}\right)^{\top} \tag{3.56}$$

for any grid orientation. For x-oriented grids (Figure 3.1 left panel), the block indices q and p are defined by equations 3.27 and 3.28 and a^{xz}_{qp} can be rewritten as follows:

$$a^{xz}_{qp} = c_m \, \frac{\mu_0}{4\pi} \left(F_x u_z + F_z u_x\right) \frac{3\,(p\,\Delta_x)\,\Delta_z}{r^5_{qp}} \, , \tag{3.57}$$

with $\frac{1}{r_{qp}}$ defined by equation 3.46. In this case, we can see that

$$\mathbf{A}^{q}_{\boldsymbol{xz}} = \mathbf{A}^{(-q)}_{\boldsymbol{xz}} \tag{3.58}$$

and

$$\mathbf{A}^{q}_{\boldsymbol{xz}} = -\left(\mathbf{A}^{q}_{\boldsymbol{xz}}\right)^{\top} . \tag{3.59}$$

This structure is called *symmetric-Block-Toeplitz skew symmetric-Toeplitz-Block* and is valid only for x-oriented grids. For y-oriented grids (Figure 3.1 right panel), the block indices q and p are defined by equations 3.29 and 3.30 and a^{xz}_{qp} can be rewritten as follows:

$$a^{xz}_{qp} = c_m \frac{\mu_0}{4\pi} \left(F_x u_z + F_z u_x\right) \frac{3\left(q\,\Delta_x\right)\Delta_z}{r^5_{qp}} , \tag{3.60}$$

with $\frac{1}{r_{qp}}$ defined by equation 3.48. Now, we conclude that

$$\mathbf{A}^{q}_{\boldsymbol{xz}} = -\mathbf{A}^{(-q)}_{\boldsymbol{xz}} \tag{3.61}$$

and

$$\mathbf{A}^{q}_{\boldsymbol{xz}} = \left(\mathbf{A}^{q}_{\boldsymbol{xz}}\right)^{\top} . \tag{3.62}$$

This structure is called *skew symmetric-Block-Toeplitz symmetric-Toeplitz-Block* and is valid only for y-oriented grids.

The same reasoning can be followed to show that

$$\mathbf{A}_{\boldsymbol{yz}} = -\left(\mathbf{A}_{\boldsymbol{yz}}\right)^{\top} \tag{3.63}$$

for any grid orientation. Besides, we can also show that

$$\mathbf{A}^{q}_{\boldsymbol{yz}} = -\mathbf{A}^{(-q)}_{\boldsymbol{yz}} \tag{3.64}$$

and

$$\mathbf{A}^{q}_{\boldsymbol{yz}} = \left(\mathbf{A}^{q}_{\boldsymbol{yz}}\right)^{\top} \tag{3.65}$$

for x-oriented grids (*skew symmetric-Block-Toeplitz symmetric-Toeplitz-Block*), while

$$\mathbf{A}^{q}_{\boldsymbol{yz}} = \mathbf{A}^{(-q)}_{\boldsymbol{yz}} \tag{3.66}$$

and

$$\mathbf{A}^{q}_{\boldsymbol{yz}} = -\left(\mathbf{A}^{q}_{\boldsymbol{yz}}\right)^{\top} \tag{3.67}$$

for y-oriented grids (*symmetric-Block-Toeplitz skew symmetric-Toeplitz-Block*). Figure 3.4 panels c) and e) show examples of this type of BTTB matrix structure for 3×2 x-oriented grids. Figure 3.5 panels c) and e) show examples for 3×2 y-oriented grids

3.4 BTTB matrix-vector product

3.4.1 BTTB matrix-vector product for gravimetric data processing

The matrix-vector product $\mathbf{A}_{\boldsymbol{g}}\tilde{\mathbf{p}}^k$ (equation 2.17) required by the fast equivalent-layer technique (SIQUEIRA *et al.*, 2017) accounts for most of its total computation time and can cause computer memory shortage when large data sets are used. This computational load can be drastically reduced by exploring the well-defined structure of matrix $\mathbf{A}_{\boldsymbol{g}}$ (equation 2.1) for the particular case in which its elements a_{ij} are defined by equation 3.11. In this case, $\mathbf{A}_{\boldsymbol{g}}$ is a symmetric BTTB matrix (equations 3.37 and 3.40–3.41) and the predicted data vector $\mathbf{d}(\mathbf{p})$ (equation 2.1) can be efficiently computed by using the 2D Discrete Fourier Transform (DFT). To do this, let us first rewrite $\mathbf{d}(\mathbf{p})$ and $\mathbf{p}_{\boldsymbol{g}}$ (equation 2.1) as the following partitioned vectors:

$$\mathbf{d}(\mathbf{p}) = \begin{bmatrix} \mathbf{d}_0(\mathbf{p}) \\ \vdots \\ \mathbf{d}_{Q-1}(\mathbf{p}) \end{bmatrix}_{N\times 1} \tag{3.68}$$

and

$$\mathbf{p} = \begin{bmatrix} \mathbf{p}_0 \\ \vdots \\ \mathbf{p}_{Q-1} \end{bmatrix}_{N\times 1} , \tag{3.69}$$

where $\mathbf{d}_q(\mathbf{p})$ and $\mathbf{p}_q$, $q = 0, \ldots, Q-1$, are $P \times 1$ vectors. Notice that q is the block index defined by equations 3.27 and 3.29, Q defines the number of blocks $\mathbf{A}_{\boldsymbol{g}}^q$ (equation 3.36) forming $\mathbf{A}_{\boldsymbol{g}}$ (equation 2.1) and P defines the number of elements forming each block $\mathbf{A}_{\boldsymbol{g}}^q$. Then, by using the partitioned vectors (equations 3.69 and 3.68) and remembering that $N = QP$, we define the auxiliary linear system

$$\mathbf{w} = \mathbf{C}\mathbf{v} \, , \tag{3.70}$$

where

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_0 \\ \vdots \\ \mathbf{w}_{Q-1} \\ \mathbf{0}_{2N\times 1} \end{bmatrix}_{4N\times 1} , \tag{3.71}$$

$$\mathbf{w}_q = \begin{bmatrix} \mathbf{d}_q(\mathbf{p}) \\ \mathbf{0}_{P\times 1} \end{bmatrix}_{2P\times 1} , \tag{3.72}$$

$$\mathbf{v} = \begin{bmatrix} \mathbf{v}_0 \\ \vdots \\ \mathbf{v}_{Q-1} \\ \mathbf{0}_{2N\times 1} \end{bmatrix}_{4N\times 1} , \tag{3.73}$$

and

$$\mathbf{v}_q = \begin{bmatrix} \mathbf{p}_q \\ \mathbf{0}_{P\times 1} \end{bmatrix}_{2P\times 1} , \tag{3.74}$$

with $\mathbf{d}_q(\mathbf{p})$ and $\mathbf{p}_q$ defined by equations 3.68 and 3.69, respectively. Finally $\mathbf{C}$ (equation 3.70) is a $4N \times 4N$ symmetric Block Circulant matrix with Circulant Blocks (BCCB) (DAVIS, 1979, p. 184). Matrix $\mathbf{C}$ (equation 3.70) is circulant blockwise, formed by $2Q \times 2Q$ blocks, where each block $\mathbf{C}_q$, $q = 0, \ldots, Q-1$, is a $2P \times 2P$ circulant matrix. Similarly to the BTTB matrix $\mathbf{A}_{\boldsymbol{g}}$ (equation 3.42), the index q varies from 0 to $Q-1$. Additionally, the blocks lying above the main diagonal are equal to those located below.

It is well-known that a circulant matrix can be defined by properly downshifting its first column (VAN LOAN, 1992, p. 206). Hence, the BCCB matrix $\mathbf{C}$ (equation 3.70) can be obtained from its first column of blocks, which is given by

$$[\mathbf{C}]_{(0)} = \begin{bmatrix} \mathbf{C}_0 \\ \vdots \\ \mathbf{C}_{Q-1} \\ \mathbf{0} \\ \mathbf{C}_{Q-1} \\ \vdots \\ \mathbf{C}_1 \end{bmatrix}_{4N\times 2P} , \tag{3.75}$$

where $\mathbf{0}$ is a $2P \times 2P$ matrix of zeros. Similarly, each block $\mathbf{C}_q$, $q = 0, \ldots, Q-1$, can be obtained by downshifting its first column

$$\mathbf{c}_0^q = \begin{bmatrix} a_0^q \\ \vdots \\ a_{P-1}^q \\ 0 \\ a_{P-1}^q \\ \vdots \\ a_1^q \end{bmatrix}_{2P\times 1} , \tag{3.76}$$

where a_p^q (equation 3.39), $p = 0, \ldots, P-1$, are the elements forming the block $\mathbf{A}_{\boldsymbol{g}}^q$ (equation 3.43). The downshift can be thought off as permutation that pushes the components of a column vector down one notch with wraparound (GOLUB and LOAN, 2013, p. 20). To illustrate this operation, consider our y-oriented grid illustrated in the right panel of Figure 3.1. In this case, the resulting BCCB matrix $\mathbf{C}$ (equation 3.70) is given by

$$\mathbf{C} = \begin{bmatrix} \mathbf{C_0} & \mathbf{C_1} & \mathbf{C_2} & \mathbf{C_3} & \mathbf{0} & \mathbf{C_3} & \mathbf{C_2} & \mathbf{C_1} \\ \mathbf{C_1} & \mathbf{C_0} & \mathbf{C_1} & \mathbf{C_2} & \mathbf{C_3} & \mathbf{0} & \mathbf{C_3} & \mathbf{C_2} \\ \mathbf{C_2} & \mathbf{C_1} & \mathbf{C_0} & \mathbf{C_1} & \mathbf{C_2} & \mathbf{C_3} & \mathbf{0} & \mathbf{C_3} \\ \mathbf{C_3} & \mathbf{C_2} & \mathbf{C_1} & \mathbf{C_0} & \mathbf{C_1} & \mathbf{C_2} & \mathbf{C_3} & \mathbf{0} \\ \mathbf{0} & \mathbf{C_3} & \mathbf{C_2} & \mathbf{C_1} & \mathbf{C_0} & \mathbf{C_1} & \mathbf{C_2} & \mathbf{C_3} \\ \mathbf{C_3} & \mathbf{0} & \mathbf{C_3} & \mathbf{C_2} & \mathbf{C_1} & \mathbf{C_0} & \mathbf{C_1} & \mathbf{C_2} \\ \mathbf{C_2} & \mathbf{C_3} & \mathbf{0} & \mathbf{C_3} & \mathbf{C_2} & \mathbf{C_1} & \mathbf{C_0} & \mathbf{C_1} \\ \mathbf{C_1} & \mathbf{C_2} & \mathbf{C_3} & \mathbf{0} & \mathbf{C_3} & \mathbf{C_2} & \mathbf{C_1} & \mathbf{C_0} \end{bmatrix}_{4N \times 4N} , \tag{3.77}$$

where each block $\mathbf{C}_q$, $q = 0, 1, 2, 3$, is represented as follows

$$\mathbf{C}_q = \begin{bmatrix} a_0^q & a_1^q & a_2^q & 0 & a_2^q & a_1^q \\ a_1^q & a_0^q & a_1^q & a_2^q & 0 & a_2^q \\ a_2^q & a_1^q & a_0^q & a_1^q & a_2^q & 0 \\ 0 & a_2^q & a_1^q & a_0^q & a_1^q & a_2^q \\ a_2^q & 0 & a_2^q & a_0^q & a_0^q & a_1^q \\ a_1^q & a_2^q & 0 & a_2^q & a_1^q & a_0^q \end{bmatrix}_{2P \times 2P} \tag{3.78}$$

in terms of the block elements a_p^q (equation 3.39). Similar matrices are obtained for our x-oriented grid illustrated in Figure 3.1a.

BCCB matrices are diagonalized by the 2D unitary DFT (DAVIS, 1979, p. 185). It means that $\mathbf{C}$ (equation 3.70) satisfies

$$\mathbf{C} = \left(\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P}\right)^{*} \mathbf{\Lambda} \left(\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P}\right) , \tag{3.79}$$

where the symbol "$\otimes$" denotes the Kronecker product (NEUDECKER, 1969), $\mathbf{F}_{2Q}$ and $\mathbf{F}_{2P}$ are the $2Q \times 2Q$ and $2P \times 2P$ unitary DFT matrices (DAVIS, 1979, p. 31), respectively, the superscritpt "$*$" denotes the complex conjugate and $\mathbf{\Lambda}$ is a $4QP \times 4QP$ diagonal matrix containing the eigenvalues of $\mathbf{C}$.

What follows shows a step-by-step description of how we use the auxiliary system (equation 3.70) to compute the matrix-vector product $\mathbf{A_g}\tilde{\mathbf{p}}^k$ (equation 2.17) in a computationally efficient way by exploring the structure of matrix $\mathbf{C}$. By substituting equation 3.79 in the auxiliary system (equation 3.70) and premultiplying both sides of the result by $(\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P})$ (see the details in section 3.4.2), we obtain

$$\mathbf{\Lambda} \left(\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P}\right) \mathbf{v} = \left(\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P}\right) \mathbf{w} \, . \tag{3.80}$$

Now, by applying the *vec*-operator to both sides of equation 3.80 (see the details in section 3.4.3), we obtain:

$$\mathbf{F}^{*}_{2Q} \left[\mathbf{L} \circ \left(\mathbf{F}_{2Q} \, \mathbf{V} \, \mathbf{F}_{2P}\right)\right] \mathbf{F}^{*}_{2P} = \mathbf{W} \, , \tag{3.81}$$

where "$\circ$" denotes the Hadamard product (HORN and JOHNSON, 1991, p. 298) and $\mathbf{L}$, $\mathbf{V}$ and $\mathbf{W}$ are $2Q \times 2P$ matrices obtained by rearranging, along their rows, the elements forming the diagonal of matrix $\mathbf{\Lambda}$, vector $\mathbf{v}$ and vector $\mathbf{w}$, respectively. The left side of equation 3.81 contains the 2D Inverse Discrete Fourier Transform (IDFT) of the term in brackets, which in turn represents the Hadamard product of matrix $\mathbf{L}$ (equation 3.112) and the 2D DFT of matrix $\mathbf{V}$ (see equations 3.112 and 3.110 in section 3.4.3). Matrix $\mathbf{L}$ contains the eigenvalues of $\mathbf{\Lambda}$ (equation 3.79) and can be efficiently computed by using only the first column of the BCCB matrix $\mathbf{C}$ (equation 3.70) (see the details in section 3.4.4). Here, we evaluate equation 3.81 and compute matrix $\mathbf{L}$ by using the 2D Fast Fourier Transform (2D FFT). This approach, that have been used in potential-field methods (e.g., QIANG *et al.*, 2019; ZHANG and WONG, 2015; ZHANG *et al.*, 2016), is actually a fast 2D discrete convolution (e.g., VAN LOAN, 1992, p. 213).

At each iteration kth of the fast equivalent-layer technique, (equation 2.17), we efficiently compute $\mathbf{A}_{\boldsymbol{g}}\tilde{\mathbf{p}}^{k} = \mathbf{d}(\tilde{\mathbf{p}}^{k})$ by following the steps below:

(1) Use equation 3.11 to compute the first column of each block $\mathbf{A}^{q}_{\boldsymbol{g}}$ (equation 3.43), $q = 0, \ldots, Q-1$, forming the BTTB matrix $\mathbf{A}_{\boldsymbol{g}}$ (equation 3.42);

(2) Rearrange the first column of $\mathbf{A}_{\boldsymbol{g}}$ according to equations 3.75 and 3.76 to obtain the first column $\mathbf{c}_0$ of the BCCB matrix $\mathbf{C}$ (equation 3.70);

(3) Rearrange $\mathbf{c}_0$ along the rows and use the 2D FFT to compute matrix $\mathbf{L}$ (equation 3.116, section 3.4.3);

(4) Rearrange the parameter vector $\tilde{\mathbf{p}}^{k}$ (equation 2.1) in its partitioned form (equation 3.69) to define the auxiliary vector $\mathbf{v}$ (equation 3.73);

(5) Rearrange $\mathbf{v}$ to obtain matrix $\mathbf{V}$, use the 2D FFT to compute its DFT and evaluate the left side of equation 3.113(see section 3.4.3);

(6) Use the 2D FFT to compute the IDFT of the result obtained in step (5) to obtain the matrix $\mathbf{W}$ (equation 3.81);

(7) Use the *vec*-operator (equation 3.103, section 3.4.3) and equations 3.71 and 3.72 to rearrange $\mathbf{W}$ in order to obtain the predicted data vector $\mathbf{d}(\tilde{\mathbf{p}}^k)$.

3.4.2 BTTB matrix-vector product for magnetic data processing

To efficiently compute the product of the sensitivity matrix $\mathbf{A_m}$ (equation 2.10) and a generic vector $\mathbf{b}$ for the magnetic equivalent layer let this product be represented by

$$\mathbf{t} = \mathbf{A_m}\,\mathbf{b}\,, \tag{3.82}$$

where

$$\mathbf{t} = \mathbf{t}_{\boldsymbol{xx}} + \mathbf{t}_{\boldsymbol{xy}} + \mathbf{t}_{\boldsymbol{xz}} + \mathbf{t}_{\boldsymbol{yy}} + \mathbf{t}_{\boldsymbol{yz}} + \mathbf{t}_{\boldsymbol{zz}} \tag{3.83}$$

and

$$\mathbf{t}_{\boldsymbol{\alpha\beta}} = \mathbf{A}^{\mathbf{m}}_{\boldsymbol{\alpha\beta}}\,\mathbf{b}\,. \tag{3.84}$$

Let us also consider that vectors

$$\mathbf{t}_{\boldsymbol{\alpha\beta}} = \begin{bmatrix} \mathbf{t}^{0}_{\boldsymbol{\alpha\beta}} \\ \vdots \\ \mathbf{t}^{Q-1}_{\boldsymbol{\alpha\beta}} \end{bmatrix}_{N\times 1} \tag{3.85}$$

and

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}^{0} \\ \vdots \\ \mathbf{b}^{Q-1} \end{bmatrix}_{N\times 1} \tag{3.86}$$

are composed of $P \times 1$ vectors $\mathbf{t}^{q}_{\boldsymbol{\alpha\beta}}$ and $\mathbf{b}^{q}$, respectively, where q is the block index (equations 3.27 and 3.29). From equation 3.84, we obtain an auxiliary matrix-vector product given by

$$\mathbf{w}_{\boldsymbol{\alpha\beta}} = \mathbf{C}_{\boldsymbol{\alpha\beta}}\,\mathbf{v}\,, \tag{3.87}$$

where $\mathbf{C}_{\boldsymbol{\alpha\beta}}$ is a $4N \times 4N$ block circulant matrix with circulant blocks (BCCB) (e.g., DAVIS, 1979, p. 184),

$$\mathbf{w}_{\boldsymbol{\alpha\beta}} = \begin{bmatrix} \mathbf{w}^{0}_{\boldsymbol{\alpha\beta}} \\ \vdots \\ \mathbf{w}^{Q-1}_{\boldsymbol{\alpha\beta}} \\ \mathbf{0}_{2N\times 1} \end{bmatrix}_{4N\times 1}\,, \tag{3.88}$$

$$\mathbf{w}^{q}_{\boldsymbol{\alpha\beta}} = \begin{bmatrix} \mathbf{t}^{q}_{\boldsymbol{\alpha\beta}} \\ \mathbf{0}_{P\times 1} \end{bmatrix}_{2P\times 1}\,, \tag{3.89}$$

$$\mathbf{v} = \begin{bmatrix} \mathbf{v}^0 \\ \vdots \\ \mathbf{v}^{Q-1} \\ \mathbf{0}_{2N\times 1} \end{bmatrix}_{4N\times 1} \tag{3.90}$$

and

$$\mathbf{v}^q = \begin{bmatrix} \mathbf{b}^q \\ \mathbf{0}_{P\times 1} \end{bmatrix}_{2P\times 1} , \tag{3.91}$$

with $\mathbf{0}_{2N\times 1}$ and $\mathbf{0}_{P\times 1}$ being vectors of zeros. As shown in section 3.4.1 the auxiliary matrix-vector product (equation 3.87) represents a 2D discrete convolution and can also be efficiently computed by using the 2D Fast Fourier Transform (2D FFT).

The BCCB matrix $\mathbf{C}_{\boldsymbol{\alpha\beta}}$ (equation 3.87) is formed by $2Q \times 2Q$ blocks, where each block $\mathbf{C}^q_{\boldsymbol{\alpha\beta}}$ is a $2P \times 2P$ circulant matrix. The entire BCCB matrix $\mathbf{C}_{\boldsymbol{\alpha\beta}}$ is defined by properly downshifting its first block column

$$[\mathbf{C}_{\boldsymbol{\alpha\beta}}]_{(0)} = \begin{bmatrix} \mathbf{C}^0_{\boldsymbol{\alpha\beta}} \\ \vdots \\ \mathbf{C}^{Q-1}_{\boldsymbol{\alpha\beta}} \\ \mathbf{0}_{2P\times 2P} \\ \mathbf{C}^{-Q+1}_{\boldsymbol{\alpha\beta}} \\ \vdots \\ \mathbf{C}^{-1}_{\boldsymbol{\alpha\beta}} \end{bmatrix}_{4N\times 2P} , \tag{3.92}$$

where $\mathbf{0}_{2P\times 2P}$ is a matrix of zeros. Similarly, each block $\mathbf{C}^q_{\boldsymbol{\alpha\beta}}$, $q = -Q+1, \ldots, Q-1$, is obtained by properly downshifting its first column

$$\mathbf{c}^q_{\boldsymbol{\alpha\beta}} = \begin{bmatrix} a^{\alpha\beta}_{q0} \\ \vdots \\ a^{\alpha\beta}_{q(P-1)} \\ 0 \\ a^{\alpha\beta}_{q(-P+1)} \\ \vdots \\ a^{\alpha\beta}_{q(-1)} \end{bmatrix}_{2P\times 1} , \tag{3.93}$$

where $a^{\alpha\beta}_{qp}$, $p = -P+1, \ldots, P-1$, are the elements of matrix component $\mathbf{A}_{\boldsymbol{\alpha\beta}}$ described in terms of block indices q and p (equations 3.27–3.30). The BCCB matrix $\mathbf{C}_{\boldsymbol{\alpha\beta}}$ is diagonalized by $\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P}$, where "$\otimes$" denotes the Kronecker product (e.g., HORN and JOHNSON, 1991, p. 242) and $\mathbf{F}_{2Q}$ and $\mathbf{F}_{2P}$ are the $2Q \times 2Q$ and $2P \times 2P$ unitary DFT matrices (DAVIS, 1979, p. 31). Due to this property, the auxiliary matrix-vector product (equation 3.87) can be computed as follows

$$\mathbf{F}_{2Q}^{*}\left[\mathbf{L}_{\boldsymbol{\alpha\beta}} \circ (\mathbf{F}_{2Q}\,\mathbf{V}\,\mathbf{F}_{2P})\right]\mathbf{F}_{2P}^{*} = \mathbf{W}_{\boldsymbol{\alpha\beta}} \,, \tag{3.94}$$

where "$\circ$" denotes the Hadamard (element-wise) product (e.g., HORN and JOHNSON, 1991, p. 298), "$*$" denotes the complex conjugate, $\mathbf{W}_{\boldsymbol{\alpha\beta}}$ and $\mathbf{V}$ are $2Q \times 2P$ matrices obtained by rearranging, respectively, vectors $\mathbf{w}_{\boldsymbol{\alpha\beta}}$ (equation 3.88) and $\mathbf{v}$ (equation 3.90) along their rows and $\mathbf{L}_{\boldsymbol{\alpha\beta}}$ is a $2Q \times 2P$ matrix given by

$$\mathbf{L}_{\boldsymbol{\alpha\beta}} = \sqrt{4QP}\,\mathbf{F}_{2Q}\,\mathbf{G}_{\boldsymbol{\alpha\beta}}\,\mathbf{F}_{2P} \,, \tag{3.95}$$

with

$$\mathbf{G}_{\boldsymbol{\alpha\beta}} = \begin{bmatrix} \left(\mathbf{c}_{\boldsymbol{\alpha\beta}}^{0}\right)^{\top} \\ \vdots \\ \left(\mathbf{c}_{\boldsymbol{\alpha\beta}}^{Q-1}\right)^{\top} \\ \mathbf{0}_{1\times 2P} \\ \left(\mathbf{c}_{\boldsymbol{\alpha\beta}}^{-Q+1}\right)^{\top} \\ \vdots \\ \left(\mathbf{c}_{\boldsymbol{\alpha\beta}}^{-1}\right)^{\top} \end{bmatrix}_{2Q\times 2P} \,, \tag{3.96}$$

defined by the first columns $\mathbf{c}_{\boldsymbol{\alpha\beta}}^{q}$ (equation 3.93), $q = -Q+1, \ldots, Q-1$, of all circulant blocks $\mathbf{C}_{\boldsymbol{\alpha\beta}}^{q}$ (equation 3.92). Hence, the whole BCCB matrix $\mathbf{C}_{\boldsymbol{\alpha\beta}}$ does not have to be formed, but only its first column. Besides, the symmetries defined by equations 3.44–3.67 imply that all elements of $\mathbf{G}_{\boldsymbol{\alpha\beta}}$ can be obtained by using only the first column of $\mathbf{A}_{\boldsymbol{\alpha\beta}}$. Consequently, the whole matrices $\mathbf{A}_{\boldsymbol{\alpha\beta}}$ do not have to be formed as well, but only their first columns.

It is important noting that the left side of equation 3.94 represents the 2D Inverse Discrete Fourier Transform (2D IDFT) of the term in brackets. This term, in turn, represents the Hadamard product of $\mathbf{L}_{\boldsymbol{\alpha\beta}}$ (equation 3.95) and the 2D Discrete Fourier Transform (2D DFT) of $\mathbf{V}$. Similarly, equation 3.95 shows that $\mathbf{L}_{\boldsymbol{\alpha\beta}}$ is obtained by computing the 2D DFT of matrix $\mathbf{G}_{\boldsymbol{\alpha\beta}}$ (equation 3.96). Hence, equations 3.94 and 3.95 can be efficiently computed by using the 2D FFT. After that, the elements of vector $\mathbf{t}_{\boldsymbol{\alpha\beta}}$ (equation 3.84) can be retrieved from the first quadrant of matrix $\mathbf{W}_{\boldsymbol{\alpha\beta}}$ (equation 3.94). By combining the results obtained for all components $\boldsymbol{\alpha\beta}$, $\boldsymbol{\alpha}, \boldsymbol{\beta} = \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$, we can show that

$$\mathbf{F}_{2Q}^{*}\left[\mathbf{L} \circ (\mathbf{F}_{2Q}\,\mathbf{V}\,\mathbf{F}_{2P})\right]\mathbf{F}_{2P}^{*} = \mathbf{W} \,, \tag{3.97}$$

where

$$\mathbf{W} = \mathbf{W}_{\boldsymbol{xx}} + \mathbf{W}_{\boldsymbol{xy}} + \mathbf{W}_{\boldsymbol{xz}} + \mathbf{W}_{\boldsymbol{yy}} + \mathbf{W}_{\boldsymbol{yz}} + \mathbf{W}_{\boldsymbol{zz}} \tag{3.98}$$

and

$$\mathbf{L} = \mathbf{L}_{\boldsymbol{xx}} + \mathbf{L}_{\boldsymbol{xy}} + \mathbf{L}_{\boldsymbol{xz}} + \mathbf{L}_{\boldsymbol{yy}} + \mathbf{L}_{\boldsymbol{yz}} + \mathbf{L}_{\boldsymbol{zz}} \ , \tag{3.99}$$

with $\mathbf{L}_{\boldsymbol{\alpha\beta}}$ defined by equation 3.95. Then, the elements of $\mathbf{t}$ (equation 3.82) are obtained from the first quadrant of $\mathbf{W}$ (equations 3.97 and 3.99).

Finally, it can be shown that the product

$$\mathbf{t} = \mathbf{A_m}^{\top}\mathbf{b} \tag{3.100}$$

can be computed by using equation 3.97. The difference is that, in this case, matrices $\mathbf{G}_{\boldsymbol{\alpha\beta}}$ (equation 3.96) are defined by using the new vectors

$$\mathbf{c}^{q}_{\boldsymbol{\alpha\beta}} = \begin{bmatrix} a^{\alpha\beta}_{q0} \\ \vdots \\ a^{\alpha\beta}_{q(-P+1)} \\ 0 \\ a^{\alpha\beta}_{q(P-1)} \\ \vdots \\ a^{\alpha\beta}_{q1} \end{bmatrix}_{2P\times 1} . \tag{3.101}$$

3.4.3 Computations with the 2D DFT

In the present section, we deduce equations 3.81 and 3.95 by using the row-ordered *vec*-operator (here designated simply as *vec*-operator). This equation can be efficiently computed by using the 2D fast Fourier Transform. This operator was implicitly used by JAIN (1989, p. 31) to show the relationship between Kronecker products and separable transformations. The *vec*-operator defined here transforms a matrix into a column vector by stacking its rows.

Let $\mathbf{M}$ be an arbitrary $N \times M$ matrix given by:

$$\mathbf{M} = \begin{bmatrix} \mathbf{m}_1^{\top} \\ \vdots \\ \mathbf{m}_N^{\top} \end{bmatrix} , \tag{3.102}$$

where $\mathbf{m}_i$, $i = 1, \ldots, N$, are $M \times 1$ vectors containing the rows of $\mathbf{M}$. The elements of this matrix can be rearranged into a column vector by using the *vec*-operator (JAIN, 1989, p. 31) as follows:

$$vec\,(\mathbf{M}) = \begin{bmatrix} \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_N \end{bmatrix}_{NM\times 1} . \tag{3.103}$$

This rearrangement is known as lexicographic ordering (JAIN, 1989, p. 150).

Two important properties of the *vec*-operator (equation 3.103) are necessary to us. To define the first one, consider an $N \times M$ matrix $\mathbf{H}$ given by

$$\mathbf{H} = \mathbf{P} \circ \mathbf{Q} \, , \tag{3.104}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are arbitrary $N \times M$ matrices and "$\circ$" represents the Hadamard product (HORN and JOHNSON, 1991, p. 298). By applying the *vec*-operator to $\mathbf{H}$ (equation 3.104), it can be shown that

$$vec\left(\mathbf{H}\right) = vec\left(\mathbf{P}\right) \circ vec\left(\mathbf{Q}\right) \; . \tag{3.105}$$

To define the second important property of *vec*-operator, consider an $N \times M$ matrix $\mathbf{S}$ defined by the separable transformation JAIN (1989, p. 31):

$$\mathbf{S} = \mathbf{P} \, \mathbf{M} \, \mathbf{Q} \, , \tag{3.106}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are arbitrary $N \times N$ and $M \times M$ matrices, respectively. By implicitly applying the *vec*-operator to the $\mathbf{S}$ (equation 3.106), JAIN (1989, p. 31) showed that:

$$vec\left(\mathbf{S}\right) = \left(\mathbf{P} \otimes \mathbf{Q}^{\top}\right) vec\left(\mathbf{M}\right) \; , \tag{3.107}$$

where "$\otimes$" denotes the Kronecker product (NEUDECKER, 1969). It is important to stress the difference between equation 3.107 and that presented by NEUDECKER (1969), which is more commonly found in the literature. While that equation uses a *vec*-operator that transforms a matrix into a column vector by stacking its columns, equation 3.107 uses the *vec*-operator defined by equation 3.103, which transforms a matrix into a column vector by stacking its rows.

Now, let us deduce equations 3.81 and 3.95 by using the above-defined properties (equation 3.105 and 3.107). We start calling attention to the right side of equation 3.80. Consider that vector $\mathbf{w}$ (equation 3.80) is obtained by applying the *vec*-operator (equation 3.103) to a matrix $\mathbf{W}$, whose 2D DFT $\tilde{\mathbf{W}}$ is represented by the following separable transformation (JAIN, 1989, p. 146):

$$\tilde{\mathbf{W}} = \mathbf{F}_{2Q} \, \mathbf{W} \, \mathbf{F}_{2P} \; , \tag{3.108}$$

where $\mathbf{F}_{2Q}$ and $\mathbf{F}_{2P}$ are the $2Q \times 2Q$ and $2P \times 2P$ unitary DFT matrices. Using equation 3.107 and the symmetry of unitary DFT matrices, we rewrite the right side of equation 3.80 as follows:

$$vec\left(\tilde{\mathbf{W}}\right) = \left(\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P}\right) vec\left(\mathbf{W}\right) \; . \tag{3.109}$$

Similarly, consider that $\mathbf{v}$ (equation 3.80) is obtained by applying the *vec*-operator (equation 3.103) to a matrix $\mathbf{V}$, whose 2D DFT (equation 3.108) is represented by $\tilde{\mathbf{V}}$. Using equation 3.107 and the symmetry of unitary DFT matrices, we can rewrite the left side of equation 3.80 as follows:

$$\mathbf{\Lambda}\, vec\left(\tilde{\mathbf{V}}\right) = \mathbf{\Lambda}\left(\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P}\right) vec\left(\mathbf{V}\right) \ . \tag{3.110}$$

Note that both sides of equation 3.110 are defined as the product of the diagonal matrix $\mathbf{\Lambda}$ (equation 3.79) and a vector. In this case, the matrix-vector product can be conveniently replaced by

$$\boldsymbol{\lambda} \circ vec\left(\tilde{\mathbf{V}}\right) = \boldsymbol{\lambda} \circ \left(\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P}\right) vec\left(\mathbf{V}\right) \ , \tag{3.111}$$

where $\boldsymbol{\lambda}$ is a $4QP \times 1$ vector containing the diagonal of $\mathbf{\Lambda}$ (equation 3.79). Then, consider that $\boldsymbol{\lambda}$ is obtained by applying the *vec*-operator (equation 3.103) to a $2Q \times 2P$ matrix $\mathbf{L}$, we can use equations 3.105 and 3.107 to rewrite equation 3.111 as follows:

$$vec\left(\mathbf{L} \circ \tilde{\mathbf{V}}\right) = vec\left[\mathbf{L} \circ \left(\mathbf{F}_{2Q}\, \mathbf{V}\, \mathbf{F}_{2P}\right)\right] \ . \tag{3.112}$$

Equations 3.108, 3.109 and 3.112 show that equation 3.80 is obtained by applying the *vec*-operator to

$$\mathbf{L} \circ \left(\mathbf{F}_{2Q}\, \mathbf{V}\, \mathbf{F}_{2P}\right) = \mathbf{F}_{2Q}\, \mathbf{W}\, \mathbf{F}_{2P} \, . \tag{3.113}$$

Finally, we premultiply both sides of equation 3.113 by $\mathbf{F}_{2Q}^{*}$ and then postmultiply both sides of the result by $\mathbf{F}_{2P}^{*}$ to deduce equation 3.81.

3.4.4 The eigenvalues of C

In the present section, we show how to efficiently compute matrix $\mathbf{L}$ (equations 3.112, 3.113, 3.81 and 3.95) by using only the first column of the BCCB matrix $\mathbf{C}$ (equations 3.70 and 3.87).

We need first premultiply both sides of equation 3.79 by $\left(\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P}\right)$ to obtain

$$\left(\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P}\right) \mathbf{C} = \mathbf{\Lambda}\left(\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P}\right) \ . \tag{3.114}$$

From equation 3.114, we can easily show that (CHAN and JIN, 2007, p. 77):

$$\left(\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P}\right)\, \mathbf{c}_0 = \frac{1}{\sqrt{4QP}}\, \boldsymbol{\lambda} \, , \tag{3.115}$$

where $\mathbf{c}_0$ is a $4QP \times 1$ vector representing the first column of $\mathbf{C}$ (equations 3.70 and 3.87) and $\boldsymbol{\lambda}$ (equation 3.111) is the $4QP \times 1$ vector that contains the diagonal of matrix $\mathbf{\Lambda}$ (equation 3.79) and is obtained by applying the *vec*-operator (equation

3.103) to matrix $\mathbf{L}$. Now, let us conveniently consider that $\mathbf{c}_0$ is obtained by applying the *vec*-operator to a $2Q \times 2P$ matrix $\mathbf{G}$. Using this matrix, the property of the *vec*-operator for separable transformations (equation 3.106) and the symmetry of unitary DFT matrices, equation 3.115 can be rewritten as follows

$$\mathbf{F}_{2Q}\, \mathbf{G}\, \mathbf{F}_{2P} = \frac{1}{\sqrt{4QP}}\, \mathbf{L}\,. \tag{3.116}$$

This equation shows that the eigenvalues of the BCCB matrix $\mathbf{C}$ (equations 3.70 and 3.87), forming the rows of $\mathbf{L}$, are obtained by computing the 2D DFT of matrix $\mathbf{G}$, which contains the elements forming the first column of the BCCB matrix $\mathbf{C}$ (equations 3.70 and 3.87).

3.5 Convolutional equivalent-layer processing

3.5.1 Convolutional equivalent layer for gravity data processing

In a normal procedure of the fast equivalent layer proposed by SIQUEIRA *et al.* (2017), at each iteration a full matrix $\mathbf{A}_{\boldsymbol{g}}$ (equation 2.1 and 3.42) is multiplied by the estimated mass distribution parameter vector $\tilde{\mathbf{p}}^k$ producing the predicted gravity data $\mathbf{d}(\mathbf{p})$ iteratively. By substituting this matrix-vector product following the steps from section 3.4.4 we will improve the computational efficiency of the technique.

3.5.2 Computational performance for gravity data processing

The number of flops (floating-point operations) necessary to estimate the $N \times 1$ parameter vector $\mathbf{p}$ in the fast equivalent-layer technique (SIQUEIRA *et al.*, 2017) is

$$f_0 = N^{it}(3N + 2N^2)\,, \tag{3.117}$$

where N^{it} is the number of iterations. In this equation, the term $2N^2$ is associated with the matrix-vector product $\mathbf{A}_{\boldsymbol{g}}\tilde{\mathbf{p}}^k$ (equation 2.17) and accounts for most of the computational complexity of this method. Our method replace this matrix-vector product by three operations: one DFT, one Hadamard product and one IDFT involving $2Q \times 2P$ matrices (left side of equation 3.81). The Hadamard product requires $24N$ flops, $N = QP$, because the entries are complex numbers. We consider that a DFT/IDFT requires $\kappa\, 4N \log_2(4N)$ flops to be computed via 2D FFT, where κ is a constant depending on the algorithm. Then, the resultant flops count of our method is given by:

$$f_1 = N^{it} \left[27N + \kappa \, 8N \log_2(4N) \right] \, . \tag{3.118}$$

Figure 3.6 shows the flops counts f_0 and f_1 (equations 3.117 and 3.118) associated with the fast equivalent-layer technique (SIQUEIRA *et al.*, 2017) and our method, respectively, as a function of the number N of observation points. We considered a fixed number of $N^{it} = 50$ iterations and $\kappa = 5$ (equation 3.118), which is compatible with a radix-2 FFT (VAN LOAN, 1992, p. 16). As we can see, the number of flops is drastically decreased in our method.

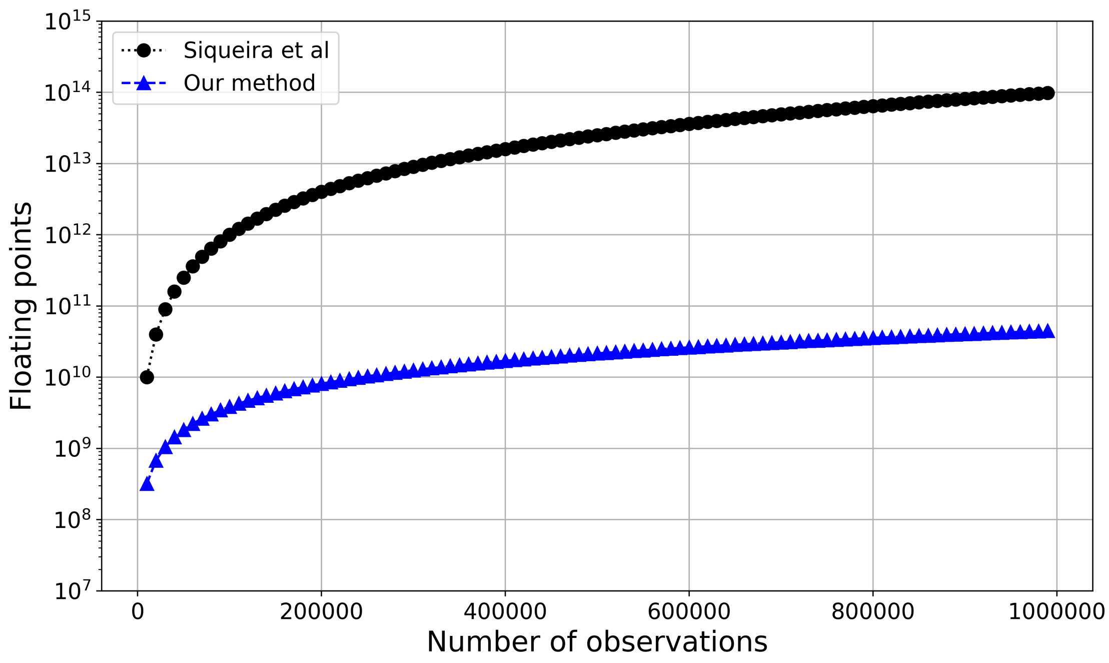

Figure 3.6: Comparison between the number of flops (equations 3.117 and 3.118) associated with the fast equivalent-layer technique (SIQUEIRA *et al.*, 2017) and our method, for N varying from $5,000$ to $1,000,000$. All values are computed with $N^{it} = 50$ iterations and $\kappa = 5$.

Another advantage of our method is concerned with the real $N \times N$ matrix $\mathbf{A_g}$ (equations 2.1 and 3.42). In the fast equivalent-layer technique, the full matrix is computed once and stored during the entire iterative process. On the other hand, our method computes only one column of $\mathbf{A_g}$ and uses it to compute the complex $2Q \times 2P$ matrix $\mathbf{L}$ (equation 3.116) via 2D FFT, which is stored during all iterations. Table 3.1 shows the computer memory usage needed to store the full matrix $\mathbf{A_g}$, a single column of $\mathbf{A_g}$ and the full matrix $\mathbf{L}$. These quantities were computed for different numbers of observations N. Notice that $N = 1,000,000$ observations require nearly 7.6 Terabytes of computer memory to store the whole matrix $\mathbf{A_g}$.

Figure 3.7 compares the running time of the fast equivalent-layer technique

(SIQUEIRA *et al.*, 2017) and of our method, considering a constant number of iterations $N^{it} = 50$. We used a PC with an Intel Core i7 4790@3.6GHz processor and 16 GB of computer memory. The computational efficiency of our approach is significantly superior to that of the fast equivalent-layer technique for a number of observations N greater than $10,000$. We could not perform this comparison with a number of observations greater than $22,500$ due to limitations of our PC in storing the full matrix $\mathbf{A}_{\boldsymbol{g}}$. Figure 3.8 shows the running time of our method with a number of observations up to 25 millions. These results shows that, while the running time of our method is ≈ 30.9 seconds for $N = 1,000,000$, the fast equivalent-layer technique spends ≈ 46.8 seconds for $N = 22,500$.

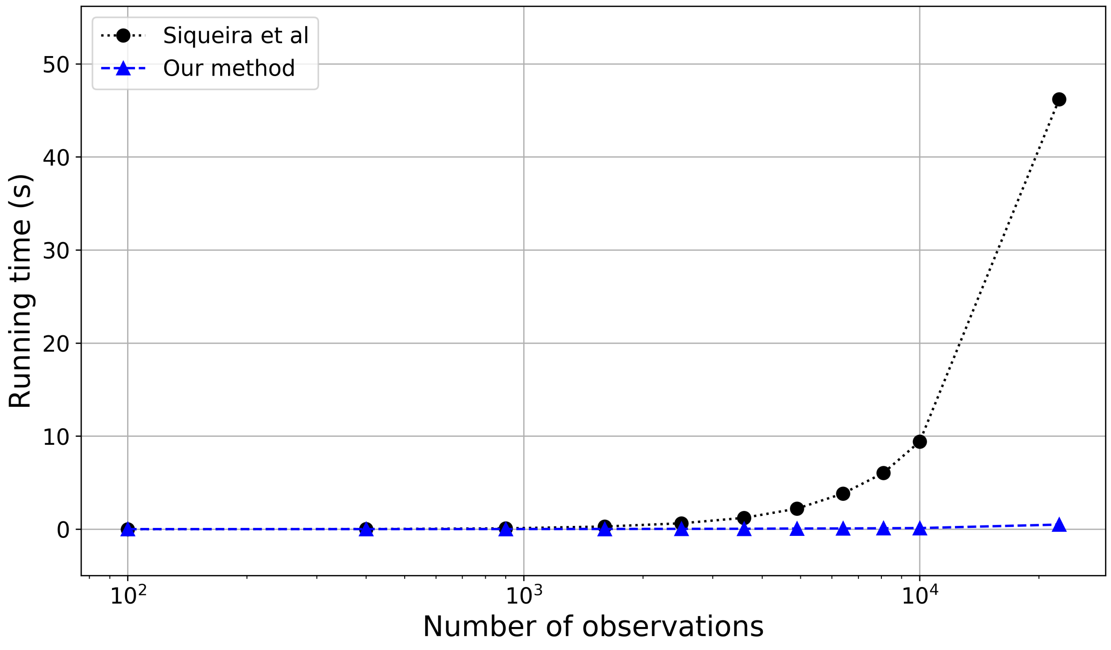

Figure 3.7: Comparison between the running time of the fast equivalent-layer technique (SIQUEIRA *et al.*, 2017) and our method. The values were obtained for $N^{it} = 50$ iterations.

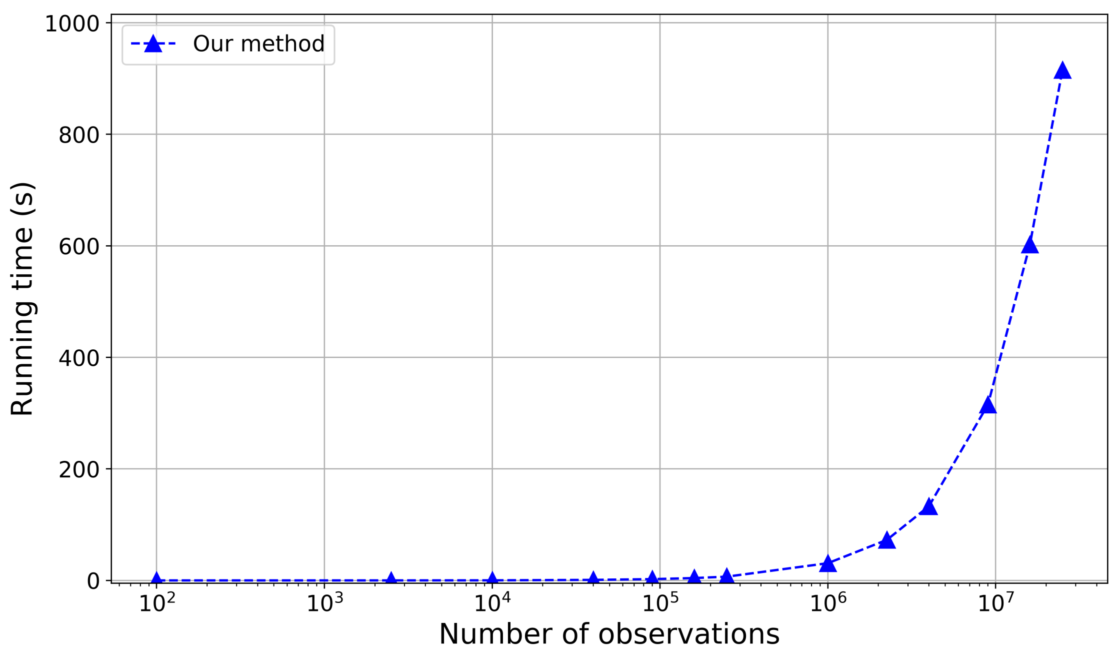

Figure 3.8: Running time of our method for a number of observations N up to 25 millions. The values were obtained for $N^{it} = 50$ iterations.

$N \times N$	**Full RAM (Mb)**	**BTTB RAM (Mb)**	**BCCB RAM (Mb)**
100×100	0.0763	0.0000763	0.0006104
400×400	1.22	0.0031	0.0248
$2\,500 \times 2\,500$	48	0.0191	0.1528
$10\,000 \times 10\,000$	763	0.00763	0.6104
$40\,000 \times 40\,000$	12 207	0.305	2.4416
$250\,000 \times 250\,000$	476 837	1.907	15.3
$500\,000 \times 500\,000$	1 907 349	3.815	30.518
$1\,000\,000 \times 1\,000\,000$	7 629 395	7.629	61.035

Table 3.1: Comparison between the system computer memory usage needed to store the full matrix, the BTTB single column of the sensitivity matrix and the BCCB eigenvalues (eight times greater than the BTTB singel column). The quantities were computed for different numbers of data (N) with the same corresponding number of equivalent sources (N). This table considers that each element of the matrix is a double-precision number, which requires 8 bytes of storage, except for the BCCB complex eigenvalues, which requires 16 bytes per element.

3.5.3 Convolutional equivalent layer for magnetic data processing

Note that the standard CGLS solution (Algorithm 1) requires neither inverse matrix nor matrix-matrix product. Instead, it only requires: one matrix-vector product out of the loop and two matrix-vector products per iteration (in steps 3 and 6). These products can be efficiently computed by using the 2D FFT, as a discrete convolution; This modified approach in which the standard CGLS method is modified to efficiently compute the matrix-vector products will be referenced throughout this work as the *convolutional equivalent layer method.*

3.5.4 Computational performance for magnetic data processing

In this section we compare the efficiency of the classical (equation 2.13), standard CGLS (Algorithm 1) and the convolutional equivalent layer method (Algorithm 1 with matrix-vector products computed according to 3.4.5). To do this, we compute the total number of *flops* associated to them (GOLUB and LOAN, 2013, p. 12).

For the classical method, we have $\frac{1}{2}N^3$ flops to compute the lower triangle of $\mathbf{A_m}^\top \mathbf{A_m}$; $\frac{1}{3}N^3$ flops to compute the Cholesky factor $\mathbf{G}$ of $\mathbf{A_m}^\top \mathbf{A_m}$ (GOLUB and LOAN, 2013, p. 164); $2\,N^2$ flops to compute the matrix-vector product $\mathbf{A_m}^\top \mathbf{d}^o$; and $2\,N^2$ flops to solve the triangular systems given by equation 2.13 (GOLUB and

LOAN, 2013, p. 106). The resultant flop count for the classical method is

$$f_{classical} = \frac{5}{6}N^3 + 4\,N^2\,. \tag{3.119}$$

For the standard CGLS method (Algorithm 1) we have $2\,N^2$ to compute the matrix-vector product $\mathbf{A_m}^\top \mathbf{s}_{(it)}$ out of the loop; $4\,N$ in step 1; $2\,N$ in step 2; $2\,N^2 + 2\,N$ in step 3; $2\,N$ in step 4; $2\,N$ in step 5; and $2\,N^2$ in step 6. The resultant flop count is given by:

$$f_{cgls} = 2N^2 + it\,(4N^2 + 12N)\,. \tag{3.120}$$

To compute the flops count of our method, we need only to replace the flops associated with matrix-vector products in the standard CGLS method by those associated with 2D convolution defined in section 3.4.2, which consists of $\kappa\,4N\log_2(4N)$ flops to compute the 2D FFT for each matrix $\mathbf{L}_{\boldsymbol{\alpha\beta}}$ (equation 3.95); $\kappa\,4N\log_2(4N)$ flops to compute $\mathbf{F}_{2Q}\,\mathbf{V}\,\mathbf{F}_{2P}$ via 2D FFT; $24\,N$ flops to compute the Hadamard product $\mathbf{L} \circ (\mathbf{F}_{2Q}\,\mathbf{V}\,\mathbf{F}_{2P})$; and $\kappa\,4N\log_2(4N)$ flops to compute the IDFT (inverse discrete Fourier transform) in equation 3.97. We use $\kappa = 5$ for the *radix-2* algorithm (VAN LOAN, 1992, p. 15). By replacing these flops into Algorithm 1, we obtain the complete number of flops

$$f_{conv} = \kappa\,16N\log_2(4N) + 24N + it\,(\kappa\,16N\log_2(4N) + 60N)\,. \tag{3.121}$$

Figure 3.9 shows a comparison between $f_{classical}$ (equation 3.119), f_{cgls} (equation 3.120) and f_{conv} (equation 3.121) for different numbers of observation points up to $1,000,000$. As we can see, the total flops count associated with our method is 7 orders of magnitude smaller than that associated with the classical method and 3 orders of magnitude smaller than that associated with the standard CGLS method by using a maximum number of iterations $N^{it} = 50$.

Figure 3.10 shows the time necessary to build matrix $\mathbf{A_m}$ (equation 3.20) and solve the linear system for N varying up to $10,000$. With $N = 10,000$, the classical method takes more than sixty-three seconds, the standard CGLS more than twelve seconds, while our method takes only half a second. The CPU used for this test was a Intel Core i7-7700HQ@2.8GHz.

Table 3.2 shows a comparison between the computer memory storage associated with each method. The classical and standard CGLS methods have to store the whole matrix $\mathbf{A_m}$ (equation 3.20). For example, a dataset with $N = 10,000$ observation points has an associated sensitivity matrix $\mathbf{A_m}$ formed by $N^2 = 100,000,000$ elements and takes approximately 763 Megabytes of memory (8 bytes per element). Using the same number of observation points $N = 10,000$, our method requires only 1.831 Megabytes to store the first columns of the BCCB matrices $\mathbf{C}_{\boldsymbol{\alpha\beta}}$ (equation

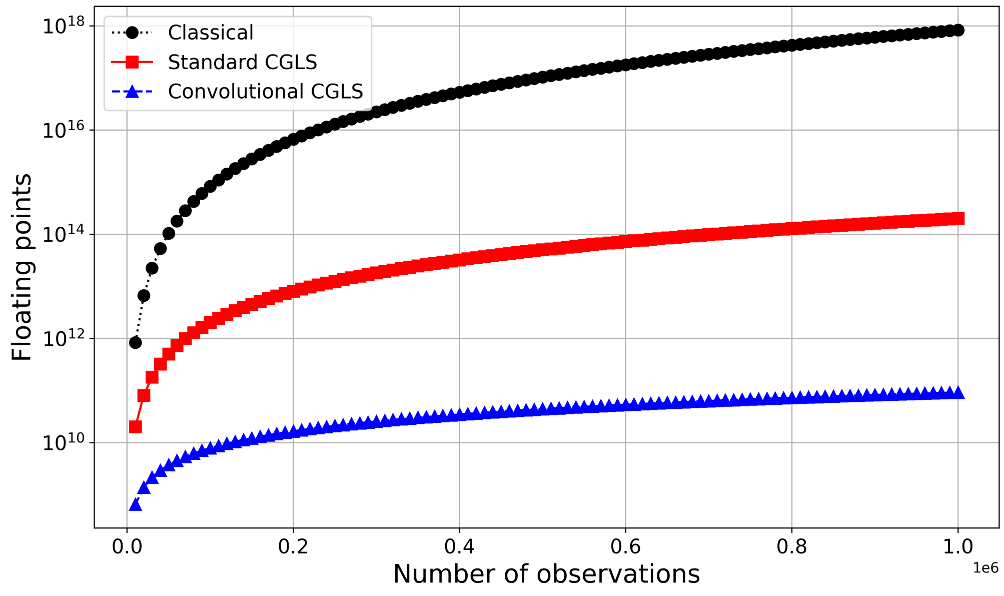

Figure 3.9: Number of flops associated with classical method (equation 3.119), the standard CGLS method (equation 3.120) and our method (equation 3.121), all of them with $N^{it} = 50$. The number of observation points N varies from $5,000$ to $1,000,000$.

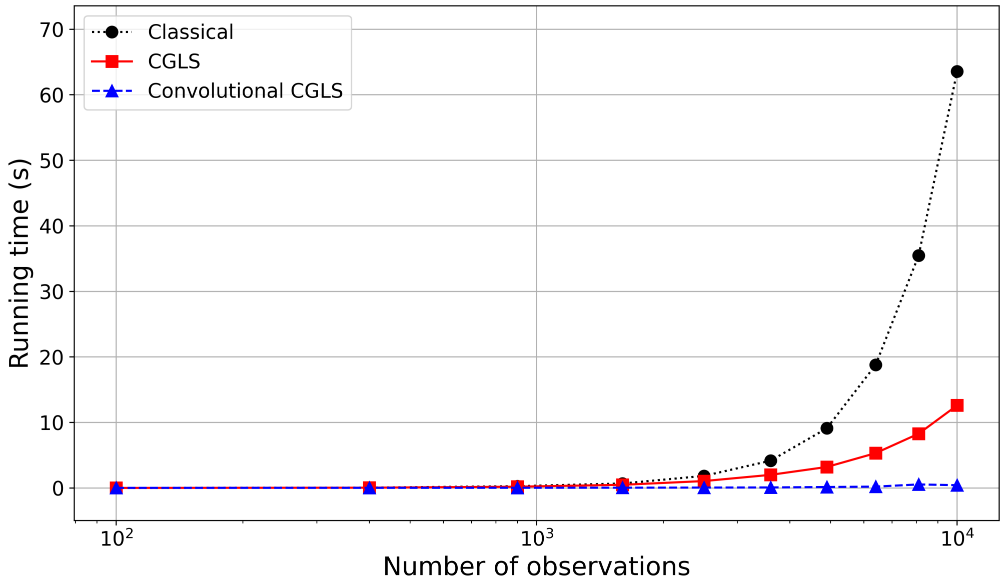

Figure 3.10: Comparison between the runtime of the equivalent-layer technique using the classical method, standard CGLS method and our method. The values for the standard CGLS and our method use $N^{it} = 50$ iterations.

3.87) and 0.6104 Megabytes to store the complex matrix $\mathbf{L}$ (equation 3.99) (16 bytes per element). For a bigger dataset with $N = 1,000,000$, the amount of necessary computer memory goes to $7,629,395$, 183.096 and 61.035 Megabytes, respectively.

N	**Matrix $\mathbf{A_m}$**	**All six first columns of BCCB matrices**	**Matrix L**
100	0.0763	0.0183	0.00610
400	1.22	0.0744	0.0248
$2,500$	48	0.458	0.1528
$10,000$	763	1.831	0.6104
$40,000$	12,207	7.32	2.4416
$250,000$	476,837	45.768	15.3
$500,000$	1,907,349	91.56	30.518
$1,000,000$	7,629,395	183.096	61.035

Table 3.2: This table shows the computer memory usage (in Megabytes) for storing the whole $N \times N$ matrix $\mathbf{A_m}$ (equation 3.20), the first columns of the BCCB matrices $\mathbf{C_{\alpha\beta}}$ (equation 3.87) (both need 8 bytes per element) and the matrix $\mathbf{L}$ (equation 3.99) (16 bytes per element).

Chapter 4

Application to synthetic data

4.1 Synthetic data applied to gravimetric processing

We have simulated three sources whose horizontal projections are shown in Figure 4.1 as black lines. These sources are a sphere with density contrast $-1.25\,\mathrm{g/cm^3}$ and two rectangular prisms with density contrasts $1.00\,\mathrm{g/cm^3}$ (upper-left prism) and $1.30\,\mathrm{g/cm^3}$ (upper-right prism). Figure 4.1 shows the gravity disturbance (vertical component of gravitational attraction) produced by these sources. The synthetic data are contaminated with additive pseudorandom Gaussian noise with zero mean and standard deviation of 0.1 mGal. The data are computed at $N = 10,000$ observation points that are regularly spaced on a 100×100 grid, at $z_1 = -100$ m. We have set a grid of equivalent sources, each one directly beneath each observation point, at $z_0 = 300$ m.

Figure 4.2a and 4.2b show the data fits obtained, respectively, by the fast equivalent-layer technique (SIQUEIRA *et al.*, 2017) and by our method. They represent the differences between the simulated data (Figure 4.1) and the predicted data produced by both methods (not shown) after $N^{it} = 40$ iterations. As we can see, both methods produce virtually the same results. This excellent agreement is confirmed by Figure 4.2c, which shows the differences between the predicted data produced by both methods.

4.1.1 Gravimetric data processing

We performed the upward- and downward-continuations of the simulated gravity data (Figure 4.1) by using the fast equivalent-layer technique (SIQUEIRA *et al.*, 2017), our method and also the classical approach in the Fourier domain. This approach consists in performing the upward- or downward-continuation by directly

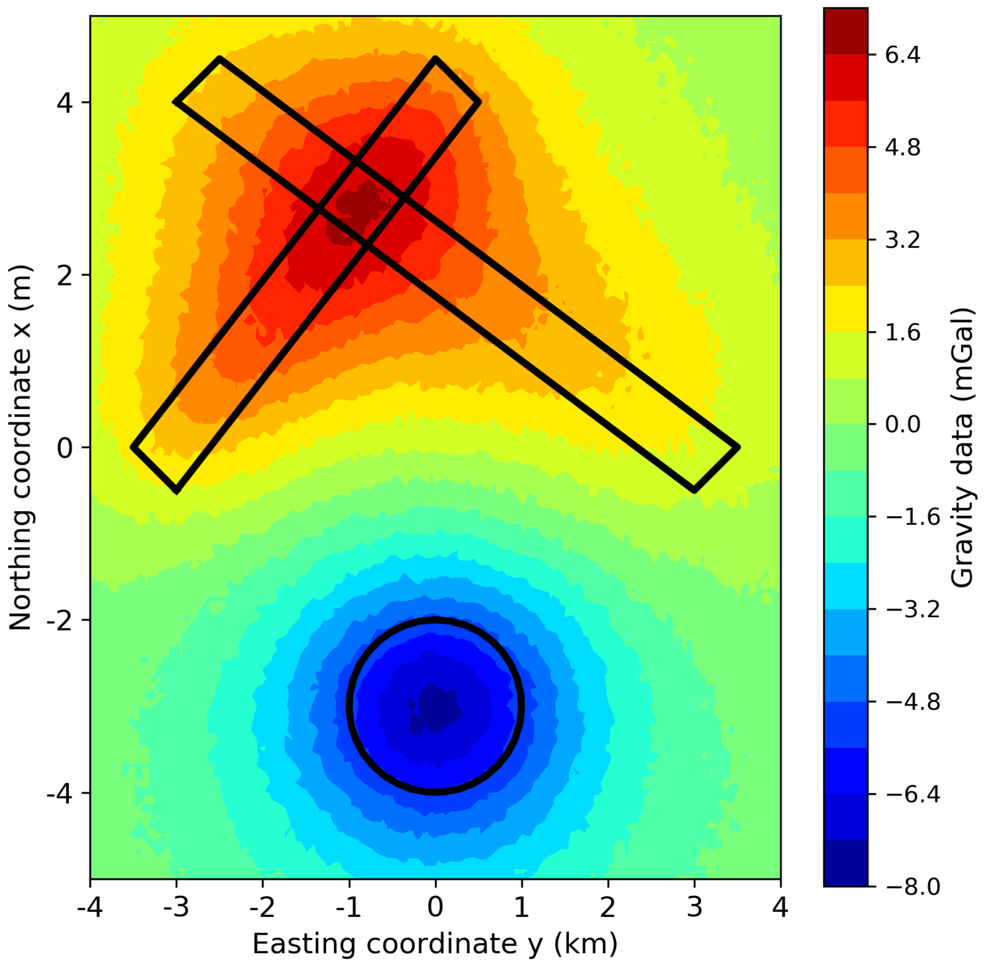

Figure 4.1: Application to synthetic data. Noise-corrupted gravity data (in color map) produced by three synthetic sources: a sphere with density contrast $-1.25\,\mathrm{g/cm^3}$ and two rectangular prisms with density contrasts $1.00\,\mathrm{g/cm^3}$ (upper-left body) and $1.30\,\mathrm{g/cm^3}$ (upper-right body). The black lines represent the horizontal projection of the sources.

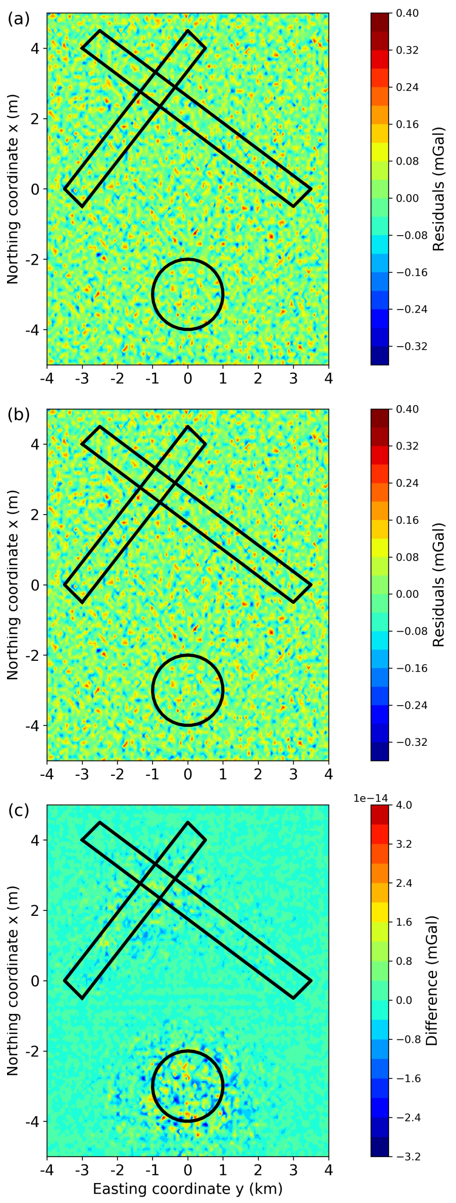

Figure 4.2: Application to synthetic data. Residuals between the simulated data (Figure 4.1) and predicted data produced by: (a) the fast equivalent-layer technique (SIQUEIRA *et al.*, 2017) and (b) our method. The mean (-4.493×10^{-5} mGal) and standard deviation (0.093 mGal) for residuals shown in a and b are exactly the same. (c) Difference between a and b. The computation times spent by the fast equivalent-layer technique and our method were 10.416 and 0.177 seconds, respectively.

computing the Fourier transform of the gravity data (e.g., BLAKELY, 1996, p. 317). Figure 4.3 shows the upward-continued gravity data obtained by the three methods. As we can see, the residuals between the true data at $z = -300$ m (Figure 4.3a) and the upward-continued data obtained by using our method (Figure 4.3b) and the fast equivalent-layer technique (Figure 4.3c) are very similar to each other. Notice that the absolute values of the residuals produced by the classical Fourier approach (Figure 4.3d) are ≈ 10 times greater than those produced by our method and the fast equivalent-layer technique (Figure 4.3b and 4.3c), with maximum values concentrated at the border of the simulated area. Differently from the results yield by our method (Figure 4.3b) and the fast equivalent-layer technique (Figure 4.3c), that obtained with the classical Fourier approach exhibits a slight noise amplification (Figure 4.3d).

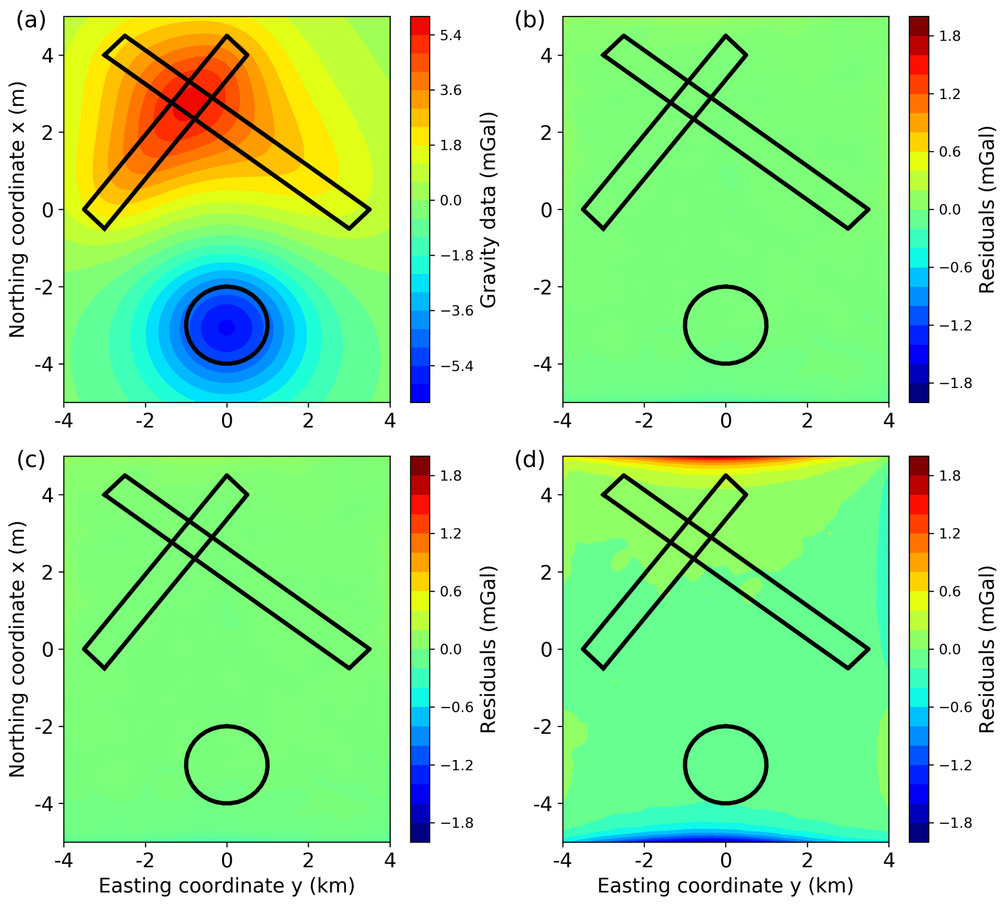

Figure 4.3: Application to synthetic data. (a) Noise-free gravity data produced by the synthetic sources at $z = -300$ m. Residuals between the data shown in a and the upward-continued data obtained by: (b) our method (not shown), with mean 0.003 mGal and standard deviation 0.034 mGal, (c) the fast equivalent-layer technique (not shown), with mean 0.003 mGal and standard deviation 0.034 mGal and (d) the classical Fourier approach (not shown), with mean -0.030 mGal and standard deviation 0.262 mGal. The computation times spent by the fast equivalent-layer technique and our method were 8.697 and 0.005 seconds, respectively.

Figure 4.4 shows the results obtained by using all methods to compute downward-continuation of gravity data. In this case, the maximum absolute values of the residuals produced by the classical Fourier approach (Figure 4.4d) are ≈ 20 times greater than those produced by our method and the fast equivalent-layer technique (Figure 4.4b and 4.4c). This noise amplification is a well-known problem of the downward-continuation produced by the classical Fourier approach (e.g., BLAKELY, 1996, p. 320).

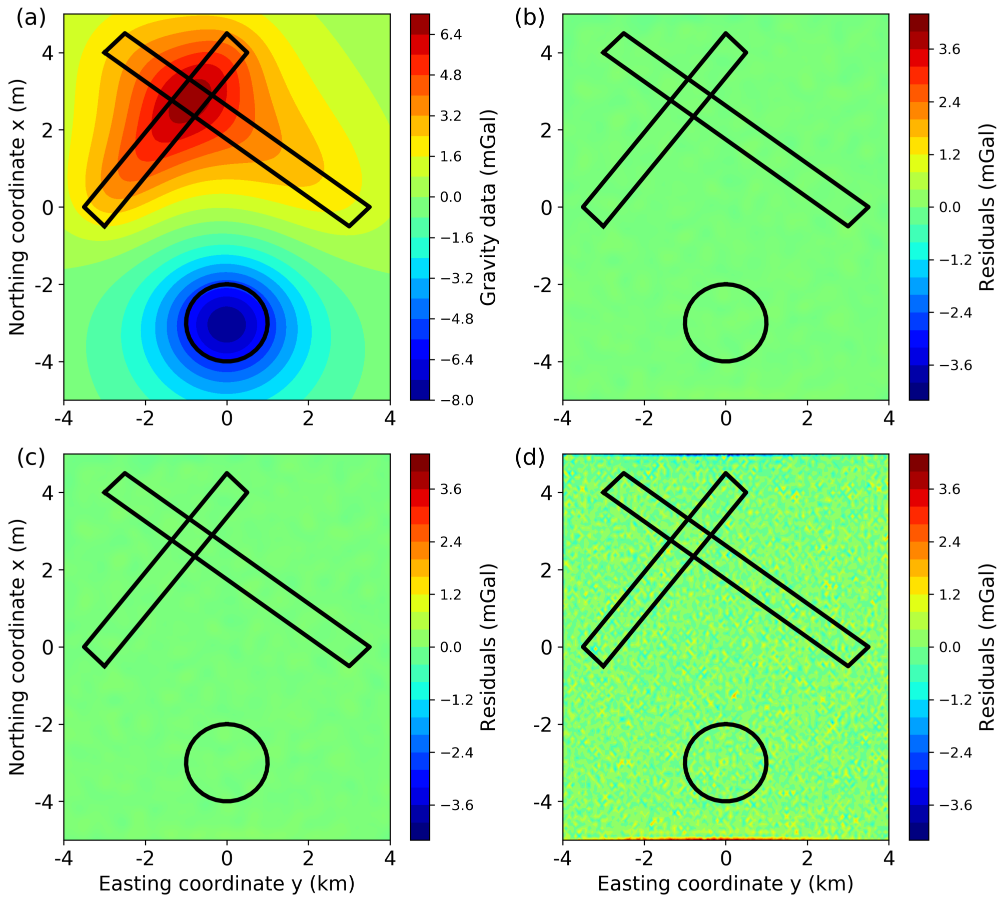

Figure 4.4: Application to synthetic data. (a) Noise-free gravity data produced by the synthetic sources at $z = -50$ m. Residuals between the data shown in a and the downward-continued data obtained by: (b) our method (not shown), with mean -0.001 mGal and standard deviation 0.038 mGal, (c) the fast equivalent-layer technique (not shown), with mean -0.001 mGal and standard deviation 0.038 mGal and (d) the classical Fourier approach (not shown), with mean -0.030 mGal and standard deviation 0.262 mGal. The computation times spent by the fast equivalent-layer technique and our method were 8.795 and 0.004 seconds, respectively.

We opted for showing all the results (Figures 4.3 and 4.4) produced by all methods without removing the border effects in order to properly compare them to each other. We also stress that no padding function to expand the data was used in applying our method, the fast equivalent-layer technique or the Fourier approach.

Another important aspect to be pointed out about these results is the computational times spent by our method and the fast equivalent-layer technique. The total computational time required by our method to estimate the physical-property distribution on the equivalent layer and to perform the upward- or downward-continuations is about two orders of magnitude lower than that spent by the fast equivalent-layer technique. This significant reduction in computational time was obtained by using a data set composed of $N = 10,000$ observation points. Considerably better results can be obtained with larger data sets.

Finally, we did not compare the total computational times spent by our method and the classical Fourier approach, but we can affirm that the second is smaller. Because the Fourier approach requires only one DFT/IDFT of the data, whereas our method requires one DFT/IDFT per iteration, it is computationally faster than our method. However, the considerably smaller noise amplification and practically nonexistent border effect are the main advantages of our method over the classical Fourier approach, especially in the downward-continuation.

4.2 Synthetic data applied to magnetic processing

Our convolutional equivalent layer method requires a regular data grid located on a horizontal and flat observation surface. Here, we evaluate the performance of our method by applying it to simulated airborne magnetic surveys formed by i) a regular data grid on a flat surface; ii) irregular data grids on a flat surface; and iii) regular data grid on undulating surfaces. Note that the simulated surveys in (ii) and (iii) violate the premises of our method.

4.2.1 Simulated airborne surveys

The first and second rows in Figure 4.5 show, respectively, the simulated flight patterns and noise-corrupted total-field anomalies of the airborne magnetic surveys used in our tests. The third row in Figure 4.5 shows the true upward-continued total-field anomalies at $z = -1,300$ m. The fourth row in Figure 4.5 shows the true reduced to pole total-field anomalies. All magnetic data (second and lower rows in Figure 4.5) are produced by the same three synthetic bodies: two prisms and one sphere with constant total-magnetization vector having inclination, declination and intensity of 35.26°, 45°, and 3.4641 A/m, respectively. The simulated main geomagnetic field has inclination and declination of 35.26° and 45°, respectively.

Figure 4.5a shows the simulated airborne survey on a regular grid of 100×50 observation points (totaling $N = 5,000$ observation points), with a grid spacing of $\Delta x = 101.01$ m and $\Delta y = 163.265$ m along the x- and y-axis, respectively.

The noise-corrupted total-field anomaly (second panel of Figure 4.5a) is calculated at $z = -900$ m, with pseudorandom Gaussian noise added having null mean and standard deviation of 0.2961 nT.

Figures 4.5b and 4.5c show the simulated surveys on irregular grids obtained by perturbing the horizontal coordinates of the regular grid (upper panel in Figure 4.5a). For the survey shown in Figure 4.5b, the x and y coordinates are perturbed with sequences of pseudorandom Gaussian noises having null mean and standard deviations equal to 20% of the corresponding grid spacing, which results in absolute values of 20.2 m and 32.6 m, along the x- and y-directions, respectively. For the survey shown in Figure 4.5c, the standard deviations are equal to 30% of the corresponding grid spacing, which results in absolute values of 30.3 m and 49.0 m along the x- and y-directions, respectively. Their noise-corrupted total-field anomalies (second panels in Figures 4.5b and 4.5c) are calculated on their corresponding irregular grids, on a flat observation surface at $z = -900$ m, with pseudorandom Gaussian noise added having null mean and standard deviation of 0.2961 nT.

Figures 4.5d and 4.5e show the simulated surveys on the same regular grid as shown in Figure 4.5a (upper panel). The difference is that observation points are located no longer on a flat, but on undulating surfaces. For the survey shown in Figure 4.5d, the z coordinates of the undulating surface are defined by a sequence of pseudorandom Gaussian noise having mean -900 m and standard deviation equal to 5% of 900 m, which corresponds to 45 m. For the survey shown in Figure 4.5e, the standard deviation is equal to 10% of 900 m, which corresponds to 90 m. The noise-corrupted total-field anomalies of these simulated surveys (second panels in Figures 4.5d and 4.5e) are calculated on their corresponding undulating surfaces (upper panels in Figures 4.5d and 4.5e), on the same regular grid shown in Figure 4.5a, with pseudorandom Gaussian noise added having null mean and standard deviation of 0.2961 nT.

4.2.2 Tests with a regular data grid on a flat surface

Figure 4.6 show the difference between the simulated (second row in Figure 4.5) and predicted data (not shown) obtained by using the classical (the upper row) and our method (the second row). From now on, we designate this difference as data residuals. The lower row in Figure 4.6 shows the convergence curve of our method.

The data residuals using the classical method (equation 2.13) are shown in the upper panel of Figure 4.6a, with mean 0.4118 nT and standard deviation 0.3780 nT. This process took 17.10 seconds. Using our method, the data residuals (the middle panel in Figure 4.6a) have mean 0.9972 nT and standard deviation 1.3904 nT. In this case, however, the processing time was only 0.25 seconds. As expected,

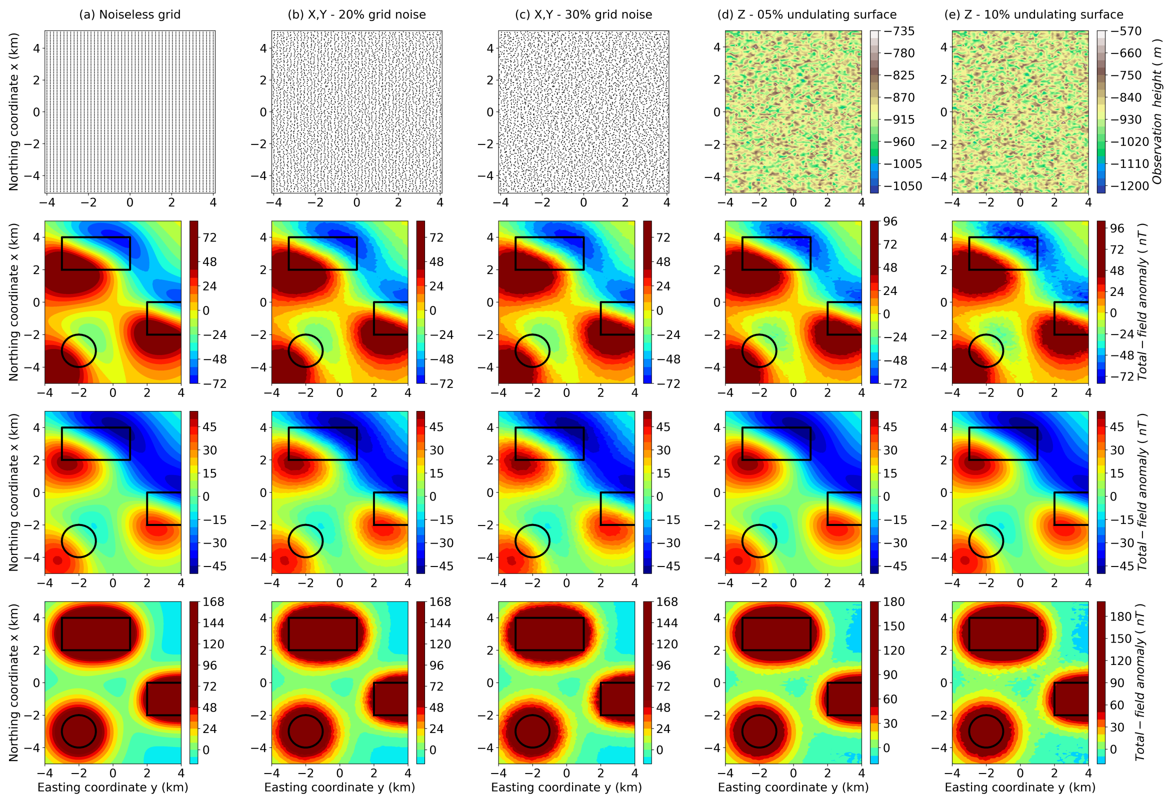

Figure 4.5: Synthetic tests: the simulated airborne magnetic surveys - The first row shows the grids of observation points and the undulating observation surfaces that simulate the airborne magnetic surveys. The second row shows the noise-corrupted total-field anomalies produced by the synthetic sources and calculated on the simulated airborne magnetic survey shown in the first row. The third row shows the noise-free total-field anomalies produced by the synthetic sources at $z = -1,300$ m (the true upward-continued total-field anomalies). The fourth row shows the noise-free total-field anomalies produced by the synthetic sources at inclination $I_0 = 90\circ$ (the true reduced to pole total-field anomalies). The results shown in these last three rows were obtained by using the simulated airborne magnetic surveys as follows: (a) A regular grid of 100×50 observation points in the $x-$ and $y-$directions and a flat observation surface at $z = -900$ m. An irregular grid with uncertainties of (b) 20% and (c) 30% in the $x-$ and $y-$coordinates and a flat observation surface at $z = -900$ m . A regular grid of 100×50 observation points in the $x-$ and $y-$coordinates and an undulating observation surface with uncertainties of (d) 5% and (e) 10%. The black lines represent the horizontal projection of the sources .

the Euclidean norm of the data residuals produced by our method (lower panel in Figure 4.6a) decreases. The convergence criterion was satisfied close to iteration 50.

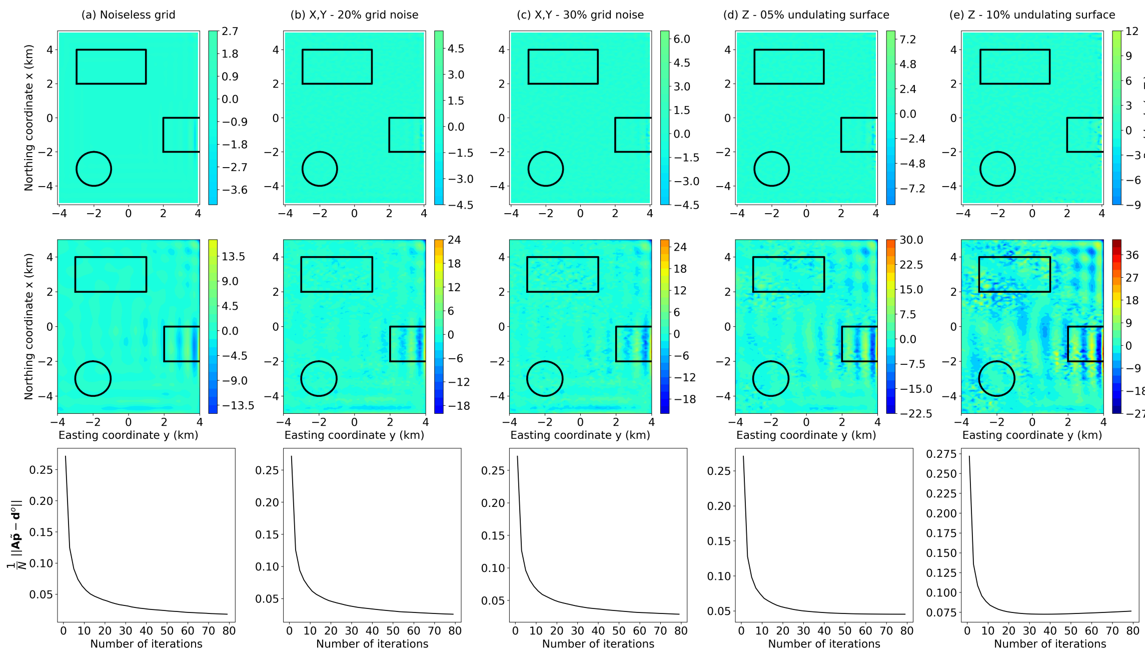

Figure 4.6: Synthetic tests: the data residuals and convergence - The first row shows the data residuals using the classical method. The second and third rows show, respectively, the data residuals and the convergence curves using the convolutional equivalent layer (our method). The results shown in these three rows were obtained by using the simulated airborne magnetic surveys shown in Figure 4.5, i.e.: (a) A regular grid of 100 × 50 observation points in the $x-$ and $y-$directions and a flat observation surface at $z = -900$ m. An irregular grid with uncertainties of (b) 20% and (c) 30% in the $x-$ and $y-$coordinates and a flat observation surface at $z = -900$ m . A regular grid of 100 × 50 observation points in the $x-$ and $y-$coordinates and an undulating observation surface with uncertainties of (d) 5% and (e) 10%. The black lines represent the horizontal projection of the sources .

4.2.3 Tests with irregular data grids on a flat surface

Figure 4.6b shows the results obtained with the irregular data grid perturbed by using 20% of the regular grid spacing. In this Figure we can see that the data residuals using the classical method (upper panel) yield a good data fit with mean 0.4084 nT and standard deviation 0.3862 nT. Using our method, the data residuals (middle panel in Figure 4.6b) also produced an acceptable data fitting with mean of 1.3125 nT and standard deviation of 1.7187 nT. The Euclidean norm of the data residuals obtained by our method (lower panel in Figure 4.6b) decreases, as expected, and converges to a constant value close to iteration 50.

Figure 4.6c shows the results obtained with the irregular data grid perturbed by using 30% of the regular grid spacing. The data residuals obtained by the classical method (upper panel in Figure 4.6c) produced an acceptable data fit, having mean 0.4070 nT and standard deviation 0.3899 nT. Using our method, the data residuals (middle panel in Figure 4.6c) with mean 1.5129 nT and standard deviation 1.8526 nT also produced a good data fitting. The convergence of our method (lower panel in Figure 4.6c) shows that, similarly to the previous results, the Euclidean norm of the residuals decreases; converging to a constant value close to iteration 50. Note that this good result was obtained by using a very perturbed data grid (upper panel in Figure 4.5c).

4.2.4 Tests with regular data grid and undulating surfaces

Figure 4.6d shows the results obtained with data on the undulating surface varying 5% of $z = 900$ m. In this case, the data residuals either using the classical method (upper panel in Figure 4.6d) or our method (middle panel in Figure 4.6d) reveal acceptable data fittings. Using the classical method, data residuals have mean 0.4316 nT and standard deviation 0.4762 nT. Using our method, they have mean 2.1069 nT and standard deviation 2.5023 nT. Likewise, the Euclidean norm of the data residuals produced by our method (lower panel in Figure 4.6d) decreases up to iteration 50 and reaches the convergence criterion in the subsequent iterations (mean residulas are less than 0.00015 between iterations).

Figure 4.6e shows the results obtained with data on the undulating surface varying 10% of $z = 900$ m. By using the classical approach, the data residuals (upper panel in Figure 4.6e) yielded a good data fitting, with mean 0.4818 nT and standard deviation 0.6565 nT. By using our method, the data residuals (middle panel in Figure 4.6e) yielded a worse data fitting with mean 3.4981 nT and standard deviation 3.8153 nT. The convergence curve (lower panel in Figure 4.6e) reveals the inadequacy of our method in dealing with observations on rugged surfaces, as the Euclidean norm of the data residuals do not decrease as much as in previous tests. We stress that, in this test, the undulating surface (upper panel in Figure 4.5e) varies in a broad range of flight values, from $z = -570$ m to about $z = -1,230$ m. Thus, this simulated airborne magnetic survey greatly violates the requirement of a flat observation surface demanded by our method.

Although our method is formulated to deal with magnetic observations measured on a horizontally regular grid, on a flat surface, the results obtained with synthetic data show that our method is robust in dealing either with irregular grids in the horizontal directions or with uneven surfaces. However, the robustness of our method has limitations. High discrepancies in the x-, y, and z-coordinates lead to unacceptable data fittings (large data residuals), as shown the middle panels in Figures 4.6c and 4.6e.

4.2.5 Magnetic data processing

We performed the upward-continuations of the synthetic total-field anomalies (second row in Figure 4.5) by using the classical method, our convolutional equivalent layer method, and the classical approach in the Fourier domain, which consists in computing the Fourier transform of the total-field anomaly (e.g., BLAKELY, 1996, p. 317).

Figure 4.7 shows the continuation residuals defined as the differences between the true upward-continued total-field anomalies (third row in Figure 4.5) and the predicted upward-continued total-field anomalies (not shown). We conveniently denote these differences as continuation residuals. The continuation residuals obtained by using the classical method (upper row) and our method (middle row) are similar to each other in most of the tests. The exceptions are the synthetic test with data over irregular grid (Figures 4.5c and 4.6c) and over an undulating surface (Figures 4.5e and 4.6e), which greatly violates the requirement of regular grids or a flat observation surface, demanded by our method. Note that the maximum absolute value of the continuation residuals produced by using our method (middle panel in Figure 4.7e) are ≈ 2 times greater than those produced by the classical method (upper panel in Figure 4.7e).

In contrast, the continuation residuals obtained by using the classical Fourier approach (lower row in Figure 4.7) are, in most of the tests, approximately 2 times greater than those produced by the classical method (upper row in Figure 4.7) and 1.5 times greater than those produced by our method (middle row in Figure 4.7). Note that, similar to our method, the maximum absolute values of the continuation residuals obtained by using the classical Fourier approach are located at the boundaries of the simulated area. However, the values are significantly higher.

Figure 4.8 shows the differences between the true reduced to pole total-field anomalies (fourth row in Figure 4.5) and the predicted reduced to pole total-field anomalies (not shown). The true reduced to pole total-field anomalies are generated by using only induced magnetization, with $I_0 = 90°$ and $D_0 = 0°$. Figure 4.8 shows that the reduced to pole residuals obtained by using the classical method (upper row) and our method (middle row) have differences when high irregular grids or non flat surfaces are used (Figures 4.8c and 4.8e). The absolute values of the reduced to pole residuals are almost ≈ 2 times greater than those of classical method when the 10% standard deviation was used (upper and middle panels in Figure 4.8e, respectively). As in the the continuation test, they are generally concentrated at the boundaries of the study area.

The reduced to pole residuals obtained by using the classical Fourier approach (lower row in Figure 4.8) are approximately 3.5 times greater than those produced

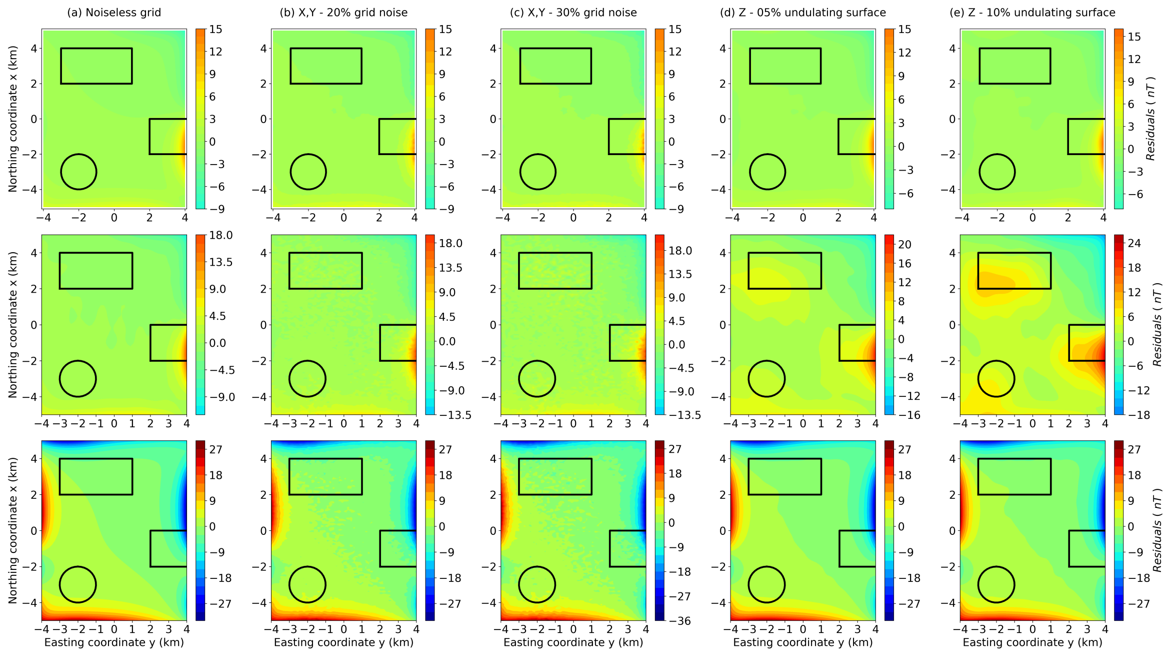

Figure 4.7: Synthetic tests: the data residuals of the upward-continued total-field anomalies (third row in Figure 4.5). The data residuals of the upward-continued total-field anomalies are defined as the difference between the noise-free total-field anomaly produced by the synthetic sources at $z = -1,300$ m (third row in Figure 4.5) and the predicted total-field anomaly at $z = -1,300$ m obtained by using three methods: the classical method (first row); the convolutional equivalent layer (second row); and the classic approach in the Fourier domain (third row). The results shown in these three rows were obtained by using the simulated airborne magnetic surveys shown in Figure 4.5, i.e.: (a) A regular grid of 100×50 observation points in the $x-$ and $y-$directions and a flat observation surface at $z = -900$ m. An irregular grid with uncertainties of (b) 20% and (c) 30% in the $x-$ and $y-$coordinates and a flat observation surface at $z = -900$ m . A regular grid of 100×50 observation points in the $x-$ and $y-$coordinates and an undulating observation surface with uncertainties of (d) 5% and (e) 10%. The black lines represent the horizontal projection of the sources .

by the classical method (upper row in Figure 4.8) and 3 times greater than those produced by our method (middle row in Figure 4.7).

Important to note that the reduction to pole, either using the equivalent layer or the Fourier approach, has the requirement of a previously knowledge of the sources magnetization directions (equation 2.7) to obtain a correct source parameter estimative, otherwise, only non-phase dependent processing can be used (upward-continuation for example).

We also call attention to the following aspects: In applying the classical method, our method, or the classical Fourier approach, we do not expand the data by using a padding scheme. The data residuals (upper and middle rows in Figure 4.6), the continuation (Figure 4.7) and reduction to pole residuals (Figure 4.8) are shown without removing edge effects. The computational time required by our method is much lower than that required by the classical method and has the same order of magnitude of that required by the classical Fourier approach. However, the classical Fourier approach shows upward-continued and reduced to pole data with strong border effects if no padding scheme is applied to expand the data.

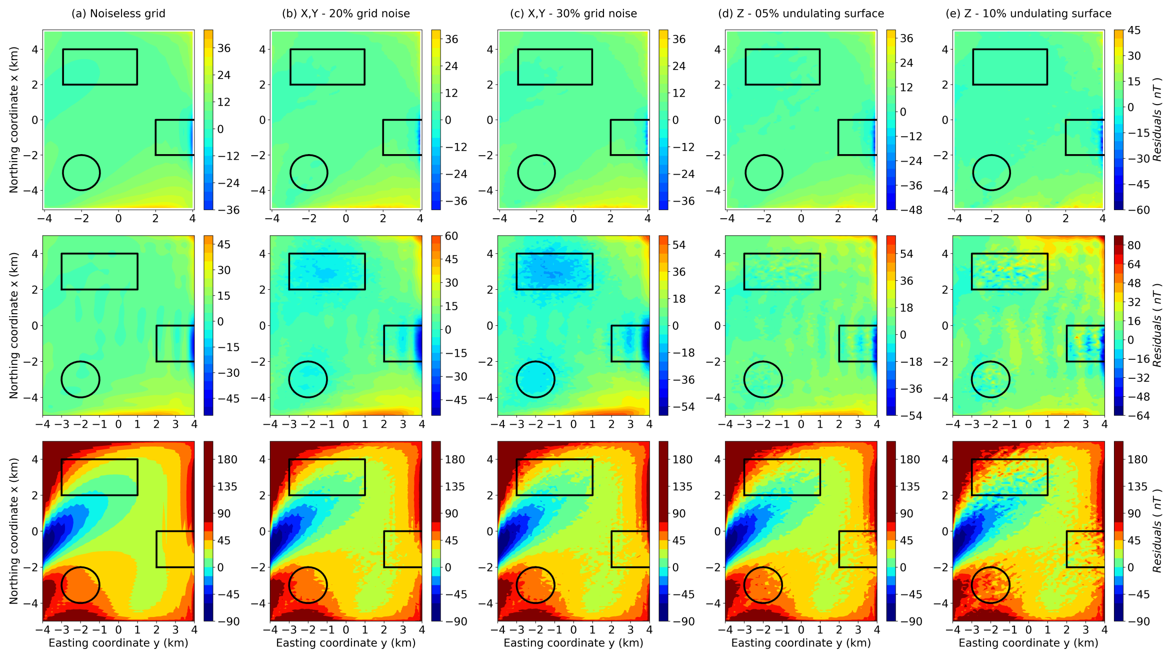

Figure 4.8: Synthetic tests: the data residuals of the reduced to pole total-field anomalies (fourth row in Figure 4.5). The data residuals of the reduced to pole total-field anomalies are defined as the difference between the noise-free total-field anomaly produced by the synthetic sources at the pole (fourth row in Figure 4.5) and the predicted total-field anomaly obtained by using three methods: the classical method (first row); the convolutional equivalent layer (second row); and the classic approach in the Fourier domain (third row). The results shown in these three rows were obtained by using the simulated airborne magnetic surveys shown in Figure 4.5, i.e.: (a) A regular grid of 100×50 observation points in the $x-$ and $y-$directions and a flat observation surface at $z = -900$ m. An irregular grid with uncertainties of (b) 20% and (c) 30% in the $x-$ and $y-$coordinates and a flat observation surface at $z = -900$ m . A regular grid of 100×50 observation points in the $x-$ and $y-$coordinates and an undulating observation surface with uncertainties of (d) 5% and (e) 10%. The black lines represent the horizontal projection of the sources .

Chapter 5

Field data results

5.1 Real field data from Carajás applied to gravimetric processing

We applied our method to airborne gravity data from Carajás, north of Brazil, which were provided by the Geological Survey of Brazil (CPRM). The data were collected along 131 north-south flight lines separated by 3 km and 29 east-west tie lines separated by 12 km. This data set was divided in two different areas, collected in different times, having samples spacing of 7.65 m and 15.21 m along the lines, totalizing $5,492,551$ observation points at a fixed height of 900 m ($z_1 = -900$ m). The gravity data were interpolated (Figure 5.1) into a regularly spaced grid of 500×500 observation points ($N = 250,000$) with a grid spacing of ≈ 717 m north-south and ≈ 782 m east-west.

To apply our method, we set an equivalent layer at $z_0 = 300$ m. Figure 5.2a shows the predicted data obtained with our method after $N^{it} = 50$ iterations. The residuals (Figure 5.2b), defined as the difference between the observed (Figure 5.1) and predicted (Figure 5.2a) data, show a very good data fit with mean close to zero (0.0003 mGal) and small standard deviation (0.1160 mGal), which corresponds to approximately 0.1 % of the maximum amplitude of the gravity data. By using the estimated mass distribution (not shown), we performed an upward-continuation of the observed gravity data to a horizontal plane located $5,000$ m above. Figure 5.3 shows a very consistent upward-continued gravity data, with a clear attenuation of the short wavelengths. By using our approach, the processing of the $250,000$ observations took only 0.216 seconds.

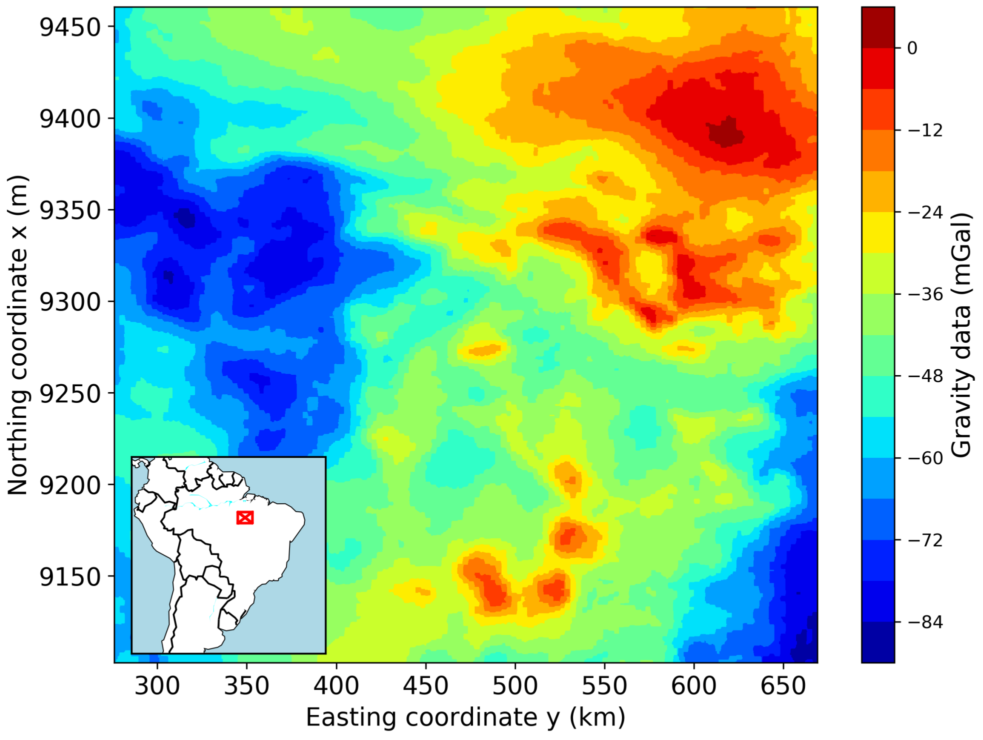

Figure 5.1: Application to field data over the Carajás Province, Brazil. Observed gravity data on a regular grid of 500×500 points, totaling $N = 250,000$ observations. The inset shows the study area (red rectangle) which covers the southeast part of the state of Pará, north of Brazil.

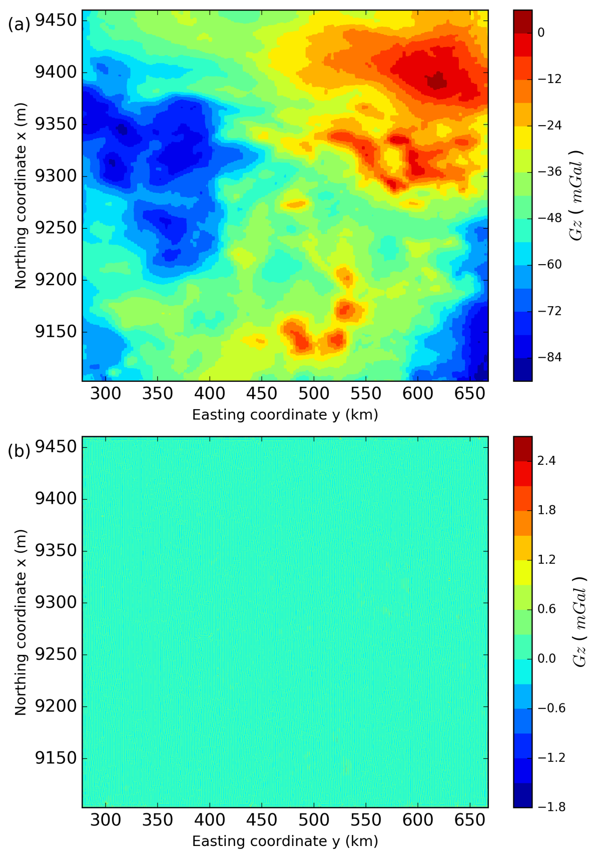

Figure 5.2: Application to field data over the Carajás Province, Brazil. (a) Predicted data produced by our method. (b) Residuals between the observed (Figure 5.1) and the predicted data (panel a), with mean 0.000292 mGal and standard deviation of 0.105 mGal.

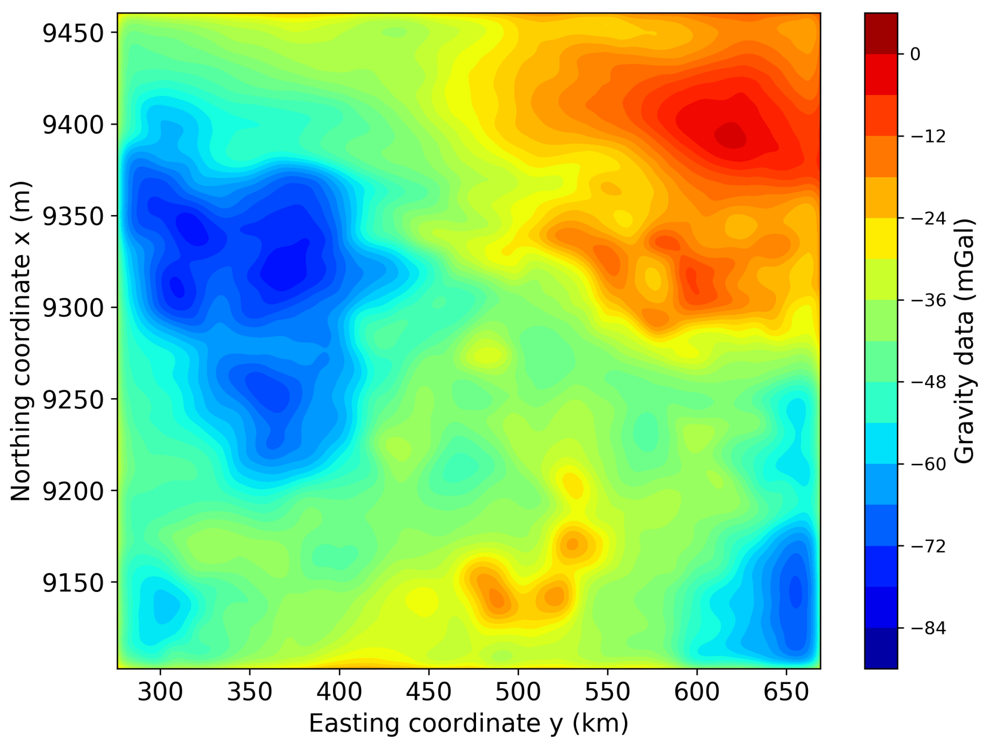

Figure 5.3: Application to field data over the Carajás Province, Brazil. The upward-continued gravity data obtained with our method 5, 000 m above the observed data (Figure 5.1). The total computation time for processing of the 250, 000 observations was 0.216 seconds.

5.2 Real field data from Carajás applied to magnetic processing

We applied the convolutional equivalent layer method to the aeromagnetic data of Carajás, northern Brazil. The survey is composed of 131 flight lines along north-south direction with line spacing of $\Delta y = 3,000$ m. Data were measured with average spacing $\Delta x = 7.65$ m along lines, with an average distance to the ground of 900 m. The total number of observation points is $N = 6,081,345$. Figure 5.4a shows the observed total-field anomaly data over the study area.

We compare the results obtained with an interpolated regular grid of $10,000 \times 131$ points, by using the nearest neighbor algorithm, and a decimated irregular grid, also with $10,000 \times 131$ points. In both cases the total $N = 1,310,000$ observation points are in the original undulating surface of the flight lines. The decimated grid was generated by choosing the nearest observation points in comparison of the regular grid presented in the interpolation. The mean and standard deviation of this irregular decimated from the regular interpolated are 6.8386 m and 107.7343 m in the x-direction and 30.8799 m and 28.3849 m in the y-direction, respectively. Both application were made with an Intel core i7 7700HQ@2.8GHz processor in single-processing and single-threading modes.

As the study area is very large, the main magnetic field varies with position. For this application, we set the main field direction as that of a mid location (latitude $-6.5°$ and longitude $-50.75°$) where the declination is $-19.86°$ and the inclination is $-7.4391°$. Both values were calculated using the magnetic field calculator from NOAA at 1st January, 2014 (epoch of the survey). We set the equivalent layer depth at 1200 meters (2100 m below the data). Figure 5.4b shows the residuals obtained after using our method to fit the interpolated data with mean 0.9089 nT and the standard deviation 3.6425 nT, revealing an acceptable data fitting. Our method took $\approx$ 390.80 seconds to converge at about 200 iterations. Figure 5.4c shows the residuals obtained after using our method to fit the decimated data with mean 0.9936 nT and standard deviation 4.0479 nT with a equally acceptable fit produced by the interpolated data. In this case, our method took $\approx$ 385.56 seconds to converge at about 200 iterations (Figure 5.4d). The convergence curve reveals a good convergence rate obtained with the decimated irregular grid. This result shows the robustness of our method in processing irregular grids. Notice that we used 200 iterations in our method of the interpolated regular grid and the mean residual still decreasing up to 2000. This happens because the invariance convergence criterion was met and the mean residuals are very small, decreasing less than 0.001 at each iteration

With $1,310,000$ observation points, it would be necessary 12.49 Terabytes of computer memory to store the full sensitivity matrix with the classical method. In this case, our method uses only 59.97 Megabytes, allowing regular desktop computers to be able to process this amount of data.

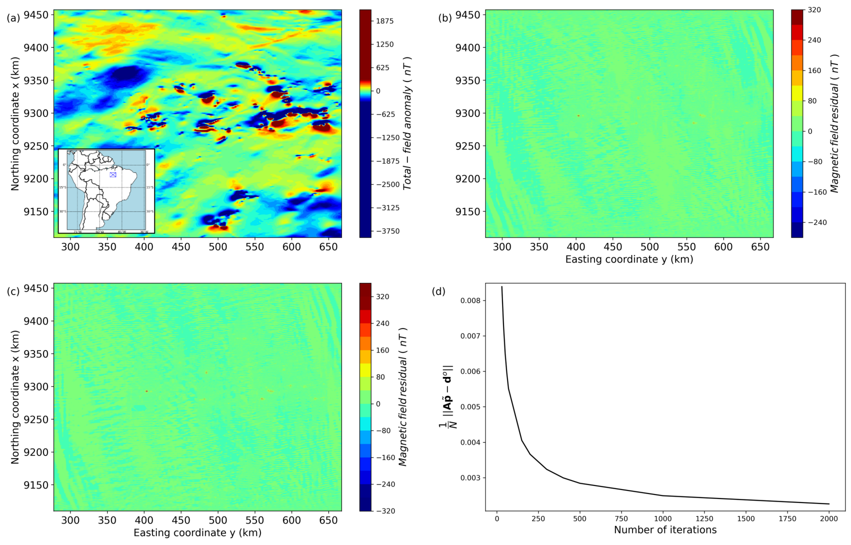

Figure 5.4: (a) Observed total-field anomaly over the Carajás Province, northen Brazil. The aeromagnetic survey was flown in 131 north-south flight lines at an average altitude of 900 m, totaling $N = 6,081,345$ observation points. (b) Data residuals, defined as the difference between the regular interpolated grid data (not shown) and the predicted data (not shown), with mean of 0.9089 nT and standard deviation of 3.6425 nT. (c) Data residuals, defined as the difference between the irregular decimated grid data (not shown) and the predicted data (not shown), with mean of 0.9936 nT and standard deviation of 4.0479 nT. (d) Convergence curve using our method to the decimated irregular grid of the real data of Carajás Province, Brazil.

Finally, Figure 5.5a shows the upward-continued magnetic data to a horizontal plane located at an altitude of 5, 000 m using the estimated equivalent layer obtained by applying our method to the decimated irregular grid. This process took ≈ 2.64 seconds, showing good results without visible errors or border effects. Figure 5.5b shows the upward-continued magnetic data to a horizontal plane located at an altitude of 5, 000 m using the classical Fourier filtering method to the decimated irregular grid. This process took ≈ 0.5 seconds. The comparison between the upward results shows a similar total-field magnetic for both cases with attenuation of the anomalies. Interestingly, the Fourier method did not present border effects to this real data. We stress that we did not use a padding scheme to expand the data.

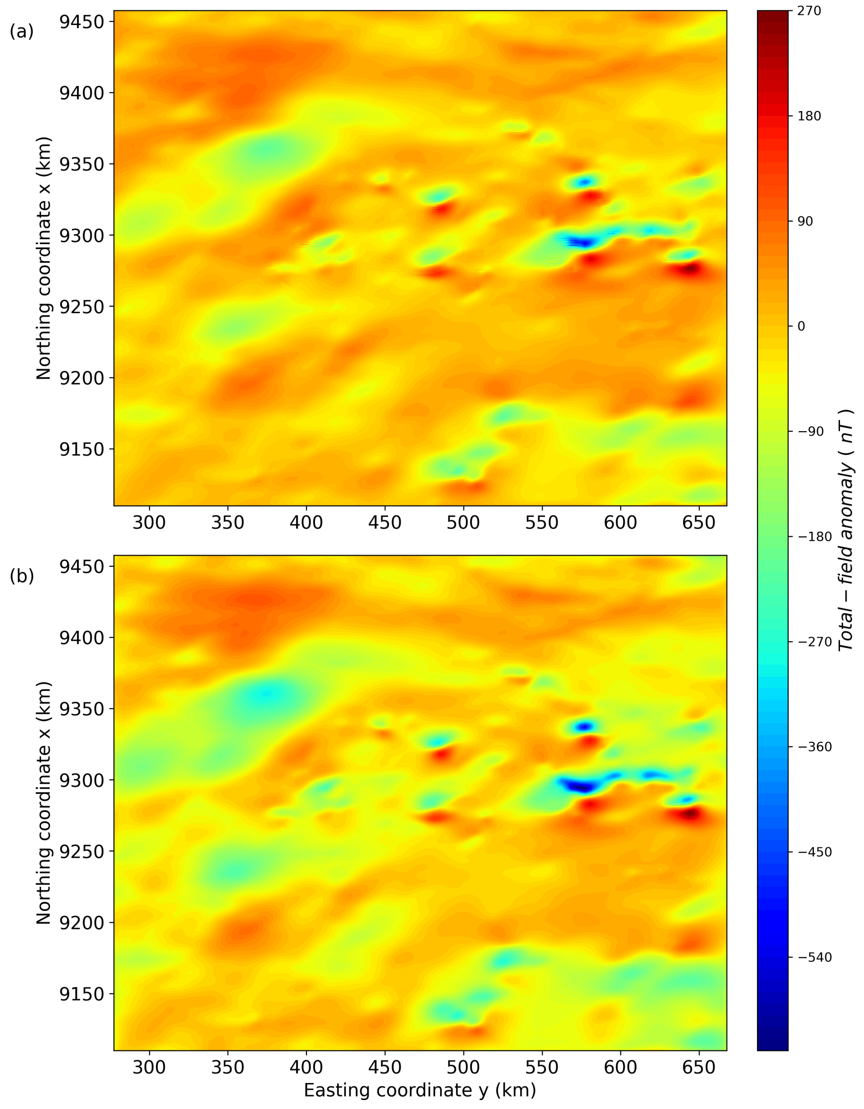

Figure 5.5: Upward-continuations of real data of Carajás Province, Brazil at altitude of 5,000 m by using: (a) the convolutional equivalent layer (our method) and (b) the classical Fourier method.

Chapter 6

Conclusions

We have proposed a fast equivalent-layer technique for processing magnetic data called convolutional equivalent layer method. We have demonstrated that the sensitivity matrix associated with planar equivalent layers of dipoles has a BTTB structure for the particular case in which the dipoles are aligned with the horizontal and regular grid of magnetic data. The product of such matrices and arbitrary vectors represents a 2D discrete convolution that can be efficiently computed via 2D Fast Fourier Transform by using only the elements forming the first column of the matrix. By using this property, we have developed a fast and memory efficient iterative method for estimating the physical-property distribution on the equivalent layer.

Comparisons between the estimated physical-property distribution obtained with our method and the classical approach that solves the least-squares normal equations via Cholesky decomposition show similar results. The differences in total number of floating-point operations (flops), memory usage and computation time, however, are noticeable. For a mid-size grid of 100×50 points, the total number of flops is about four orders of magnitude smaller than that required by the classical method. Besides, our method uses less than 1% of the computer memory and takes about 3% of the computation time associated with the classical method in this case. Significantly better results can be obtained with larger data sets.

Tests with synthetic data show that the computational time required by our method has the same order of magnitude of that required by the classical approach in the Fourier domain to perform magnetic data processing. However, the classical Fourier approach shows considerable larger border effects if no previous padding scheme is used to expand the data. Besides, although both methods require the magnetic data be on a planar and regular grid, tests with synthetic data show the robustness of our method to deal with data either on irregular grids or on undulating observation surfaces.

While the classical equivalent-layer method would require 12.49 Terabytes of computer memory to store the full sensitivity matrix associated with the irregular grid of $1,310,000$ observation points over the Carajás Province, northern Brazil, our method requires only 59.97 Megabytes. When performed on a standard laptop computer with an Intel Core i7 7700HQ@2.8GHz processor in single-processing and single-threading modes, the total times spent by our method to estimate the physical-property distribution over the equivalent layer and to compute the upward-continuation of the $1,310,000$ magnetic observations over the Carajás province was approximately 385.56 seconds and 2.64 seconds.

Further investigation could usefully explore different preconditioning strategies to improve the convergence rate of our method. Besides, considerably more work will need to be done to generalize our convolutional equivalent layer method for dealing with irregularly spaced data sets on undulating observation surfaces.

Bibliography

ASTER, R. C., BORCHERS, B., THURBER, C. H., 2019, *Parameter estimation and inverse problems*. Elsevier.

BARNES, G., LUMLEY, J., 2011, "Processing gravity gradient data", *GEOPHYSICS*, v. 76, n. 2, pp. I33–I47. doi: 10.1190/1.3548548.

BLAKELY, R. J., 1996, *Potential Theory in Gravity and Magnetic Applications*. Cambridge University Press. ISBN: 0-521-57547-8.

BOGGS, D., DRANSFIELD, M., 2004, "Analysis of errors in gravity derived from the Falcon airborne gravity gradiometer". In: *ASEG-PESA Airborne Gravity 2004 Workshop*, pp. 135–141. Geoscience Australia Record.

BOTT, M., 1967, "Solution of the linear inverse problem in magnetic interpretation with application to oceanic magnetic anomalies", *Geophysical Journal International*, v. 13, n. 1-3, pp. 313–323.

CHAN, R. H.-F., JIN, X.-Q., 2007, *An introduction to iterative Toeplitz solvers*, v. 5. SIAM.

CORDELL, L., 1992, "A scattered equivalent-source method for interpolation and gridding of potential-field data in three dimensions", *GEOPHYSICS*, v. 57, n. 4, pp. 629–636. doi: 10.1190/1.1443275.

DAMPNEY, C. N. G., 1969, "The equivalent source technique", *GEOPHYSICS*, v. 34, n. 1, pp. 39–53. doi: 10.1190/1.1439996.

DANES, Z., 1961, "Structure calculations from gravity data and density logs", *Mining Trans., 223 C.*

DAVIS, P. J., 1979, *Circulant matrices*. John Wiley & Sons, Inc.

EMILIA, D. A., 1973, "Equivalent sources used as an analytic base for processing total magnetic field profiles", *GEOPHYSICS*, v. 38, n. 2, pp. 339–348. doi: 10.1190/1.1440344.

GOLUB, G. H., LOAN, C. F. V., 2013, *Matrix Computations (Johns Hopkins Studies in the Mathematical Sciences).* Johns Hopkins University Press. ISBN: 978-1-4214-0794-4.

GRAHAM, L., KNUTH, D. E., PATASHNIK, O., 1994, *Concrete mathematics: a foundation for computer science.* Addison-Wesley Publishing Company. ISBN: 0-201-55802-5.

GUSPÍ, F., NOVARA, I., 2009, "Reduction to the pole and transformations of scattered magnetic data using Newtonian equivalent sources", *GEOPHYSICS*, v. 74, n. 5, pp. L67–L73. doi: 10.1190/1.3170690.

HANSEN, R. O., MIYAZAKI, Y., 1984, "Continuation of potential fields between arbitrary surfaces", *GEOPHYSICS*, v. 49, n. 6, pp. 787–795. doi: 10.1190/1.1441707.

HOGUE, J. D., RENAUT, R. A., VATANKHAH, S., 2020, "A tutorial and open source software for the efficient evaluation of gravity and magnetic kernels", *Computers & Geosciences*, v. 144, pp. 104575.

HORN, R. A., JOHNSON, C. R., 1991, *Topics in Matrix Analysis.* Cambridge University Press. ISBN: 0-521-30587-X.

JAIN, A. K., 1989, *Fundamentals of Digital Image Processing.* Pearson. ISBN: 0-13-336165-9.

KELLOGG, O. D., 1929, *Foundations of Potential Theory.* Frederick Ungar Publishing Company.

LEÃO, J. W. D., SILVA, J. B. C., 1989, "Discrete linear transformations of potential field data", *GEOPHYSICS*, v. 54, n. 4, pp. 497–507. doi: 10.1190/1.1442676.

LI, Y., OLDENBURG, D. W., 2010, "Rapid construction of equivalent sources using wavelets", *GEOPHYSICS*, v. 75, n. 3, pp. L51–L59. doi: 10.1190/1.3378764.

LI, Y., NABIGHIAN, M., OLDENBURG, D. W., 2014, "Using an equivalent source with positivity for low-latitude reduction to the pole without striation", *Geophysics*, v. 79, n. 6, pp. J81–J90.

MENDONÇA, C. A., SILVA, J. B. C., 1994, "The equivalent data concept applied to the interpolation of potential field data", *GEOPHYSICS*, v. 59, n. 5, pp. 722–732. doi: 10.1190/1.1443630.

NEUDECKER, H., 1969, "Some Theorems on Matrix Differentiation with Special Reference to Kronecker Matrix Products", *Journal of the American Statistical Association*, v. 64, n. 327, pp. 953–963. doi: 10.1080/01621459.1969.10501027. Disponível em: <`https://www.tandfonline.com/doi/abs/10.1080/01621459.1969.10501027`>.

OLIVEIRA JR., V. C., BARBOSA, V. C. F., UIEDA, L., 2013, "Polynomial equivalent layer", *GEOPHYSICS*, v. 78, n. 1, pp. G1–G13. doi: 10.1190/geo2012-0196.1.

QIANG, J., ZHANG, W., LU, K., et al., 2019, "A fast forward algorithm for three-dimensional magnetic anomaly on undulating terrain", *Journal of Applied Geophysics*, v. 166, pp. 33 – 41. ISSN: 0926-9851. doi: https://doi.org/10.1016/j.jappgeo.2019.04.009. Disponível em: <`http://www.sciencedirect.com/science/article/pii/S0926985118309558`>.

RENAUT, R. A., HOGUE, J. D., VATANKHAH, S., et al., 2020, "A fast methodology for large-scale focusing inversion of gravity and magnetic data using the structured model matrix and the 2-D fast Fourier transform", *Geophysical Journal International*, v. 223, n. 2, pp. 1378–1397.

SILVA, J. B. C., 1986, "Reduction to the pole as an inverse problem and its application to low-latitude anomalies", *GEOPHYSICS*, v. 51, n. 2, pp. 369–382. doi: 10.1190/1.1442096.

SIQUEIRA, F. C., OLIVEIRA JR, V. C., BARBOSA, V. C., 2017, "Fast iterative equivalent-layer technique for gravity data processing: A method grounded on excess mass constraint", *Geophysics*, v. 82, n. 4, pp. G57–G69.

SUN, S., CHEN, C., LIU, Y., 2019, "Constrained 3D inversion of magnetic data with structural orientation and borehole lithology: A case study in the Macheng iron deposit, Hebei, China", *Geophysics*, v. 84, n. 2, pp. B121–B133.

VAN LOAN, C., 1992, *Computational Frameworks for the fast Fourier transform*. SIAM.

XIA, J., SPROWL, D. R., 1991, "Correction of topographic distortion in gravity data", *GEOPHYSICS*, v. 56, n. 4, pp. 537–541. doi: 10.1190/1.1443070.

XIA, J., SPROWL, D. R., ADKINS-HELJESON, D., 1993, "Correction of topographic distortions in potential-field data; a fast and accurate approach", *GEOPHYSICS*, v. 58, n. 4, pp. 515–523. doi: 10.1190/1.1443434.

ZHANG, Y., WONG, Y. S., 2015, "BTTB-based numerical schemes for three-dimensional gravity field inversion", *Geophysical Journal International*, v. 203, n. 1, pp. 243–256.

ZHANG, Y., WONG, Y. S., LIN, Y., 2016, "BTTB–RRCG method for downward continuation of potential field data", *Journal of applied Geophysics*, v. 126, pp. 74–86.