Convolutional equivalent layer for magnetic data processing

Diego Takahashi[†], Vanderlei C. Oliveira Jr.[†*] and Valéria C. F. Barbosa[†]

[†] *Observatório Nacional, Department of Geophysics, Rio de Janeiro, Brazil*

[*] *Corresponding author: vanderlei@on.br*

(May 18, 2021)

GEO-XXXX

Running head: **Magnetic convolutional equivalent layer**

ABSTRACT

INTRODUCTION

Equivalent layer background:

Dampney (1969), Blakely (1996), Emilia (1973), Li et al. (2014)

Fast methods for equivalent layer:

Leão and Silva (1989), Mendonça and Silva (1994), Oliveira Jr. et al. (2013), Li and Oldenburg (2010), Siqueira et al. (2017), Mendonça (2020)

Methods for solving Toeplitz matrices:

Golub and Loan (2013), Levinson (1946), Chan and Jin (2007)

Toeplitz matrices in Potential methods:

Zhang and Wong (2015), Zhang et al., 2016, Hogue et al. (2020), Renaut et al. (2020)

Convolutional equivalent layer grav:

Siqueira et al. (2017), Takahashi et al. (2020)

About this work:

METHODOLOGY

The total-field anomaly

The Earth's magnetic field is commonly divided in three parts: main field, crustal field and external field. The main field is generated in the outter core in a process of self-sustaining dynamo, the crustal field is generated by magnetic bodies in the litosphere and the external field is generated by electrical currents in the ionosphere and magnetosphere. For exploration geophysics, the crustal field is the object of study, which makes the separation of this data from the acquisition dataset a very important step.

The combination of the main field and crustal field is known as internal field or total-field. Taking the difference between this internal field and the main field given by a model (e.g. IGRF), at the same location, we have the total-field anomaly.

Let $(x_i, y_i, z_i), i = 1, 2, 3, ...N$, be a observed dataset in a region, the total-field anomaly can be approximated to:

$$\Delta T(x_i, y_i, z_i) = \hat{\mathbf{F}}^\top \mathbf{B}(x_i, y_i, z_i)\,, \tag{1}$$

where, $\mathbf{B}(x_i, y_i, z_i)$ is the crustal field, $\hat{\mathbf{F}}^\top$ is the transposed unitary vector with the main field directions, with $\hat{\mathbf{F}}$ described as:

$$\hat{\mathbf{F}} = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} \cos(I_0)\,\cos(D_0) \\ \cos(I_0)\,\sin(D_0) \\ \sin(I_0) \end{bmatrix}, \tag{2}$$

where I_0 is the inclination and D_0 the declination of the main field , respectively.

Considering a uniform magnetized body with volume v and a magnetization $\mathbf{m}$, the

induced magnetic field is:

$$\mathbf{B}(x_i, y_i, z_i) = c_m \, \frac{\mu_0}{4\pi} \, \mathbf{M}(x_i, y_i, z_i) \, \mathbf{m} \, , \tag{3}$$

where, $\mu_0 = 4\pi \, 10^{-7}$ H/m is the magnetic constant, $c_m = 10^9$ is a constant that transforms the induced magnetic field from Tesla (T) to nanotesla (nT) and $\mathbf{M}(x_i, y_i, z_i)$ is a matrix given by:

$$\mathbf{M}(x_i, y_i, z_i) = \begin{bmatrix} \partial_{x_i x_i}\phi(x_i, y_i, z_i) & \partial_{xy}\phi(x_i, y_i, z_i) & \partial_{x_i z_i}\phi(x_i, y_i, z_i) \\ \partial_{x_i y_i}\phi(x_i, y_i, z_i) & \partial_{yy}\phi(x_i, y_i, z_i) & \partial_{y_i z_i}\phi(x_i, y_i, z_i) \\ \partial_{x_i z_i}\phi(x_i, y_i, z_i) & \partial_{yz}\phi(x_i, y_i, z_i) & \partial_{z_i z_i}\phi(x_i, y_i, z_i) \end{bmatrix} , \tag{4}$$

where, $\partial_{\alpha\beta}\phi(x_i, y_i, z_i, x_j, y_j, z_j)$ with $\alpha = x_i, y_i, z_i$ and $\beta = x_i, y_i, z_i$, are the second derivatives of the function $\phi(x_i, y_i, z_i, x_j, y_j, z_j)$ with respect to x_i, y_i and z_i:

$$\phi(x_i, y_i, z_i) = \int\int_v \int \frac{1}{r} \, dv, \tag{5}$$

where the function $\dfrac{1}{r}$ is given by:

$$\frac{1}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}} \, , \tag{6}$$

and x_j, y_j, z_j are the jth Cartesian coordinates of the volume elements where the integral (equation 5) is conducted within the body with volume v.

Rewriting the equation 1 using the magnetic induction given by equation 3 the total-field anomaly $\Delta T(x, y, z)$ is given by:

$$\Delta T(x_i, y_i, z_i) = c_m \, \frac{\mu_0}{4\pi} \, m \, \hat{\mathbf{F}}^\top \mathbf{M}(x_i, y_i, z_i) \, \hat{\mathbf{m}} \, , \tag{7}$$

where m is the magnetization intensity and $\hat{\mathbf{m}}$ is the directional unitary vector.

Equivalent layer for magnetic data

It is possible to calculate the total-field anomaly $\Delta T(x_i, y_i, z_i)$ (equation 7) with the convolution between the harmonic function and the physical property:

$$\Delta T(x_i, y_i, z_i) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} p(x_j, y_j, z_c)\, \left[c_m\, \frac{\mu_0}{4\pi}\, \hat{\mathbf{F}}^\top \mathbf{H}\, \hat{\mathbf{h}}\right]\, dx\, dy\,, \tag{8}$$

where z_c is a constant representing the depth of the equivalent layer with $z_i < z_c$. The unitary vector $\hat{\mathbf{h}}$ is the magnetization directions of the equivalent sources across the layer:

$$\hat{\mathbf{h}} = \begin{bmatrix} h_x \\ h_y \\ h_z \end{bmatrix} = \begin{bmatrix} \cos(I)\, \cos(D) \\ \cos(I)\, \sin(D) \\ \sin(I) \end{bmatrix}\,, \tag{9}$$

and $\mathbf{H}$ is a 3×3 matrix that contains the second derivatives in relation to the observed dataset Cartesian coordinates x_i, y_i, z_i as presented in equation 6:

$$\mathbf{H} = \begin{bmatrix} \partial_{xx}\frac{1}{r} & \partial_{xy}\frac{1}{r} & \partial_{xz}\frac{1}{r} \\ \partial_{xy}\frac{1}{r} & \partial_{yy}\frac{1}{r} & \partial_{yz}\frac{1}{r} \\ \partial_{xz}\frac{1}{r} & \partial_{yz}\frac{1}{r} & \partial_{zz}\frac{1}{r} \end{bmatrix} = \begin{bmatrix} H_{xx} & H_{xy} & H_{xz} \\ H_{xy} & H_{yy} & H_{yz} \\ H_{xz} & H_{yz} & H_{zz} \end{bmatrix}\,, \tag{10}$$

and the physical property $p(x_j, y_j, z_c)$ represents the magnetic dipole moment distribution over the layer and has unit of Am^2.

Discretizing equation 8:

$$\Delta T(x_i, y_i, z_i) = \sum_{j=1}^{M} p_j a_{ij}\,, \tag{11}$$

where $j = 1, 2, 3, ..., M$ are the equivalent sources discretizied and distributed across the layer and a_{ij} is given by:

$$a_{ij} = c_m\, \frac{\mu_0}{4\pi}\, \hat{\mathbf{F}}^\top \mathbf{H}_{ij}\, \hat{\mathbf{h}}\,. \tag{12}$$

The equation 12 of the magnetic equivalent layer can be written in the matrix form as:

$$\mathbf{d}(\mathbf{p}) = \mathbf{A}\mathbf{p}\,, \tag{13}$$

where $\mathbf{d}(\mathbf{p})$ is the vector of total-field anomaly data, $\mathbf{A}$ is a matrix containing the elements given in equation 12, also known as the sensitivity matrix and $\mathbf{p}$ is the vector of dipole moments of each equivalent source. By using the normal equations, a conventional manner of solving linear systems, the estimative of the dipole moment parameters vector can be calculated from the total-field anomaly as:

$$\hat{\mathbf{p}} = \left(\mathbf{A}^\top \mathbf{A}\right)^{-1} \mathbf{A}^\top \mathbf{d}^o \,. \tag{14}$$

Equation 14 will be referenced throughout this work as the classical method for solving the equivalent layer.

Conjugate Gradient Least Square method (CGLS)

As an alternative from the clasical method of parameter estimative, the conjugate gradient (CG) is a well-known iterative method for solving symmetric and positive definite linear systems. By minimizing the quadratic form:

$$min\, \Phi(\mathbf{p}) = \frac{1}{2}\, \mathbf{p}^\top\, \mathbf{A}\, \mathbf{p}\, - \mathbf{d^o}^\top\, \mathbf{p}\,, \tag{15}$$

it is possible to solve the system by constructing a basis of conjugate directions $c \in R^N$ (Aster et al., 2018). As the matrix $\mathbf{A}$ (equation 12) is not symmetric, instead we apply a general least square method minimizing:

$$min\, ||\mathbf{A}\, \mathbf{p} - \mathbf{d}^o||_2\,, \tag{16}$$

by applying CG to the normal equations (equation 14).

In theory, this method is bound to converge at N iterations maximum, but in a later part of this work we show with numerical results that the convergence is much faster for the linear system we are solving.

A pseudocode for the CGLS method follows:

Algorithm 1 Conjugate Gradient Least Square pseudocode (Aster et al., 2018, p. 166).

Input: $\mathbf{A} \in R^{N \times M}$ and $\mathbf{d}^o \in R^N$.

Output: Estimative of parameter vector $\hat{\mathbf{p}}$.

Let $it = 0$, $\hat{\mathbf{p}}^{(0)} = \mathbf{0}$, $\mathbf{c}^{(-1)} = \mathbf{0}$, $\beta_0 = 0$, $\mathbf{s}^{(0)} = \mathbf{d}^o - \mathbf{A}\hat{\mathbf{p}}^{(0)}$ and $\mathbf{r}^{(0)} = \mathbf{A}^\top \mathbf{s}^{(0)}$.

1 - If $it > 0$, let $\beta_{it} = \dfrac{{\mathbf{r}^{(it)}}^\top \mathbf{r}^{(it)}}{{\mathbf{r}^{(it-1)}}^\top \mathbf{r}^{(it-1)}}$

2 - $\mathbf{c}^{(it)} = \mathbf{r}^{it} - \alpha_{it}\, \beta_{it}\, \mathbf{c}^{(it-1)}$.

3 - $\alpha_{it} = \dfrac{||\mathbf{r}^{(it)}||_2^2}{({\mathbf{c}^{(it)}}^\top \mathbf{A}^\top)(\mathbf{A}\, \mathbf{c}^{(it)})}$.

4 - $\hat{\mathbf{p}}^{(it+1)} = \hat{\mathbf{p}}^{it} - \alpha_{it}\, \mathbf{c}^{(it)}$.

5 - $\mathbf{s}^{(it+1)} = \mathbf{s}^{it} - \alpha_{it}\, \mathbf{A}\, \mathbf{c}^{(it)}$.

6 - $\mathbf{r}^{(it+1)} = \mathbf{A}^\top\, \mathbf{s}^{(it+1)}$.

7 - $it = it + 1$.

8 - Repeat previous steps until convergence.

As shown in Algorithm 1, different from the classical method (equation 14), no inverse matrix and no matrix-matrix product need to be calculate, but one matrix-vector multiplication is needed out of the loop and two matrix-vector multiplications, in steps 3 and 6, are necessary at each iteration. In this work, we will reduce the computational cost of the equivalent layer by substituting exactly these two matrix-vector products with a much more efficient algorithm.

Non-symmetric Block-Toeplitz Toeplitz-Block structure of matrix A

The notation used in this work will be the same as the one presented in Takahashi et al. (2020), where the authors described the structure of the symmetric Block-Toeplitz Toeplitz-

Block matrix of the gravimetric equivalent layer by establishing a relation between the pair of *matrix coordinates* (x_i, y_i), $i = 1, ..., N$ or (x_j, y_j), $j = 1, ..., M = N$ to a pair of *grid coordinates* (x_k, y_l) given as:

$$x_i \equiv x_k = x_1 + [k(i) - 1]\ \Delta x\ , \tag{17}$$

and

$$y_i \equiv y_l = y_1 + [l(i) - 1]\ \Delta y\ . \tag{18}$$

In a *x-oriented grid* the indices i (or j) relate as integer functions of k and l by:

$$i(k, l) = (l - 1)\, N_x + k \quad , \tag{19}$$

$$l(i) = \left\lceil \frac{i}{N_x} \right\rceil \tag{20}$$

and

$$k(i) = i - \left\lceil \frac{i}{N_x} \right\rceil N_x + N_x \quad . \tag{21}$$

For *y-oriented grid* they are given by:

$$i(k, l) = (k - 1)\, N_y + l \quad , \tag{22}$$

$$k(i) = \left\lceil \frac{i}{N_y} \right\rceil \tag{23}$$

and

$$l(i) = i - \left\lceil \frac{i}{N_y} \right\rceil N_y + N_y \quad . \tag{24}$$

Figure **??** shows an example of a grid $N_x \times N_y$, where $N_x = 4$ and $N_y = 3$ demonstrating the relation between the *matrix coordinates* with $k(i)$ and $l(i)$ depending on the orientation of the grid.

In a *x-oriented grid* q and p gives the number of blocks ($Q = N_y$) and the number of elements of each block ($P = N_x$). They can be defined by the integer functions:

$$q(i, j) = \ l(i) - l(j) \tag{25}$$

and

$$p(i,j) = \ k(i) - k(j) \quad . \tag{26}$$

Equations 20 and 21 describes $l(i)$ and $l(j)$ and $k(i)$ and $k(j)$, respectively. When using y-oriented grids, q and p still defines the number of block and block elements, but $Q = N_x$ and $P = N_y$. But, the integer functions changes to:

$$q(i,j) = \ k(i) - k(j) \tag{27}$$

and

$$p(i,j) = \ l(i) - l(j) \quad , \tag{28}$$

where equation 23 now defines $k(i)$ and $k(j)$ and equation 24 defines $l(i)$ and $l(j)$. Note that equations 25, 26, 27 and 27 differs from the ones presented in Takahashi et al. (2020) by the absence of the module.

In both x or y-*orientation*, matrix $\mathbf{A}$ (equation 12) can be rewritten by the indices q, $Q = -Q+1, ..., 0, ..., Q-1$ defining the number of its blocks:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}^0 & \mathbf{A}^{-1} & \cdots & \mathbf{A}^{-Q+1} \\ \mathbf{A}^1 & \mathbf{A}^0 & \mathbf{A}^{-1} & \vdots \\ \vdots & \mathbf{A}^1 & \ddots & \mathbf{A}^{-1} \\ \mathbf{A}^{Q-1} & \cdots & \mathbf{A}^1 & \mathbf{A}^0 \end{bmatrix}_{N \times N} \quad , \tag{29}$$

and by indice p, where each block has $P \times P$ elements a_p^q, $p = -P+1, ..., 0, \ldots, P-1$, :

$$\mathbf{A}_q = \begin{bmatrix} a_0^q & a_{-1}^q & \cdots & a_{-P+1}^q \\ a_1^q & a_0^q & a_{-1}^q & \vdots \\ \vdots & a_1^q & \ddots & a_{-1}^q \\ a_{P-1}^q & \cdots & a_1^q & a_0^q \end{bmatrix}_{P \times P} \quad , \tag{30}$$

In general, matrix $\mathbf{A}$ (equation 12) is a non-symmetric BTTB, i.e., its blocks are non-symmetric ($\mathbf{A}^{-1} \neq \mathbf{A}^{1}$) and its elements also are non-symmetric ($a^{q}_{-1} \neq a^{q}_{1}$). Depending on specific values of the main field direction and the equivalent sources magnetization directions, matrix $\mathbf{A}$ can assume other structures, for example, when $\hat{\mathbf{F}} = [0, 0, 1]$ and $\hat{\mathbf{h}} = [0, 0, 1]$ it becomes symmetric. In this work we are considering the more commom situation for the matrix $\mathbf{A}$.

Also differently for the symmetric sensitivity matrix described by Takahashi et al. (2020), the non-symmetric BTTB matrix cannot be reconstructed only by its first column. Matrix $\mathbf{A}$ (equation 12) needs four: the first an last columns of the first column of blocks and the first and last columns of the last column of blocks. This has a physical implication in the equivalent layer which is not possible to use only one equivalent source to reprduce all elements of matrix $\mathbf{A}$, but instead takes four equivalent sources positioned at each corner. Figure **??** shows the positioning of the equivalent sources in a regular grid $N_x = 4$ $N_y = 3$ necessary to calculate the four columns capable of recover matrix $\mathbf{A}$.

In this work, we propose a different approach, by calculating the first column of all six different components of second derivatives matrices from $\mathbf{H}_{ij}$ (equation 10). These matrices are, in fact, symmetrics or skew-symmetrics BTTBs, meaning that the first column has all elements of each matrix.

By substituting equations 2, 9 and 10 into equation 12, it is possible to describe each element of the sensitivity matrix by the second derivative components of $\mathbf{H}_{ij}$:

$$\begin{aligned} a_{ij} = c_m \frac{\mu_0}{4\pi} \left(F_x H_{xx} + F_y H_{xy} + F_z H_{xz}\right) h_x + \\ \left(F_x H_{xy} + F_y H_{yy} + F_z H_{yz}\right) h_y + \\ \left(F_x H_{xz} + F_y H_{yz} + F_z H_{zz}\right) h_z \, . \end{aligned} \tag{31}$$

If we consider that $c_m \frac{\mu_0}{4\pi}$, $\hat{\mathbf{F}}$ (equation 2) and $\hat{\mathbf{h}}$ (equation9) are constants multiplying the second derivatives $\mathbf{H}$ (equation 10), the sensitivity matrix $\mathbf{A}$ (equation 12) is purely the sum of the components $H_{xx}+H_{xy}+H_{xz}+H_{xy}+H_{yy}+H_{yz}+H_{xz}+H_{yz}+H_{zz}$ multiplied by the respectives constants of each component. Thus, despite $\mathbf{A}$ not being a symmetric BTTB matrix, it can be in fact, written by calculating only the first column of these components. In the next few sections we will describe each component $\mathbf{H}$ as its own matrix.

Structure of matrices components $\mathbf{H_{xx}}$, $\mathbf{H_{yy}}$ and $\mathbf{H_{zz}}$

We can describe the elements of $\mathbf{H_{xx}}$, $\mathbf{H_{yy}}$ and $\mathbf{H_{zz}}$ by substituting equations 17 and 18 in equation 10 as:

$$h_{ij}^{xx} = \frac{1}{\left[(\Delta k_{ij}\,\Delta x)^2 + (\Delta l_{ij}\,\Delta y)^2 + (\Delta z)^2\right]^{\frac{3}{2}}} + \frac{3(\Delta k_{ij}\,\Delta x)^2}{\left[(\Delta k_{ij}\,\Delta x)^2 + (\Delta l_{ij}\,\Delta y)^2 + (\Delta z)^2\right]^{\frac{5}{2}}}\,, \quad (32)$$

$$h_{ij}^{yy} = \frac{1}{\left[(\Delta k_{ij}\,\Delta x)^2 + (\Delta l_{ij}\,\Delta y)^2 + (\Delta z)^2\right]^{\frac{3}{2}}} + \frac{3(\Delta k_{ij}\,\Delta y)^2}{\left[(\Delta k_{ij}\,\Delta x)^2 + (\Delta l_{ij}\,\Delta y)^2 + (\Delta z)^2\right]^{\frac{5}{2}}} \quad (33)$$

and

$$h_{ij}^{zz} = \frac{1}{\left[(\Delta k_{ij}\,\Delta x)^2 + (\Delta l_{ij}\,\Delta y)^2 + (\Delta z)^2\right]^{\frac{3}{2}}} + \frac{3(\Delta z)^2}{\left[(\Delta k_{ij}\,\Delta x)^2 + (\Delta l_{ij}\,\Delta y)^2 + (\Delta z)^2\right]^{\frac{5}{2}}}\,. \quad (34)$$

The principal components $\mathbf{H_{xx}}$, $\mathbf{H_{yy}}$ and $\mathbf{H_{zz}}$ (equations 32, 33 and 34, respectively) of matrix $\mathbf{H}$ (equation 10) are symmetric-Block-Toeplitz symmetric-Toeplitz-Block matrices. This means that $\mathbf{H_{xx}}$, $\mathbf{H_{yy}}$ and $\mathbf{H_{zz}}$ are Toeplitz and symmetric by its blocks and each of the blocks are symmetric Toeplitz matrices. For example, $\mathbf{H_{xx}}$ can be described by the *block indice* q that represent the block diagonals of this matrix as a grid of $Q \times Q$ blocks

$\mathbf{H}^q_{\mathbf{xx}}$, $q = 0, \ldots, Q-1$:

$$\mathbf{H_{xx}} = \begin{bmatrix} \mathbf{H}^0_{\mathbf{xx}} & \mathbf{H}^1_{\mathbf{xx}} & \cdots & \mathbf{H}^{Q-1}_{\mathbf{xx}} \\ \mathbf{H}^1_{\mathbf{xx}} & \mathbf{H}^0_{\mathbf{xx}} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{H}^1_{\mathbf{xx}} \\ \mathbf{H}^{Q-1}_{\mathbf{xx}} & \cdots & \mathbf{H}^1_{\mathbf{xx}} & \mathbf{H}^0_{\mathbf{xx}} \end{bmatrix}_{N \times N} . \tag{35}$$

And each diagonal of the blocks are represented by $P \times P$ elements h^q_p, $p = 0, \ldots, P-1$:

$$\mathbf{H}^q_{\mathbf{xx}} = \begin{bmatrix} h^q_0 & h^q_1 & \cdots & h^q_{P-1} \\ h^q_1 & h^q_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & h^q_1 \\ h^q_{P-1} & \cdots & h^q_1 & h^q_0 \end{bmatrix}_{P \times P} . \tag{36}$$

In a *x-oriented grid* $Q = N_x$, $P = N_y$ and q and p can be defined by the functions:

$$q(i,j) = \mid l(i) - l(j) \mid \tag{37}$$

and

$$p(i,j) = \mid k(i) - k(j) \mid \quad , \tag{38}$$

where $l(i)$ and $l(j)$ are defined by equation 20 and $k(i)$ and $k(j)$ are defined by equation 21. For y-oriented grids, $Q = N_x$, $P = N_y$ and the block indices q and p are defined, respectively, by the following integer functions of the matrix indices i and j:

$$q(i,j) = \mid k(i) - k(j) \mid \tag{39}$$

and

$$p(i,j) = \mid l(i) - l(j) \mid \quad , \tag{40}$$

This struture can also describe matrices $\mathbf{H_{yy}}$ and $\mathbf{H_{zz}}$ in the same manner.

Structure of the components matrices $\mathbf{H_{xy}}$

By substituting equations 17 and 18 in equation 10, we can also describe the elements of $\mathbf{H_{xy}}$, as:

$$h_{ij}^{xy} = \frac{3(\Delta k_{ij}\,\Delta x)(\Delta l_{ij}\,\Delta y)}{\left[(\Delta k_{ij}\,\Delta x)^2 + (\Delta l_{ij}\,\Delta y)^2 + (\Delta z)^2\right]^{\frac{5}{2}}}\,, \tag{41}$$

The component $\mathbf{H_{xy}}$ (equation 41) of matrix $\mathbf{H}$ (equation 10) are skew symmetric-Block-Toeplitz skew symmetric-Toeplitz-Block matrices. This means that $\mathbf{H_{xy}}$ is Toeplitz and skew symmetric by its blocks and each of the blocks are skew symmetric Toeplitz matrices. This way, matrix $\mathbf{H_{xy}}$ can be described by the *block indice* q that represent the block diagonals of this matrix as a grid of $Q \times Q$ blocks $\mathbf{H_{xx}^q}$, $q = -Q+1, \ldots, 0, \ldots, Q-1$:

$$\mathbf{H_{xy}} = \begin{bmatrix} \mathbf{H_{xx}^0} & \mathbf{H_{xy}^{-1}} & \cdots & \mathbf{H_{xy}^{-Q+1}} \\ \mathbf{H_{xy}^1} & \mathbf{H_{xy}^0} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{H_{xy}^{-1}} \\ \mathbf{H_{xy}^{Q-1}} & \cdots & \mathbf{H_{xy}^1} & \mathbf{H_{xy}^0} \end{bmatrix}_{N\times N} . \tag{42}$$

And each diagonal of the blocks are represented by $P \times P$ elements h_p^q, $p = -P+1, \ldots, 0, \ldots, P-1$:

$$\mathbf{H_{xx}^q} = \begin{bmatrix} h_0^q & h_{-1}^q & \cdots & h_{-P+1}^q \\ h_1^q & h_0^q & \ddots & \vdots \\ \vdots & \ddots & \ddots & h_{-1}^q \\ h_{P-1}^q & \cdots & h_1^q & h_0^q \end{bmatrix}_{P\times P} . \tag{43}$$

In a *x-oriented grid* $Q = N_x$, $P = N_y$ and q and p can be defined by the functions:

$$q(i,j) = \; l(i) - l(j) \tag{44}$$

and

$$p(i,j) = \; k(i) - k(j) \quad , \tag{45}$$

where $l(i)$ and $l(j)$ are defined by equation 20 and $k(i)$ and $k(j)$ are defined by equation 21. For y-oriented grids, $Q = N_x$, $P = N_y$ and the block indices q and p are defined, respectively, by the following integer functions of the matrix indices i and j:

$$q(i,j) = \; k(i) - k(j) \tag{46}$$

and

$$p(i,j) = \; l(i) - l(j) \quad , \tag{47}$$

Important to clarify that in this case, as a skew symmetric matrix, the values of oposing diagonals have oposing signals, e.g., $\mathbf{H}_{\mathbf{xx}}^{-1} = -\mathbf{H}_{\mathbf{xx}}^{1}$ and $h_{-1}^{q} = -h_{1}^{q}$.

Structure of the components matrices $\mathbf{H_{xz}}$

By substituting equations 17 and 18 in equation 10, the elements of $\mathbf{H_{xz}}$, are given by:

$$h_{ij}^{xz} = \frac{3(\Delta k_{ij}\,\Delta x)(\Delta z)}{\left[(\Delta k_{ij}\,\Delta x)^2 + (\Delta l_{ij}\,\Delta y)^2 + (\Delta z)^2\right]^{\frac{5}{2}}} \, , \tag{48}$$

The component $\mathbf{H_{xz}}$ (equation 48) of matrix $\mathbf{H}$ (equation 10) are skew symmetric-Block-Toeplitz symmetric-Toeplitz-Block matrices. This means that $\mathbf{H_{xz}}$ is Toeplitz and skew symmetric by its blocks and each of the blocks are symmetric Toeplitz matrices. Thus, matrix $\mathbf{H_{xz}}$ can be described by the *block indice* q that represent the block diagonals of this matrix as a grid of $Q \times Q$ blocks $\mathbf{H}_{\mathbf{xx}}^{q}$, $q = -Q+1, \ldots, 0, \ldots, Q-1$:

$$\mathbf{H_{xz}} = \begin{bmatrix} \mathbf{H}_{\mathbf{xz}}^{0} & \mathbf{H}_{\mathbf{xz}}^{-1} & \cdots & \mathbf{H}_{\mathbf{xz}}^{-Q+1} \\ \mathbf{H}_{\mathbf{xz}}^{1} & \mathbf{H}_{\mathbf{xz}}^{0} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{H}_{\mathbf{xz}}^{-1} \\ \mathbf{H}_{\mathbf{xz}}^{Q-1} & \cdots & \mathbf{H}_{\mathbf{xz}}^{1} & \mathbf{H}_{\mathbf{xz}}^{0} \end{bmatrix}_{N \times N} \, . \tag{49}$$

And each diagonal of the blocks are represented by $P \times P$ elements h_p^q, $p = 0, \ldots, P-1$:

$$\mathbf{H}_{\mathbf{xx}}^q = \begin{bmatrix} h_0^q & h_1^q & \cdots & h_{P-1}^q \\ h_1^q & h_0^q & \ddots & \vdots \\ \vdots & \ddots & \ddots & h_1^q \\ h_{P-1}^q & \cdots & h_1^q & h_0^q \end{bmatrix}_{P \times P} \quad . \tag{50}$$

In a *x-oriented grid* $Q = N_x$, $P = N_y$ and q and p can be defined by the functions:

$$q(i,j) = \; l(i) - l(j) \tag{51}$$

and

$$p(i,j) = \mid k(i) - k(j) \mid \quad , \tag{52}$$

where $l(i)$ and $l(j)$ are defined by equation 20 and $k(i)$ and $k(j)$ are defined by equation 21. For y-oriented grids, $Q = N_x$, $P = N_y$ and the block indices q and p are defined, respectively, by the following integer functions of the matrix indices i and j:

$$q(i,j) = \; k(i) - k(j) \tag{53}$$

and

$$p(i,j) = \mid l(i) - l(j) \mid \quad , \tag{54}$$

In this case as a skew symmetric matrix by blocks, the values of oposing diagonals blocks have oposing signals, e.g., $\mathbf{H}_{\mathbf{xx}}^{-1} = -\mathbf{H}_{\mathbf{xx}}^{1}$ but each block is a symmetric matrix.

Structure of the components matrices $\mathbf{H_{yz}}$

Finally, by substituting equations 17 and 18 in equation 10, we can also describe the elements of $\mathbf{H_{yz}}$, as:

$$h_{ij}^{yz} = \frac{3(\Delta l_{ij}\, \Delta y)(\Delta z)}{\left[(\Delta k_{ij}\, \Delta x)^2 + (\Delta l_{ij}\, \Delta y)^2 + (\Delta z)^2 \right]^{\frac{5}{2}}} \, , \tag{55}$$

The component $\mathbf{H_{yz}}$ (equation 55) of matrix $\mathbf{H}$ (equation 10) are symmetric-Block-Toeplitz skew symmetric-Toeplitz-Block matrices. This means that $\mathbf{H_{yz}}$ is Toeplitz and symmetric by its blocks and each of the blocks are skew symmetric Toeplitz matrices. Thus, matrix $\mathbf{H_{yz}}$ can be described by the *block indice* q that represent the block diagonals of this matrix as a grid of $Q \times Q$ blocks $\mathbf{H}^q_{\mathbf{xx}}$, $q = 0, \ldots, Q-1$:

$$\mathbf{H_{yz}} = \begin{bmatrix} \mathbf{H}^0_{\mathbf{yz}} & \mathbf{H}^1_{\mathbf{yz}} & \cdots & \mathbf{H}^{Q-1}_{\mathbf{yz}} \\ \mathbf{H}^1_{\mathbf{yz}} & \mathbf{H}^0_{\mathbf{yz}} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{H}^1_{\mathbf{yz}} \\ \mathbf{H}^{Q-1}_{\mathbf{yz}} & \cdots & \mathbf{H}^1_{\mathbf{yz}} & \mathbf{H}^0_{\mathbf{yz}} \end{bmatrix}_{N \times N} . \tag{56}$$

And each diagonal of the blocks are represented by $P \times P$ elements h^q_p, $p = -P + 1, \ldots, 0, \ldots, P-1$:

$$\mathbf{H}^q_{\mathbf{xx}} = \begin{bmatrix} h^q_0 & h^q_{-1} & \cdots & h^q_{-P+1} \\ h^q_1 & h^q_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & h^q_{-1} \\ h^q_{P-1} & \cdots & h^q_1 & h^q_0 \end{bmatrix}_{P \times P} . \tag{57}$$

In a *x-oriented grid* $Q = N_x$, $P = N_y$ and q and p can be defined by the functions:

$$q(i,j) = \mid l(i) - l(j) \mid \tag{58}$$

and

$$p(i,j) = \ k(i) - k(j) \quad , \tag{59}$$

where $l(i)$ and $l(j)$ are defined by equation 20 and $k(i)$ and $k(j)$ are defined by equation 21. For y-oriented grids, $Q = N_x$, $P = N_y$ and the block indices q and p are defined, respectively, by the following integer functions of the matrix indices i and j:

$$q(i,j) = \mid k(i) - k(j) \mid \tag{60}$$

and

$$p(i,j) = \ l(i) - l(j) \quad , \tag{61}$$

Being a symmetric matrix by blocks, the values of $\mathbf{H_{yz}}$ from oposing diagonals blocks are equal, but each block have skew symmetric oposing diagonals, i.e., $h^q_{-1} = -h^q_1$.

CGLS matrix-vector substitution

As pointed earlier in this work, the main improvement inside the CGLS method (Algorithm 1) for estimating the parameter vector $\hat{\mathbf{p}}$ (equation 14) is to substitute the matrix-vector multiplication $\mathbf{A}^\top \mathbf{s}^{(0)}$ out of the loop and primarily the two matrix-vector multiplications inside the loop at steps 3 an 6, $\mathbf{A}\,\mathbf{c}^{(it)}$ and $\mathbf{A}^\top\,\mathbf{s}^{(it+1)}$, that is necessary at each iteration and takes most of its runtime.

Our method consists in calculating the six first columns of the second derivatives of $\mathbf{H}$ (equation 10) and embbed them into the first six columns of the block-circulant circulant-block (BCCB) matrices related to the $\mathbf{H}$ components. Thus, it is possible to calculate the first column of the BCCB matrix embbeded from matrix $\mathbf{A}$ (equation 12) by multiplying each component with its respective constants and summing as shown in equation 31. In Takahashi et al. (2020), Appendix A, the authors demonstrated in details how to transform a symmetric BTTB matrix into a BCCB matrix $\mathbf{C}$. The process here is the same and that work can be referenced to achieve the same results.

A new auxuliary linear system is constructed to carry the matrix-vector product:

$$\mathbf{w} = \mathbf{C}\mathbf{v}\,, \tag{62}$$

where

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_0 \\ \vdots \\ \mathbf{w}_{Q-1} \\ \mathbf{0}_{2N\times 1} \end{bmatrix}_{4N\times 1} , \tag{63}$$

$$\mathbf{w}_q = \begin{bmatrix} \mathbf{d}_q(\mathbf{p}) \\ \mathbf{0}_{P\times 1} \end{bmatrix}_{2P\times 1} , \tag{64}$$

$$\mathbf{v} = \begin{bmatrix} \mathbf{v}_0 \\ \vdots \\ \mathbf{v}_{Q-1} \\ \mathbf{0}_{2N\times 1} \end{bmatrix}_{4N\times 1} , \tag{65}$$

and

$$\mathbf{v}_q = \begin{bmatrix} \mathbf{p}_q \\ \mathbf{0}_{P\times 1} \end{bmatrix}_{2P\times 1} , \tag{66}$$

where $\mathbf{C}$ (equation 62) is a $4N \times 4N$ non-symmetric (BCCB) resulted from transforming $\mathbf{A}$ (equation 12). Without having to calculate the whole BCCB matrix, its first column can be used to carry the multiplication of this new system (equation 62). Appendix A and C in Takahashi et al. (2020) shows how to use the 2D-FFT to compute the eigenvalues of matrix $\mathbf{C}$, store in a $2Q \times 2P$ matrix using the *vec*-operator and to carry the matrix-vector product. Denoting matrix $\mathbf{L}$ as the eigenvalues matrix follows:

$$\mathbf{F}^*_{2Q} \left[\mathbf{L} \circ (\mathbf{F}_{2Q}\, \mathbf{V}\, \mathbf{F}_{2P})\right] \mathbf{F}^*_{2P} = \mathbf{W} , \tag{67}$$

where the symbol "$\circ$" references the Hadamard product, i.e., a element-wise complex multiplication between the eingenvalues and the 2D-FFT of the matrix rearranged along the rows of the parameters $\mathbf{V}$ (equation 65) using the same *vec*-operator. The resulting inverse

2D-FFT denoted by $\mathbf{F}^*_{2Q} \otimes \mathbf{F}^*_{2P}$ is also a $2Q \times 2P$ matrix ($\mathbf{W}$) that can be rearranged to the predicted data vector $\mathbf{d}(\hat{\mathbf{p}})$ size N.

Computational performance

To compare the efficiency of our algorithm we will use a numerical approach and calculate the floating-point operations (*flops*), i.e., count the number of mathematical operations necessary to complete the estimative of parameter vector $\hat{\mathbf{p}}$ of the normal equations (equation 14) and both the CGLS methods (algorithm 1) for calculating the matrix-vector product by its standart way and our approach.

The *flops* needed to solve the linear system in equation 14 using the Cholesky factorization is:

$$f_{classical} = \frac{7}{3}N^3 + 6N^2 \,, \tag{68}$$

where N is the total number of observation points and also the size of estimated parameter vector $\hat{\mathbf{p}}$.

For the more efficient CGLS algorithm the estimative can be done in:

$$f_{cgls} = 2N^2 + it\,(4N^2 + 12N) \,. \tag{69}$$

However, our approach reduces further to:

$$f_{ours} = \kappa\, 16N \log_2(4N) + 24N + it\,(\kappa\, 16N \log_2(4N) + 60N) \,, \tag{70}$$

where κ depends on the FFT algorithm. By default, in this work we will use $\kappa = 5$ for the *radix-2* algorithm (Van Loan, 1992).

Figure **??** shows a comparative between the methods varying the number of observation points up to $1,000,000$, where it is possible to observe a reduction of 10^7 orders of magnitude

to estimate parameter vector $\hat{\mathbf{p}}$ in relation to the non-iterative classical method and 10^3 orders of magnitude in relation to the standart CGLS algorithm using 50 iterations. A more detailed, step by step, flops count of the classical and CGLS algorithm can be found in Appendix A.

In figure **??** we show the time necessary to construct matrix $\mathbf{A}$ (equation 12) and solve the linear system up to $10,000$ points of observation. With this dataset the classical method takes more than sixty-three seconds, the 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.

In figure **??** a comparison between the time to complete the task to calculate the first column of the BCCB matrix embbeded from the from $\mathbf{A}$ (equation 12) by using only one equivalent source, i.e., calculating all six first column of the second derivatives matrices from $\mathbf{H}$ (equation 10) and using four equivalent sources to calculate the four necessary columns from the non-symmetric matrix $\mathbf{A}$ (equation 12). Although, very similar in time, with one source a small advantage can be observed as the number of data N increases and goes beyond $N = 200,000$. This test was done from $N = 10,000$ to $N = 700,000$ with increases of $5,625$ observation points.

In Table 1 there is comparison between how much RAM memory is adressed to store the sensitivity matrix for each of the methods. The classical approach and the CGLS have to store the whole matrix $\mathbf{A}$ (equation 12), this means that a dataset with for example $N = 10,000$ observation points, the sensitivity matrix has $N^2 = 100,000,000$ elements and takes approximately 763 Megabytes of memory (8 bytes per element). For our method, it is necessary to store the first six columns of each of the components from matrix $\mathbf{H}$ (equation 10) embedded into the BCCB matrices. With the same dataset $N = 10,000$ it

needs 1.831 Megabytes. After completing the steps to store the eigenvalues of matrix $\mathbf{C}$ (equation 62) it takes only 0.6104 Megabytes (here we are considering 16 bytes per element as the eigenvalues are complex numbers resulting from the 2D FFT). For a bigger dataset as $N = 1,000,000$ the amount of RAM necessary goes to $7,629,395$, 183.096 and 61.035 Megabytes, respectively, showing the necessity to find improved and efficient methods for the equivalent layer technique as the one presented in this work. We remember that throughout our work we are always considering $N = M$.

APPLICATION TO SYNTHETIC DATA

The synthetic data application of the fast equivalent layer for magnetic data was conducted on a regular grid of 80×80 points, totaling $N = 6,400$ observation points. Three bodies were modeled: two prisms and a sphere with inclination, declination and intensity of $0°$ and $45°$ and $2\sqrt{2}$ A/m, respectively. The main field has inclination and declination of $10°$ and $37°$, respectively. Figure **??**a shows the synthetic data created for this test.

Using a classical linear inversion method (equation 14) a predicted data was estimated in figure **??**a. The residuals between the observed (**??**b) and the predicted data (**??**a), with mean -0.0006181 nT and standart deviation of 0.4649 nT is shown in figure **??**b. This process took 17.10 seconds.

Using the CGLS method with the fast BTTB matrix-vector product a predicted data was estimated in figure **??**a. The residuals between the observed (**??**b) and the predicted data (**??**a), with mean -0.01540 nT and standart deviation of 0.6748 nT is shown in figure **??**b. This process took 0.25 seconds.

In figure **??** we have the convergence analysis of the CGLS method to estimate the equivalent sources parameter vector $\mathbf{d}(\hat{\mathbf{p}})$ used for this synthetic test. The squared euclidian norm of residuals decreases as expected, with good results at 50 iterations, stabilizing afterwards. This shows that is not necessary to run the conjugate gradient least square method at N iterations to get a theoretically exactly solution (which in practice would never occur due to roundoff errors), as this could be very demanding to process even with a small layer with $N = 6,400$ equivalent sources. Setting a minimum tolerance of the residuals is a good option to carry this algorithm in a not so costful process and still obtaining very good results. Another possibility is to set an invariance to the euclidian norm of residuals

between iteractions, wich would increase algorithm runtime, but with smaller residuals. We chose the first option, as we achieve satisfatory results.

Tests with irregulars grids

As shown in theory, a regular grid of observation points is needed to arise the BTTB matrix. In this section, we show the results when this algorithm is used directly into irregular grids of $N = 5\,000$ observation points (100×50) with different levels of deviations. First we used three different levels of pertubation on the *x-direction* and *y-direction*: 10%, 30% and 50% of the Δx and Δy spacing (101.0101 m and 163.2653 m, respectively). All perturbations are random noises with mean equals to 0.

Figure **??**a shows a irregular grid with deviations of 10% in the *x-direction* and 10% in the y-direction of the observation points. Using the classical approach the residuals between the observed (**??**b) and the predicted data (**??**a) has mean -0.0006 nT, standart deviation of 0.4539 nT and is shown in figure **??**b. Using the new approach of this work the residuals between the observed (**??**b) and the predicted data (**??**a) has mean -0.0237 nT, standart deviation of 0.7824 nT and is shown in figure **??**b. In figure **??** we have the convergence analysis of the CGLS method to estimate the equivalent sources used for this synthetic test. The squared euclidian norm decreases as expected, with good results at 50 iterations, stabilizing afterwards.

Figure **??**a shows a irregular grid with deviations of 20% in the *x-direction* and 20% in the *y-direction* of the observation points. Using the classical approach the residuals between the observed (**??**b) and the predicted data (**??**a) has mean -0.0006003 nT, standart deviation of 0.4543 nT and is shown in figure **??**b. Using the new approach of this work

the residuals between the observed (**??**b) and the predicted data (**??**a) has mean -0.005673 nT, standart deviation of 0.9093 nT and is shown in figure **??**b. In figure **??** we have the convergence analysis of the CGLS method to estimate the equivalent sources used for this synthetic test. The squared euclidian norm decreases as expected and stabilizing afterwards.

Figure **??**a shows a irregular grid with deviations of 30% in the *x-direction* and 30% in the *y-direction* of the observation points. Using the classical approach the residuals between the observed (**??**b) and the predicted data (**??**a) has mean -0.0006351 nT, standart deviation of 0.4548 nT and is shown in figure **??**b. Using the new approach of this work the residuals between the observed (**??**b) and the predicted data (**??**a) has mean -0.03092 nT, standart deviation of 1.2304 nT and is shown in figure **??**b. In figure **??** we have the convergence analysis of the CGLS method to estimate the equivalent sources used for this synthetic test. The squared euclidian norm decreases as expected in the begining, but starts increasing and not converging afterwards.

This results show that, when this approach is used in irregular grids, there is a limit of how much deviation the observation points can have before it starts to produce errors in predicted data. This is confirmed with the convergence analysis, when the irregularity is too large, the linear system stops converging.

Another set of tests were also made with the same previous grid configuration, but now with deviations in the *z-direction*, i.e., the observation points were no longer in a plane. Again, we used three different levels of pertubation: 5%, 10% and 20% of the observation points height equal to -900 m, with random noises means equal to 0.

Figure **??**a shows a irregular grid with deviations of 5% in the *z-direction* of the observation points. Using the classical approach the residuals between the observed (**??**b) and the

predicted data (**??**a) has mean −0.002141 nT, standart deviation of 0.4692 nT and is shown in figure **??**b. Using the new approach of this work the residuals between the observed (**??**b) and the predicted data (**??**a) has mean 0.01091 nT, standart deviation of 1.3077 nT and is shown in figure **??**b. In figure **??** we have the convergence analysis of the CGLS method to estimate the equivalent sources used for this synthetic test. The squared euclidian norm decreases as expected, with good results at 50 iterations, stabilizing afterwards.

Figure **??**a shows a irregular grid with deviations of 10% in the *z-direction* of the observation points. Using the classical approach the residuals between the observed (**??**b) and the predicted data (**??**a) has mean −0.006071 nT, standart deviation of 0.5027 nT and is shown in figure **??**b. Using the new approach of this work the residuals between the observed (**??**b) and the predicted data (**??**a) has mean 0.04800 nT, standart deviation of 2.2860 nT and is shown in figure **??**b. In figure **??** we have the convergence analysis of the CGLS method to estimate the equivalent sources used for this synthetic test. The squared euclidian norm decreases as expected, with good results at 50 iterations, stabilizing afterwards.

Figure **??**a shows a irregular grid with deviations of 20% in the *z-direction* of the observation points. Using the classical approach the residuals between the observed (**??**b) and the predicted data (**??**a) has mean −0.01717 nT, standart deviation of 0.5768 nT and is shown in figure **??**b. Using the new approach of this work the residuals between the observed (**??**b) and the predicted data (**??**a) has mean −0.3614 nT, standart deviation of 5.9610 nT and is shown in figure **??**b. In figure **??** we have the convergence analysis of the CGLS method to estimate the equivalent sources used for this synthetic test. The squared euclidian norm decreases slower than previous tests and starts increasing afterwards showing that the convergence is not possible.

Once again, we observe that while the classical linear inversion method can predict the data even with high irregularities in the observation points grid, the method presented in this work stars to create errors in the estimative. Through the convergence graphs it is possible to see the increase of the squared euclidian norm of the residuals, but the sensitivity in the *z-direction* is higher than the x and *y-direction*s. With an average of 20% of deviation in the *z-direction* the system stopped converging while only at an average deviation of 30% in the *x-direction* and 30% in the *y-direction* that the same convergence problem occured.

APPLICATION TO FIELD DATA

The field data application was made with the aeromagnetic data of Carajás, Pará, Brazil, provided by the CPRM. The total number of data is 7,706,153. For this case we gridded the data to $500 \times 500 = 2,500,000$ observation points, which means that storing the full sensitivity matrix would be necessary 45.47 Terabytes of RAM. However, taking advantage that the second derivatives of equation 6 are symmetric or skew-symmetric matrices, it is possible to reconstruct the whole sensitivity matrix storing only the first column of each component of equation 10, thus, using only 114.44 Megabytes.

To achieve high efficiency in property estimative of the equivalent sources, the method CGLS for inversion was used, combined with a fast matrix-vector product, only possible because of the Block-Toeplitz Toeplitz-Block(BTTB) structure of the sensitivity matrix. This fast matrix-vector product was also used for data processing (upward continuation) in a very efficient way.

As this area is very large different values of the magnetic main field can be considered. For this test it was considered an approximated mid location of the area (latitude $-6.5°$ and longitude $-50.75°$) where the declination is $-19.86°$ for the IGRF model in 01/01/2014. The inclination was calculated considering the Geocentric axial dipole model ($tang\, I = 2 \times tan\, \lambda$) and is equal to 12.84°. As the source magnetization is unknown, inclination and declination equals to the main field is being used.

Figure **??** shows the observed magnetic field data of the area. Using a equivalent layer at 300 meters above the ground the predicted data and its residual are shown in figure **??**. The mean of -2.7135×10^{-07} nT and the standart deviation of 3.2726×10^{-06} nT of the residual shows the good result of physical property estimative. It was used 50 iterations of

the CGLS method taking 15.5 seconds with a Intel core i7 7700HQ@2.8GHz processor.

In figure **??** the upward continuation transformation was made at 5000 meters and took 0.49 seconds.

CONCLUSIONS

In this work, we were able to develop a fast equivalent layer technique for processing magnetic data with the method of Conjugate Gradient Least Square using the convolutional equivalent layer theory presented in Siqueira et al. (2017) to obtain similar results. The sensitivity matrix of the magnetic equivalent layer carries the structure of BTTB matrices, which means a very low computational cost matrix-vector product and also the possibility to store only the first column of the matrix BCCB. In this work we propose a method to calculate the first six columns of the second derivatives matrices to embbed them into BCCB and then finally arrive in what would be the first column of the BCCB matrix embbeded from the sensitivity matrix.

Synthetic tests showed similar results estimating the physical property using a classical approach to solve a linear system and our method using the CGLS combined with the BTTB matrix-vector product. The difference in time, however, is noticeable: 2.04 seconds using the classical approach and 0.083 seconds using our approach. This difference was obtained with a mid-size mesh of 80×80 points, greater results can be obtained if more observation points are used.

Real data test were also conducted in the region of Carajás, Pará, Brazil. With a regular grid of 2,500,000 observation points, store the full sensitivity matrix it would be necessary 45.47 Terabytes of RAM. However, taking advantage symmetric or skew-symmetric matrices structures, it is possible to reconstruct the whole sensitivity matrix using only 114.44 Megabytes. Using 50 iterations of the CGLS method took 15.5 seconds and very good results of property estimative were obtained. Also the upward continuation transformation was successfull with good results and taking only 0.49 seconds.

ACKNOWLEDGEMENTS

Diego Takahashi was supported by a Phd scholarship from CAPES. Valéria C.F. Barbosa was supported by fellowships from CNPq (grant 307135/2014-4) and FAPERJ (grant 26/202.582/2019). Vanderlei C. Oliveira Jr. was supported by fellowships from CNPq (grant 308945/2017-4) and FAPERJ (grant E-26/202.729/2018). The authors thank the Geological Survey of Brazil (CPRM) for providing the field data.

APPENDIX A

FLOPS COMPUTATIONS

Classical flops count

The flops count of the classical approach to solve the linear system (equation 14) using the Cholesky factorization is given by equation 68. The step-by-step count follows:

(1) $\mathbf{A}^\top\mathbf{A}$: $2N^3$ (one matrix-matrix product).

(2) $\mathbf{A}^\top\mathbf{A}$: $\frac{1}{3}N^3$ (one Cholesky factorization $\mathbf{C_f}$).

(3) $\mathbf{A}^\top\mathbf{d}^o$: $2N^2$ (one matrix-vector product).

(4) $\mathbf{C_f}(\mathbf{A}^\top\mathbf{d}^o)$: $2N^2$ (one matrix-vector product).

(5) $\mathbf{C_f}^\top(\mathbf{C_f}\mathbf{A}^\top\mathbf{d}^o)$: $2N^2$ (one matrix-vector product).

Summing all calculations:

$$f_{classical} = \frac{7}{3}N^3 + 6N^2\,, \tag{A-1}$$

CGLS flops count

The flops count of CGLS algorithm 1 can be summarized as:

Out of the loop:

(1) $\mathbf{A}^\top\mathbf{s}$: $2N^2$ (one matrix-vector product).

Inside the loop:

(1) $\dfrac{\mathbf{r}^{(it)^\top}\mathbf{r}^{(it)}}{\mathbf{r}^{(it-1)^\top}\mathbf{r}^{(it-1)}}$: $4N$ (two vector-vector products).

(2) $\mathbf{r}^{it} - \alpha_{it}\,\beta_{it}\,\mathbf{c}^{(it-1)}$: $2N$ (one scalar-vector product and one vector subtraction).

(3) $\dfrac{{||\mathbf{r}^{(it)}||_2}^2}{(\mathbf{c}^{(it)^\top}\mathbf{A}^\top)(\mathbf{A}\,\mathbf{c}^{(it)})}$: $2N^2 + 2N$ (one matrix-vector and one vector-vector product).

(4) $\hat{\mathbf{p}}^{it} - \alpha_{it}\,\mathbf{c}^{(it)}$: $2N$ (one vector subtraction).

(5) $\mathbf{s}^{it} - \alpha_{it}\,\mathbf{A}\,\mathbf{c}^{(it)}$: $2N$ (one vector subtraction, the matrix-vector product was calculated in step 3).

(6) $\mathbf{A}^\top\,\mathbf{s}^{(it+1)}$: $2N^2$ (one matrix-vector product).

The result of all flops count leads to:

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

Our modified CGLS flops count

All the flops count presented in previous section for the CGLS remains, only substituting the out of the loop matrix-vector product in step 1 and the two matrix-vector products inside the loop in steps 3 and 6. The computations necessary to carry the matrix-vector used in this work are given by:

(1) $\mathbf{L}$: $\kappa\,4N\log_2(4N)$ (one 2D FFT for the eigenvalues calculation of the sensitivity matrix $\mathbf{A}$ or the transposed sensitivity matrix $\mathbf{A}^\top$).

(2) $\mathbf{F}_{2Q}\,\mathbf{V}\,\mathbf{F}_{2P}$: $\kappa\,4N\log_2(4N)$ (one 2D FFT).

(3) $\mathbf{L}\circ(\mathbf{F}_{2Q}\,\mathbf{V}\,\mathbf{F}_{2P})$: $24N$ (one complex Hadamard matrix multiplication).

(4) $\mathbf{F}_{2Q}^{*}\left[\mathbf{L} \circ \left(\mathbf{F}_{2Q}\,\mathbf{V}\,\mathbf{F}_{2P}\right)\right]\mathbf{F}_{2P}^{*}$: $\kappa\, 4N \log_2(4N)$ (one inverse 2D FFT).

Matrix-vector product total: $\kappa\, 12N \log_2(4N) + 24N$.

As matrix $\mathbf{A}$ (equation 12) and its transposed never changes, it is not necessary to calculate the eigenvalues inside the loop at each iteration, we are considering that both are calculated out of the loop. Inside the loop, steps 2 to 4 are repeated two times per iteration. Substituting this result into the CGLS flops count (equation A-2) leads to:

$$f_{ours} = \kappa\, 16N \log_2(4N) + 24N + it\,(\kappa\, 16N \log_2(4N) + 60N). \tag{A-3}$$

REFERENCES

Aster, R. C., B. Borchers, and C. H. Thurber, 2018, Parameter estimation and inverse problems: Elsevier.

Blakely, R. J., 1996, Potential theory in gravity and magnetic applications, 1 ed.: Cambridge University Press.

Chan, R. H.-F., and X.-Q. Jin, 2007, An introduction to iterative Toeplitz solvers: SIAM, **5**.

Dampney, C. N. G., 1969, The equivalent source technique: Geophysics, **34**, no. 1, 39–53.

Emilia, D. A., 1973, Equivalent sources used as an analytic base for processing total magnetic field profiles: Geophysics, **38**, no. 2, 339–348.

Golub, G. H., and C. F. V. Loan, 2013, Matrix computations (johns hopkins studies in the mathematical sciences), 4 ed.: Johns Hopkins University Press.

Hogue, J. D., R. A. Renaut, and S. Vatankhah, 2020, A tutorial and open source software for the efficient evaluation of gravity and magnetic kernels: Computers & Geosciences, **144**, 104575.

Leão, J. W. D., and J. B. C. Silva, 1989, Discrete linear transformations of potential field data: Geophysics, **54**, no. 4, 497–507.

Levinson, N., 1946, The wiener (root mean square) error criterion in filter design and prediction: Journal of Mathematics and Physics, **25**, no. 1-4, 261–278.

Li, Y., M. Nabighian, and D. W. Oldenburg, 2014, Using an equivalent source with positivity for low-latitude reduction to the pole without striation: Geophysics, **79**, J81–J90.

Li, Y., and D. W. Oldenburg, 2010, Rapid construction of equivalent sources using wavelets: Geophysics, **75**, no. 3, L51–L59.

Mendonça, C. A., 2020, Subspace method for solving large-scale equivalent layer and density

mapping problems: Geophysics, **85**, G57–G68.

Mendonça, C. A., and J. B. C. Silva, 1994, The equivalent data concept applied to the interpolation of potential field data: Geophysics, **59**, no. 5, 722–732.

Oliveira Jr., V. C., V. C. F. Barbosa, and L. Uieda, 2013, Polynomial equivalent layer: Geophysics, **78**, no. 1, G1–G13.

Renaut, R. A., J. D. Hogue, S. Vatankhah, and S. Liu, 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, **223**, 1378–1397.

Siqueira, F. C., V. C. Oliveira Jr, and V. C. Barbosa, 2017, Fast iterative equivalent-layer technique for gravity data processing: A method grounded on excess mass constraint: Geophysics, **82**, no. 4, G57–G69.

Takahashi, D., V. C. Oliveira Jr, and V. C. Barbosa, 2020, Convolutional equivalent layer for gravity data processing: Geophysics, **85**, G129–G141.

Van Loan, C., 1992, Computational frameworks for the Fast Fourier Transform: SIAM.

Zhang, Y., and Y. S. Wong, 2015, BTTB-based numerical schemes for three-dimensional gravity field inversion: Geophysical Journal International, **203**, no. 1, 243–256.

Zhang, Y., Y. S. Wong, and Y. Lin, 2016, BTTB–RRCG method for downward continuation of potential field data: Journal of Applied Geophysics, **126**, 74–86.

LIST OF TABLES

N	**Matrix A**	**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 1: This table shows the RAM memory usage (in Megabytes) for storing the whole matrix **A** (equation 12), the sum of all six first columns of the BCCB matrices embedded from the components of the matrix **H** from equation 10 (both need 8 bytes per element) and the matrix **L** containing the eigenvalues complex numbers (16 bytes per element) resulting from the diagonalization of matrix **C** (equation 62). Here we must consider that N observation points forms a $N \times N$ matrix.