frontiers

1

The computation aspects of the equivalent-layer technique: review and perspective

Diego Takahashi [1,*], **André L. A. Reis** [2], **Vanderlei C. Oliveira Jr.** [1] **and Valéria C. F. Barbosa** [1]

[1] *Observatório Nacional, Department of Geophysics, Rio de Janeiro, Brasil*
[2] *Universidade do Estado do Rio de Janeiro, Department of Applied Geology, Rio de Janeiro, Brasil*

Correspondence*:
Valéria C.F. Barbosa
valcris@on.br

1 FUNDAMENTALS

Let $\mathbf{d}$ be a $D \times 1$ vector, whose i-th element d_i is the observed potential field at the position (x_i, y_i, z_i), $i \in \{1 : D\}$, of a topocentric Cartesian system with x, y and z axes pointing to north, east and down, respectively. Consider that d_i can be satisfactorily approximated by a harmonic function

$$f_i = \sum_{j=1}^{P} g_{ij}\, p_j \,, \quad i \in \{1 : D\} \,, \tag{1}$$

where, p_j represents the scalar physical property of a virtual source (i.e., monopole, dipole, prism) located at (x_j, y_j, z_j), $j \in \{1 : P\}$ and

$$g_{ij} \equiv g(x_i - x_j, y_i - y_j, z_i - z_j) \,, \quad z_i < \min\{z_j\} \,, \quad \forall i \in \{1 : D\} \,, \tag{2}$$

is a harmonic function, where $\min\{z_j\}$ denotes the minimum z_j, or the vertical coordinate of the shallowest virtual source. These virtual sources are called *equivalent sources* and they form an *equivalent layer*. In matrix notation, the potential field produced by all equivalent sources at all points (x_i, y_i, z_i), $i \in \{1 : D\}$, is given by:

$$\mathbf{f} = \mathbf{G}\mathbf{p} \,, \tag{3}$$

where $\mathbf{p}$ is a $P \times 1$ vector with j-th element p_j representing the scalar physical property of the j-th equivalent source and $\mathbf{G}$ is a $D \times P$ matrix with element g_{ij} given by equation 2.

The equivalent-layer technique consists in solving a linear inverse problem to determine a parameter vector $\mathbf{p}$ leading to a predicted data vector $\mathbf{f}$ (equation 3) *sufficiently close to* the observed data vector $\mathbf{d}$, whose i-th element d_i is the observed potential field at (x_i, y_i, z_i). The notion of *closeness* is intrinsically related to the concept of *vector norm* (e.g., Golub and Van Loan, 2013, p. 68) or *measure of length* (e.g., Menke, 2018, p. 41). Because of that, almost all methods for determining $\mathbf{p}$ actually estimate a parameter vector $\tilde{\mathbf{p}}$ minimizing a length measure of the difference between $\mathbf{f}$ and $\mathbf{d}$ (see subsection 1.3). Given an estimate $\tilde{\mathbf{p}}$, it is then possible to compute a potential field transformation

$$\mathbf{t} = \mathbf{A}\tilde{\mathbf{p}} \,, \tag{4}$$

where $\mathbf{t}$ is a $T \times 1$ vector with k-th element t_k representing the transformed potential field at the position (x_k, y_y, z_k), $k \in \{1 : T\}$, and

$$a_{kj} \equiv a(x_k - x_j, y_k - y_j, z_k - z_j) \,, \quad z_k < \min\{z_j\} \,, \quad \forall k \in \{1 : T\} \,, \tag{5}$$

is a harmonic function representing the kj-th element of the $T \times P$ matrix $\mathbf{A}$.

1.1 Spatial distribution and total number of equivalent sources

There is no well-established criteria to define the optimum number P or the spatial distribution of the equivalent sources. We know that setting an equivalent layer with more (less) sources than potential-field data usually leads to an underdetermined (overdetermined) inverse problem (e.g., Menke, 2018, p. 52–53). Concerning the spatial distribution of the equivalent sources, the only condition is that they must rely on a surface that is located below and does not cross that containing the potential field data. Soler and Uieda (2021) present a practical discussion about this topic.

From a theoretical point of view, the equivalent layer reproducing a given potential field data set cannot cross the true gravity or magnetic sources. This condition is a consequence of recognizing that the equivalent layer is essentially an indirect solution of a boundary value problem of potential theory (e.g., Roy, 1962; Zidarov, 1965; Dampney, 1969; Li et al., 2014; Reis et al., 2020). In practical applications, however, there is no guarantee that this condition is satisfied. Actually, its is widely known from practical experience (e.g., Gonzalez et al., 2022) that the equivalent-layer technique works even for the case in which the layer cross the true sources.

Regarding the depth of the equivalent layer, Dampney (1969) proposed a criterion based on horizontal data sampling, suggesting that the equivalent-layer depth should be between two and six times the horizontal grid spacing, considering evenly spaced data. However, when dealing with a survey pattern that has unevenly spaced data, Reis et al. (2020) adopted an alternative empirical criterion. According to their proposal, the depth of the equivalent layer should range from two to three times the spacing between adjacent flight lines. The criteria of Dampney (1969) and Reis et al. (2020) are valid for planar equivalent layers. Cordell (1992) have proposed and an alternative criterion for scattered data that leads to an undulating equivalent layer. This criterion have been slightly modified by Guspí et al. (2004), Guspí and Novara (2009) and Soler and Uieda (2021), for example, and consists in setting one equivalent source below each datum at a depth proportional to the horizontal distance to the nearest neighboring data points. Soler and Uieda (2021) have compared different strategies for defining the equivalent sources depth for the specific problem of interpolating gravity data, but they have not found significant differences between them.

1.2 Matrix G

Generally, the harmonic function g_{ij} (equation 2) is defined in terms of the inverse distance between the observation point (x_i, y_i, z_i) and the j-th equivalent source at (x_j, y_j, z_j),

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

or by its partial derivatives of first and second orders, respectively given by

$$\partial_\alpha \frac{1}{r_{ij}} \equiv \frac{-(\alpha_i - \alpha_j)}{r_{ij}^3} \,, \quad \alpha \in \{x, y, z\} \,, \tag{7}$$

and

$$\partial_{\alpha\beta} \frac{1}{r_{ij}} \equiv \begin{cases} \frac{3\,(\alpha_i - \alpha_j)^2}{r_{ij}^5} \,, & \alpha = \beta \,, \\ \frac{3\,(\alpha_i - \alpha_j)\,(\beta_i - \beta_j)}{r_{ij}^5} - \frac{1}{r_{ij}^3} \,, & \alpha \neq \beta \,, \end{cases} \quad \alpha, \beta \in \{x, y, z\} \,. \tag{8}$$

In this case, the equivalent layer is formed by punctual sources representing monopoles or dipoles (e.g., Dampney, 1969; Emilia, 1973; Leão and Silva, 1989; Cordell, 1992; Oliveira Jr. et al., 2013; Siqueira et al., 2017; Reis et al., 2020; Takahashi et al., 2020; Soler and Uieda, 2021; Takahashi et al., 2022). Another common approach consists in not defining g_{ij} by using equations 6–8, but other harmonic functions obtained by integrating them over the volume of regular prisms (e.g., Li and Oldenburg, 2010; Barnes and Lumley, 2011; Li et al., 2014; Jirigalatu and Ebbing, 2019). There are also some less common approaches defining the harmonic function g_{ij} (equation 2) as the potential field due to plane faces with constant physical property (Hansen and Miyazaki, 1984), doublets (Silva, 1986) or by computing the double integration of the inverse distance function with respect to z (Guspí and Novara, 2009).

A common assumption for most of the equivalent-layer methods is that the harmonic function g_{ij} (equation 2) is independent on the actual physical relationship between the observed potential field and their true sources (e.g., Cordell, 1992; Guspí and Novara, 2009; Li et al., 2014). Hence, g_{ij} can be defined according to the problem. The only condition imposed to this function is that it decays to zero as the observation point (x_i, y_i, z_i) goes away from the position (x_j, y_j, z_j) of the j-th equivalent source. However, several methods use a function g_{ij} that preserves the physical relationship between the observed potential field and their true sources. For the case in which the observed potential field is gravity data, g_{ij} is commonly defined as a component of the gravitational field produced at (x_i, y_i, z_i) by a point mass or prism located at (x_j, y_j, z_j), with unit density. On the other hand, g_{ij} is commonly defined as a component of the magnetic induction field produced at (x_i, y_i, z_i) by a dipole or prism located at (x_j, y_j, z_j), with unit magnetization intensity, when the observed potential field is magnetic data.

The main challenge in the equivalent-layer technique is the computational complexity associated with handling large datasets. This complexity arises because the sensitivity matrix $\mathbf{G}$ (equation 3) is dense regardless of the harmonic function g_{ij} (equation 2) employed. In the case of scattered potential-field data, the structure of $\mathbf{G}$ is not well-defined, regardless of the spatial distribution of the equivalent sources. However, in a specific scenario where (i) each potential-field datum is directly associated with a single equivalent source located directly below it, and (ii) both the data and sources are based on planar and regularly spaced grids, Takahashi et al. (2020, 2022) demonstrate that $\mathbf{G}$ exhibits a block-Toeplitz Toeplitz-block (BTTB) structure. In such cases, the product of $\mathbf{G}$ and an arbitrary vector can be efficiently computed using a 2D fast Fourier transform as a discrete convolution.

1.3 General formulation

A general formulation for almost all equivalent-layer methods can be achieved by first considering that the $P \times 1$ parameter vector $\mathbf{p}$ (equation 3) can be reparameterized into a $Q \times 1$ vector $\mathbf{q}$ according to:

$$\mathbf{p} = \mathbf{H}\,\mathbf{q}\,, \tag{9}$$

where $\mathbf{H}$ is a $P \times Q$ matrix. The predicted data vector $\mathbf{f}$ (equation 3) can then be rewritten as follows:

$$\mathbf{f} = \mathbf{G}\,\mathbf{H}\,\mathbf{q}\,. \tag{10}$$

Note that the original parameter vector $\mathbf{p}$ is defined in a P-dimensional space whereas the reparameterized parameter vector $\mathbf{q}$ (equation 9) lies in a Q-dimensional space. For convenience, we use the terms P-space and Q-space to designate them.

In this case, the problem of estimating a parameter vector $\tilde{\mathbf{p}}$ minimizing a length measure of the difference between $\mathbf{f}$ (equation 3) and $\mathbf{d}$ is replaced by that of estimating an auxiliary vector $\tilde{\mathbf{q}}$ minimizing the goal function

$$\Gamma(\mathbf{q}) = \Phi(\mathbf{q}) + \mu\,\Theta(\mathbf{q})\,, \tag{11}$$

which is a combination of particular measures of length given by

$$\Phi(\mathbf{q}) = (\mathbf{d} - \mathbf{f})^\top\,\mathbf{W}_d\,(\mathbf{d} - \mathbf{f})\;, \tag{12}$$

and

$$\Theta(\mathbf{q}) = (\mathbf{q} - \bar{\mathbf{q}})^\top\,\mathbf{W}_q\,(\mathbf{q} - \bar{\mathbf{q}})\;, \tag{13}$$

where the regularization parameter μ is a positive scalar controlling the trade-off between the data-misfit function $\Phi(\mathbf{q})$ and the regularization function $\Theta(\mathbf{q})$; $\mathbf{W}_d$ is a $D \times D$ symmetric matrix defining the relative importance of each observed datum d_i; $\mathbf{W}_q$ is a $Q \times Q$ symmetric matrix imposing prior information on $\mathbf{q}$; and $\bar{\mathbf{q}}$ is a $Q \times 1$ vector of reference values for $\mathbf{q}$ that satisfies

$$\bar{\mathbf{p}} = \mathbf{H}\,\bar{\mathbf{q}}\,, \tag{14}$$

where $\bar{\mathbf{p}}$ is a $P \times 1$ vector containing reference values for the original parameter vector $\mathbf{p}$.

After obtaining an estimate $\tilde{\mathbf{q}}$ for the reparameterized parameter vector $\mathbf{q}$ (equation 9), the estimate $\tilde{\mathbf{p}}$ for the original parameter vector (equation 3) is computed by

$$\tilde{\mathbf{p}} = \mathbf{H}\,\tilde{\mathbf{q}}\,. \tag{15}$$

The reparameterized vector $\tilde{\mathbf{q}}$ is obtained by first computing the gradient of $\Gamma(\mathbf{q})$,

$$\boldsymbol{\nabla}\Gamma(\mathbf{q}) = -2\,\mathbf{H}^\top\mathbf{G}^\top\mathbf{W}_d\,(\mathbf{d} - \mathbf{f}) + 2\,\mu\,\mathbf{W}_q\,(\mathbf{q} - \bar{\mathbf{q}})\;. \tag{16}$$

Then, by considering that $\boldsymbol{\nabla}\Gamma(\tilde{\mathbf{q}}) = \mathbf{0}$ (equation 16), where $\mathbf{0}$ is a vector of zeros, as well as adding and subtracting the term $\left(\mathbf{H}^\top\mathbf{G}^\top\mathbf{W}_d\mathbf{G}\,\mathbf{H}\right)\bar{\mathbf{q}}$, we obtain

$$\tilde{\boldsymbol{\delta}}_q = \mathbf{B}\,\boldsymbol{\delta}_d\,, \tag{17}$$

where

$$\tilde{\boldsymbol{\delta}}_q = \tilde{\mathbf{q}} - \bar{\mathbf{q}}\,, \tag{18}$$

$$\boldsymbol{\delta}_d = \mathbf{d} - \mathbf{G}\,\mathbf{H}\,\bar{\mathbf{q}}\,, \tag{19}$$

$$\mathbf{B} = \left(\mathbf{H}^\top\mathbf{G}^\top\mathbf{W}_d\,\mathbf{G}\,\mathbf{H} + \mu\,\mathbf{W}_q\right)^{-1}\mathbf{H}^\top\mathbf{G}^\top\mathbf{W}_d\,, \tag{20}$$

or, equivalently (Menke, 2018, p. 62),

$$\mathbf{B} = \mathbf{W}_q^{-1}\,\mathbf{H}^\top\mathbf{G}^\top\left(\mathbf{G}\,\mathbf{H}\,\mathbf{W}_q^{-1}\,\mathbf{H}^\top\mathbf{G}^\top + \mu\mathbf{W}_d^{-1}\right)^{-1}\;. \tag{21}$$

Evidently, we have considered that all inverses exist in equations 20 and 21.

The $Q \times D$ matrix $\mathbf{B}$ defined by equation 20 is commonly used for the case in which $D > Q$, i.e., when there are more data than parameters (overdetermined problems). In this case, we consider that the estimate $\tilde{\mathbf{q}}$ is obtained by solving the following linear system for $\tilde{\boldsymbol{\delta}}_q$ (equation 18):

$$\left(\mathbf{H}^\top\mathbf{G}^\top\mathbf{W}_d\,\mathbf{G}\,\mathbf{H} + \mu\,\mathbf{W}_q\right)\tilde{\boldsymbol{\delta}}_q = \mathbf{H}^\top\mathbf{G}^\top\mathbf{W}_d\,\boldsymbol{\delta}_d\,. \tag{22}$$

On the other hand, for the cases in which $D < Q$ (underdetermined problems), matrix $\mathbf{B}$ is usually defined according to equation 21. In this case, the general approach involves estimating $\tilde{\mathbf{q}}$ in two steps. The first consists in solving a linear system for a dummy vector, which is subsequently used to compute $\tilde{\mathbf{q}}$ by a

matrix-vector product as follows:

$$\begin{aligned}\left(\mathbf{G}\,\mathbf{H}\,\mathbf{W}_q^{-1}\,\mathbf{H}^\top\mathbf{G}^\top + \mu\mathbf{W}_d^{-1}\right)\mathbf{u} &= \boldsymbol{\delta}_d \\ \tilde{\boldsymbol{\delta}}_q &= \mathbf{W}_q^{-1}\,\mathbf{H}^\top\mathbf{G}^\top\mathbf{u}\end{aligned}\quad , \tag{23}$$

where $\mathbf{u}$ is a dummy vector. After obtaining $\tilde{\boldsymbol{\delta}}_q$ (equations 22 and 23), the estimate $\tilde{\mathbf{q}}$ is computed with equation 18.

Note that, for the particular case in which $\mathbf{H} = \mathbf{I}_P$ (equation 9), $P = Q$, $\mathbf{p} = \mathbf{q}$, $\bar{\mathbf{p}} = \bar{\mathbf{q}}$ (equation 14) and $\tilde{\mathbf{p}} = \tilde{\mathbf{q}}$ (equation 15). In this case, the linear system (equations 22 and 23) is directly solved for

$$\tilde{\boldsymbol{\delta}}_p = \tilde{\mathbf{p}} - \bar{\mathbf{p}}\,, \tag{24}$$

instead of $\tilde{\boldsymbol{\delta}}_q$ (equation 18).

2 NOTATION FOR SUBVECTORS AND SUBMATRICES

Here, we use a notation inspired on that presented by (Van Loan, 1992, p. 4) to represent subvectors and submatrices. Subvectors of $\mathbf{d}$, for example, are specified by $\mathbf{d}[\mathbf{i}]$, where $\mathbf{i}$ is a list of integer numbers that "pick out" the elements of $\mathbf{d}$ forming the subvector $\mathbf{d}[\mathbf{i}]$. For example, $\mathbf{i} = (1, 6, 4, 6)$ gives the subvector $\mathbf{d}[\mathbf{i}] = [d_1\ \ d_6\ \ d_4\ \ d_6]^\top$. Note that the list $\mathbf{i}$ of indices may be sorted or not and it may also have repeated indices. For the particular case in which the list has a single element $\mathbf{i} = (i)$, then it can be used to extract the i-th element $d_i \equiv \mathbf{d}[i]$ of $\mathbf{d}$. Sequential lists with increment of 1, if the starting index is smaller than the final index, or -1, if the starting index is greater than the final index, can be represented by using the colon notation. For example,

$$\begin{aligned}\mathbf{i} = (3:8) &\Leftrightarrow \mathbf{d}[\mathbf{i}] = [d_3\ \ d_4\ \ \ldots\ \ d_8]^\top \\ \mathbf{i} = (8:3) &\Leftrightarrow \mathbf{d}[\mathbf{i}] = [d_8\ \ d_7\ \ \ldots\ \ d_3]^\top \\ \mathbf{i} = (:8) &\Leftrightarrow \mathbf{d}[\mathbf{i}] = [d_1\ \ d_2\ \ \ldots\ \ d_8]^\top \\ \mathbf{i} = (3:) &\Leftrightarrow \mathbf{d}[\mathbf{i}] = [d_3\ \ d_4\ \ \ldots\ \ d_D]^\top\end{aligned}\quad ,$$

where D is the number of elements forming $\mathbf{d}$.

The notation above can also be used to define submatrices of the $D \times P$ matrix $\mathbf{G}$. For example, $\mathbf{i} = (2, 7, 4, 6)$ and $\mathbf{j} = (1, 3, 8)$ lead to the submatrix

$$\mathbf{G}[\mathbf{i}, \mathbf{j}] = \begin{bmatrix} g_{21} & g_{23} & g_{28} \\ g_{71} & g_{73} & g_{78} \\ g_{41} & g_{43} & g_{48} \\ g_{61} & g_{63} & g_{68} \end{bmatrix}\,.$$

Note that, in this case, the lists $\mathbf{i}$ and $\mathbf{j}$ "pick out", respectively, the rows and columns of $\mathbf{G}$ that form the submatrix $\mathbf{G}[\mathbf{i}, \mathbf{j}]$. The i-th row of $\mathbf{G}$ is given by the $1 \times P$ vector $\mathbf{G}[i, :]$. Similarly, the $D \times 1$ vector $\mathbf{G}[:, j]$

represents the j-th column. Finally, we may use the colon notation to define the following submatrix:

$$\mathbf{i} = (2:5), \mathbf{j} = (3:7) \Leftrightarrow \mathbf{G}[\mathbf{i}, \mathbf{j}] = \begin{bmatrix} g_{23} & g_{24} & g_{25} & g_{26} & g_{27} \\ g_{33} & g_{34} & g_{35} & g_{36} & g_{37} \\ g_{43} & g_{44} & g_{45} & g_{46} & g_{47} \\ g_{53} & g_{54} & g_{55} & g_{56} & g_{57} \end{bmatrix} ,$$

which contains the contiguous elements of $\mathbf{G}$ from rows 2 to 5 and from columns 3 to 7.

3 COMPUTATIONAL STRATEGIES

Here, we review some strategies for reducing the computational cost of equivalent-layer technique. Typically, estimating a parameter vector $\tilde{\mathbf{p}}$ or $\tilde{\mathbf{q}}$ requires to solve a large-scale linear inversion (equations 22 and 23). This, in turn, means to deal with some obstacles concerning large computational cost:

(i) the large computer memory to store large and full matrices;

(ii) the long computation time to multiply a matrix by a vector; and

(iii) the long computation time to solve a large linear system of equations.

Two important factors affecting the efficiency of a given matrix algorithm are the storage and amount of required arithmetic. Here, we quantify this last factor by counting *flops*, which are floating point additions, subtractions, multiplications or divisions (Golub and Van Loan, 2013, p. 12–14). We focus on the overall strategies used by the selected methods.

3.1 Moving window

The initial approach to enhance the computational efficiency of the equivalent-layer technique is commonly denoted *moving window* and involves first splitting the observed data d_i, $i \in \{1:D\}$, into M overlapping subsets (or data windows) formed by D^m data each, $m \in \{1:M\}$. The data inside the m-th window are usually adjacent to each other and have indices defined by an integer list $\mathbf{i}^m$ having D^m elements. The number of data D^m forming the data windows are not necessarily equal to each other. Each data window has a $D^m \times 1$ observed data vector $\mathbf{d}^m \equiv \mathbf{d}[\mathbf{i}^m]$. The second step consists in defining a set of P equivalent sources with scalar physical property p_j, $j \in \{1:P\}$, and also split them into M overlapping subsets (or source windows) formed by P^m data each, $m \in \{1:M\}$. The sources inside the m-th window have indices defined by an integer list $\mathbf{j}^m$ having P^m elements. Each source window has a $P^m \times 1$ parameter vector $\mathbf{p}^m$ and is located right below the corresponding m-th data window. Then, each $\mathbf{d}^m \equiv \mathbf{d}[\mathbf{i}^m]$ is approximated by

$$\mathbf{f}^m = \mathbf{G}^m \mathbf{p}^m , \tag{25}$$

where $\mathbf{G}^m \equiv \mathbf{G}[\mathbf{i}^m, \mathbf{j}^m]$ is a submatrix of $\mathbf{G}$ (equation 3) formed by the elements computed with equation 2 using only the data and equivalent sources located inside the window m-th. The main idea of the moving-window approach is using the $\tilde{\mathbf{p}}^m$ estimated for each window to obtain (i) an estimate $\tilde{\mathbf{p}}$ of the parameter vector for the entire equivalent layer or (ii) a given potential-field transformation $\mathbf{t}$ (equation 4). The main advantages of this approach is that (i) the estimated parameter vector $\tilde{\mathbf{p}}$ or transformed potential field are not obtained by solving the full, but smaller linear systems and (ii) the full matrix $\mathbf{G}$ (equation 3) is never stored.

Leão and Silva (1989) presented a pioneer work using the moving-window approach. Their method requires a regularly-spaced grid of observed data on a horizontal plane z_0. The data windows are defined by square local grids of $\sqrt{D'} \times \sqrt{D'}$ adjacent points, all of them having the same number of points D'. The equivalent sources in the m-th data window are located below the observation plane, at a constant vertical distance Δz_0. They are arranged on a regular grid of $\sqrt{P'} \times \sqrt{P'}$ adjacent points following the same grid pattern of the observed data. The local grid of sources for all data windows have the same number of elements P'. Besides, they are vertically aligned, but expands the limits of their corresponding data windows, so that $D' < P'$. Because of this spatial configuration of observed data and equivalent sources, we have that $\mathbf{G}^m = \mathbf{G}'$ (equation 25) for all data windows (i.e., $\forall\, m \in \{1 : M\}$), where $\mathbf{G}'$ is a $D' \times P'$ constant matrix.

By omitting the normalization strategy used by Leão and Silva (1989), their method consists in directly computing the transformed potential field t_c^m at the central point $(x_c^m, y_c^m, z_0 + \Delta z_0)$ of each data window as follows:

$$t_c^m = \left(\mathbf{a}'\right)^\top \mathbf{B}'\, \mathbf{d}^m \,, \quad m \in \{1 : M\} \,, \tag{26}$$

where $\mathbf{a}'$ is a $P' \times 1$ vector with elements computed by equation 5 by using all equivalent sources in the m-th window and only the coordinate of the central point in the m-th data window and

$$\mathbf{B}' = \left(\mathbf{G}'\right)^\top \left[\mathbf{G}' \left(\mathbf{G}'\right)^\top + \mu\, \mathbf{I}_{D'}\right]^{-1} \tag{27}$$

is a particular case of matrix $\mathbf{B}$ associated with underdetermined problems (equation 21) for the particular case in which $\mathbf{H} = \mathbf{W}_q = \mathbf{I}_{P'}$ (equations 9 and 13), $\mathbf{W}_d = \mathbf{I}_{D'}$ (equation 12), $\bar{\mathbf{p}} = \mathbf{0}$ (equation 14), where $\mathbf{I}_{P'}$ and $\mathbf{I}_{D'}$ are identity matrices of order P' and D', respectively, and $\mathbf{0}$ is a vector of zeros. Due to the presumed spatial configuration of the observed data and equivalent sources, $\mathbf{a}'$ and $\mathbf{G}'$ are the same for all data windows. Hence, only the data vector $\mathbf{d}^m$ is modified according to the position of the data window. Note that equation 26 combines the potential-field transformation (equation 4) with the solution of the undetermined problem (equation 23).

The method proposed by Leão and Silva (1989) can be outlined by the Algorithm 1. Note that Leão and Silva (1989) directly compute the transformed potential t_c^m at the central point of each data window without explicitly computing and storing an estimated for $\mathbf{p}^m$ (equation 25). It means that their method allows computing a single potential-field transformation. A different transformation or the same one evaluated at different points require running their moving-data window method again.

Soler and Uieda (2021) generalized the method proposed by Leão and Silva (1989) for irregularly spaced data on an undulating surface. A direct consequence of this generalization is that a different submatrix $\mathbf{G}^m \equiv \mathbf{G}[\mathbf{i}^m, \mathbf{j}^m]$ (equation 25) must be computed for each window. Differently from Leão and Silva (1989), Soler and Uieda (2021) store the computed $\tilde{\mathbf{p}}^m$ for all windows and subsequently use them to obtain a desired potential-field transformation (equation 4) as the superposed effect of all windows. The estimated $\tilde{\mathbf{p}}^m$ for all windows are combined to form a single $P \times 1$ vector $\tilde{\mathbf{p}}$, which is an estimate for original parameter vector $\mathbf{p}$ (equation 3). For each data window, Soler and Uieda (2021) solve an overdetermined problem (equation 22) for $\tilde{\mathbf{p}}^m$ by using $\mathbf{H} = \mathbf{W}_q = \mathbf{I}_{P^m}$ (equations 9 and 13), $\mathbf{W}_d^m$ (equation 12) equal to a diagonal matrix of weights for the data inside the m-th window and $\bar{p} = \mathbf{0}$ (equation 14), so that

$$\left[(\mathbf{G}^m)^\top\, \mathbf{W}_d^m\, \mathbf{G}^m + \mu\, \mathbf{I}_{P'}\right] \tilde{\mathbf{p}}^m = (\mathbf{G}^m)^\top\, \mathbf{W}_d^m\, \mathbf{d}^m \,. \tag{28}$$

Algorithm 1: Generic pseudo-code for the method proposed by Leão and Silva (1989).

Initialization :

Set the indices $\mathbf{i}^m$ for each data window, $m \in \{1 : M\}$;
Set the indices $\mathbf{j}^m$ for each source window, $m \in \{1 : M\}$;
Set the constant depth $z_0 + \Delta z_0$ for all equivalent sources ;
Compute the vector $\mathbf{a}'$ associated with the desired potential-field transformation ;
Compute the matrix $\mathbf{G}'$;
Compute the matrix $\mathbf{B}'$ (equation 27) ;
Compute the vector $(\mathbf{a}')^\top \mathbf{B}'$;
$m = 1$;
while $m < M$ **do**
    Compute t_c^m (equation 26) ;
    $m \leftarrow m + 1$;
end

Unlike Leão and Silva (1989), Soler and Uieda (2021) do not adopt a sequential order of the data windows; rather, they adopt a randomized order of windows in their iterations. The overall steps of the method proposed by Soler and Uieda (2021) are defined by the Algorithm 2. For convenience, we have omitted the details about the randomized window order and the normalization strategy employed by Soler and Uieda (2021). Note that this algorithm starts with a residuals vector $\mathbf{r}$ that is iteratively updated. The iterative algorithm in Soler and Uieda (2021) estimates a solution ($\tilde{\mathbf{p}}^m$ in equation 28) using the data and the equivalent sources that fall within a moving-data window; however, it calculates the predicted data and the residual data in the whole survey data. Next, the residual data that fall within a new position of the data window is used as input data to estimate a new solution within the data window which, in turn, is used to calculated a new predicted data and a new residual data in the whole survey data. Regarding the equivalent-source layout, Soler and Uieda (2021) proposed the block-averaged sources locations in which the survey area is divided into horizontal blocks and one single equivalent source is assigned to each block. Each single source per block is placed over the layer with its horizontal coordinates given by the average horizontal positions of observation points. According to Soler and Uieda (2021), the block-averaged sources layout may prevent aliasing of the interpolated values, specially when the observations are unevenly sampled. This strategy also reduces the number of equivalent sources without affecting the accuracy of the potential-field interpolation. This reduction reduces the computational load for estimating the physical property on the equivalent layer. This block-averaged sources layout is not included in the Algorithm 2.

3.2 Column-action update

We call the computational strategy *column-action update* because a single source is used to calculate the predicted data and the residual data in the whole survey data. Hence, a single column of the sensitivity matrix $\mathbf{G}$ (equation 3) is calculated iteratively.

Cordell (1992) proposed a computational strategy that was later used by Guspí and Novara (2009) and relies on first defining one equivalent source located right below each observed data d_i, $i \in \{1 : D\}$, at a vertical coordinate $z_i + \Delta z_i$, where Δz_i is proportional to the distance from the i-th observation point (x_i, y_i, z_i) to its closest neighbor. The second step consists in updating the physical property p_j of a single equivalent source, $j \in \{1 : D\}$ and remove its predicted potential field from the observed data vector $\mathbf{d}$, producing a residuals vector $\mathbf{r}$. At each iteration, the single equivalent source is the one located vertically beneath the observation station of the maximum data residual. Next, the predicted data produced by this

Algorithm 2: Generic pseudo-code for the method proposed by Soler and Uieda (2021).

Initialization :
Set the indices $\mathbf{i}^m$ for each data window, $m \in \{1 : M\}$;
Set the indices $\mathbf{j}^m$ for each source window, $m \in \{1 : M\}$;
Set the depth of all equivalent sources ;
Set a $D \times 1$ residuals vector $\mathbf{r} = \mathbf{d}$;
Set a $P \times 1$ vector $\tilde{\mathbf{p}} = \mathbf{0}$;
$m = 1$;
while $m < M$ **do**
    Set the matrix $\mathbf{W}_d^m$;
    Compute the matrix $\mathbf{G}^m$;
    Compute $\tilde{\mathbf{p}}^m$ (equation 28) ;
    $\tilde{\mathbf{p}}[\mathbf{j}^m] \leftarrow \tilde{\mathbf{p}}[\mathbf{j}^m] + \tilde{\mathbf{p}}^m$;
    $\mathbf{r} \leftarrow \mathbf{r} - \mathbf{G}[:, \mathbf{j}^m]\, \tilde{\mathbf{p}}^m$;
    $m \leftarrow m + 1$;
end

single source is calculated over all of the observation points and a new data residual $\mathbf{r}$ and the $D \times 1$ parameter vector $\mathbf{p}$ containing the physical property of all equivalent sources are updated iteratively. During each subsequent iteration, Cordell's method either incorporates a single equivalent source or adjusts an existing equivalent source to match the maximum amplitude of the current residual field. The convergence occurs when all of the residuals are bounded by an envelope of prespecified expected error. At the end, the algorithm produces an estimate $\tilde{\mathbf{p}}$ for the parameter vector yielding a predicted potential field $\mathbf{f}$ (equation 3) satisfactorily fitting the observed data $\mathbf{d}$ according to a given criterion. Note that the method proposed by Cordell (1992) iteratively solves the linear $\mathbf{G}\tilde{\mathbf{p}} \approx \mathbf{d}$ with a $D \times D$ matrix $\mathbf{G}$. At each iteration, only a single column of $\mathbf{G}$ (equation 3) is used. An advantage of this *column-action update approach* is that the full matrix $\mathbf{G}$ is never stored.

Algorithm 3 delineates the Cordell's method. Note that a single column $\mathbf{G}[:, i_{\texttt{max}}]$ of the $D \times D$ matrix $\mathbf{G}$ (equation 3) is used per iteration, where $i_{\texttt{max}}$ is the index of the maximum absolute value in $\mathbf{r}$. As pointed out by Cordell (1992), the method does not necessarily decrease monotonically along the iterations. Besides, the method may not converge depending on how the vertical distances Δz_i, $i \in \{1 : D\}$, controlling the depths of the equivalent sources are set. According to Cordell (1992), the maximum absolute value $r_{\texttt{max}}$ in $\mathbf{r}$ decreases robustly at the beginning and oscillates within a narrowing envelope for the subsequent iterations.

Guspí and Novara (2009) generalized Cordell's method to perform reduction to the pole and other transformations on scattered magnetic observations by using two steps. The first step involves computing the vertical component of the observed field using equivalent sources while preserving the magnetization direction. In the second step, the vertical observation direction is maintained, but the magnetization direction is shifted to the vertical. The main idea employed by both Cordell (1992) and Guspí and Novara (2009) is an iterative scheme that uses a single equivalent source positioned below a measurement station to compute both the predicted data and residual data for all stations. This approach entails a computational strategy where a single column of the sensitivity matrix $\mathbf{G}$ (equation 3) is calculated per iteration.

3.3 Row-action update

We call the computational strategy *row-action update* because a single row of the sensitivity matrix $\mathbf{G}$ (equation 3) is calculated iteratively. Hence, the equivalent-layer solution is updated by processing a

Algorithm 3: Generic pseudo-code for the method proposed by Cordell (1992).

Initialization :

Compute a $D \times 1$ vector $\mathbf{\Delta z}$ whose i-th element Δz_i is a vertical distance controlling the depth of the i-th equivalent source, $i \in \{1 : D\}$;
Set a tolerance ϵ ;
Set a maximum number of iterations `ITMAX` ;
Set a $D \times 1$ residuals vector $\mathbf{r} = \mathbf{d}$;
Set a $D \times 1$ vector $\tilde{\mathbf{p}} = \mathbf{0}$;
Define the maximum absolute value $r_{\texttt{max}}$ in $\mathbf{r}$;
$m = 1$;
while $(r_{\texttt{max}} > \epsilon)$ **and** $(m < \texttt{ITMAX})$ **do**
    Define the coordinates $(x_{\texttt{max}}, y_{\texttt{max}}, z_{\texttt{max}})$ and index $i_{\texttt{max}}$ of the observation point associated with $r_{\texttt{max}}$;
    $\tilde{\mathbf{p}}[i_{\texttt{max}}] \leftarrow \tilde{\mathbf{p}}[i_{\texttt{max}}] + (r_{\texttt{max}}\, \mathbf{\Delta z}[i_{\texttt{max}}])$;
    $\mathbf{r} \leftarrow \mathbf{r} - (\mathbf{G}[:, i_{\texttt{max}}]\, \tilde{\mathbf{p}}[i_{\texttt{max}}])$;
    Define the new $r_{\texttt{max}}$ in $\mathbf{r}$;
    $m \leftarrow m + 1$;
end

new datum (one matrix row) at each iteration. To reduce the total processing time and memory usage of equivalent-layer technique, Mendonça and Silva (1994) proposed a strategy called *equivalent data concept*. The equivalent data concept is grounded on the principle that there is a subset of redundant data that does not contribute to the final solution and thus can be dispensed. Conversely, there is a subset of observations, called equivalent data, that contributes effectively to the final solution and fits the remaining observations (redundant data). Iteratively, Mendonça and Silva (1994) selected the subset of equivalent data that is substantially smaller than the original dataset. This selection is carried out by incorporating one data point at a time.

Mendonça and Silva (1994) proposes an algebraic reconstruction technique (ART) (e.g., van der Sluis and van der Vorst, 1987, p. 58) to estimate a parameter vector $\tilde{\mathbf{p}}$ for a regular grid of P equivalent sources on a horizontal plane z_0. Such methods iterate on the linear system rows to estimate corrections for the parameter vector, which may substantially save computer time and memory required to compute and store the full linear system matrix along the iterations. The convergence of such *row-update methods* depends on the linear system condition. The main advantage of such methods is not computing and storing the full linear system matrix, but iteratively using its rows. In contrast to ART-type algorithms, the rows in Mendonça and Silva (1994) are not processed sequentially. Instead, in Mendonça and Silva (1994), the rows are introduced according to their residual magnitudes (maximum absolute value in $\mathbf{r}$), which are computed based on the estimate over the equivalent layer from the previous iteration. The particular ART method proposed by Mendonça and Silva (1994) considers that

$$\mathbf{d} = \begin{bmatrix} \mathbf{d}_e \\ \mathbf{d}_r \end{bmatrix} \quad , \quad \mathbf{G} = \begin{bmatrix} \mathbf{G}_e \\ \mathbf{R}_r \end{bmatrix} \quad , \tag{29}$$

where $\mathbf{d}_e$ and $\mathbf{d}_r$ are $D_e \times 1$ and $D_r \times 1$ vectors and $\mathbf{G}_e$ and $\mathbf{G}_r$ are $D_e \times P$ and $D_r \times P$ matrices, respectively. Mendonça and Silva (1994) designate $\mathbf{d}_e$ and $\mathbf{d}_r$ as, respectively, *equivalent* and *redundant* data. With the exception of a normalization strategy, Mendonça and Silva (1994) calculate a $P \times 1$ estimated parameter vector $\tilde{\mathbf{p}}$ by solving an underdetermined problem (equation 23) involving only the equivalent data $\mathbf{d}_e$ (equation 29) for the particular case in which $\mathbf{H} = \mathbf{W}_p = \mathbf{I}_P$ (equations 9 and 13), $\mathbf{W}_d = \mathbf{I}_{D_e}$

(equation 12) and $\bar{p} = \mathbf{0}$ (equation 14), which results in

$$\begin{aligned} (\mathbf{F} + \mu\,\mathbf{I}_{D_e})\,\mathbf{u} &= \mathbf{d}_e \\ \tilde{\mathbf{p}} &= \mathbf{G}_e^{\top}\mathbf{u} \end{aligned} \quad , \tag{30}$$

where $\mathbf{F}$ is a computationally-efficient $D_e \times D_e$ matrix that approximates $\mathbf{G}_e\,\mathbf{G}_e^{\top}$. Mendonça and Silva (1994) presume that the estimated parameter vector $\tilde{\mathbf{p}}$ obtained from equation 30 leads to a $D_r \times 1$ residuals vector

$$\mathbf{r} = \mathbf{d}_r - \mathbf{G}_r\tilde{\mathbf{p}} \tag{31}$$

having a maximum absolute value $r_{\texttt{max}} \leq \epsilon$, where ϵ is a predefined tolerance.

The overall method of Mendonça and Silva (1994) is defined by Algorithm 4. It is important noting that the number D_e of equivalent data in $\mathbf{d}_e$ increases by one per iteration, which means that the order of the linear system in equation 30 also increases by one at each iteration. Those authors also propose a computational strategy based on Cholesky factorization (e.g., Golub and Van Loan, 2013, p. 163) for efficiently updating $(\mathbf{F} + \mu\,\mathbf{I}_{D_e})$ at a given iteration (line 16 in Algorithm 4) by computing only its new elements with respect to those computed in the previous iteration.

Algorithm 4: Generic pseudo-code for the method proposed by Mendonça and Silva (1994).

Initialization :

1 Set a regular grid of P equivalent sources at a horizontal plane z_0 ;
2 Set a tolerance ϵ ;
3 Set a $D \times 1$ residuals vector $\mathbf{r} = \mathbf{d}$;
4 Define the maximum absolute value $r_{\texttt{max}}$ in $\mathbf{r}$;
5 Define the index $i_{\texttt{max}}$ of $r_{\texttt{max}}$;
6 Define the list of indices $\mathbf{i}_r$ of the remaining data in $\mathbf{r}$;
7 Define $\mathbf{d}_e = \mathbf{d}[i_{\texttt{max}}]$;
8 Compute $(\mathbf{F} + \mu\,\mathbf{I}_{D_e})$ and $\mathbf{G}_e$;
9 Compute $\tilde{\mathbf{p}}$ (equation 30) ;
10 Compute $\mathbf{r} = \mathbf{d}[\mathbf{i}_r] - \mathbf{G}[\mathbf{i}_r, :]\,\tilde{\mathbf{p}}$;
11 Define the maximum absolute value $r_{\texttt{max}}$ in $\mathbf{r}$;
12 **while** $(r_{\texttt{max}} > \epsilon)$ **do**
13 Define the index $i_{\texttt{max}}$ of $r_{\texttt{max}}$;
14 Define the list of indices $\mathbf{i}_r$ of the remaining elements in $\mathbf{r}$;
15 $\mathbf{d}_e \leftarrow \begin{bmatrix} \mathbf{d}_e \\ \mathbf{d}[i_{\texttt{max}}] \end{bmatrix}$;
16 Update $(\mathbf{F} + \mu\,\mathbf{I}_{D_e})$ and $\mathbf{G}_e$;
17 Update $\tilde{\mathbf{p}}$ (equation 30) ;
18 Update $\mathbf{r} = \mathbf{d}[\mathbf{i}_r] - \mathbf{G}[\mathbf{i}_r, :]\,\tilde{\mathbf{p}}$;
19 Define the maximum absolute value $r_{\texttt{max}}$ in $\mathbf{r}$;
20 **end**

3.4 Reparameterization

Another approach for improving the computational performance of equivalent-layer technique consists in setting a $P \times Q$ reparameterization matrix $\mathbf{H}$ (equation 9) with $Q << P$. This strategy has been used in applied geophysics for decades (e.g., Skilling and Bryan, 1984; Kennett et al., 1988; Oldenburg et al., 1993; Barbosa et al., 1997) and is known as *subspace method*. The main idea relies in reducing the linear

system dimension from the original P-space to a lower-dimensional subspace (the Q-space). An estimate $\tilde{\mathbf{q}}$ for the reparameterized parameter vector $\mathbf{q}$ is obtained in the Q-space and subsequently used to obtain an estimate $\tilde{\mathbf{p}}$ for the parameter vector $\mathbf{p}$ (equation 3) in the P-space by using equation 9. Hence, the key aspect of this *reparameterization approach* is solving an appreciably smaller linear inverse problem for $\tilde{\mathbf{q}}$ than that for the original parameter vector $\tilde{\mathbf{p}}$ (equation 3).

Oliveira Jr. et al. (2013) have used this approach to describe the physical property distribution on the equivalent layer in terms of piecewise bivariate polynomials. Specifically, their method consists in splitting a regular grid of equivalent sources into source windows inside which the physical-property distribution is described by bivariate polynomial functions. The key aspect of their method relies on the fact that the total number of coefficients required to define the bivariate polynomials is considerably smaller than the original number of equivalent sources. Hence, they formulate a linear inverse problem for estimating the polynomial coefficients and use them later to compute the physical property distribution on the equivalent layer.

The method proposed by Oliveira Jr. et al. (2013) consists in solving an overdetermined problem (equation 22) for estimating the polynomial coefficients $\tilde{\mathbf{q}}$ with $\mathbf{W}_d = \mathbf{I}_D$ (equation 12) and $\bar{q} = \mathbf{0}$ (equation 14), so that

$$\left(\mathbf{H}^\top \mathbf{G}^\top \mathbf{G}\, \mathbf{H} + \mu\, \mathbf{W}_q\right) \tilde{\mathbf{q}} = \mathbf{H}^\top \mathbf{G}^\top\, \mathbf{d}\,, \tag{32}$$

where $\mathbf{W}_q = \mathbf{H}^\top \mathbf{W}_p\, \mathbf{H}$ is defined by a matrix $\mathbf{W}_p$ representing the zeroth- and first-order Tikhonov regularization (e.g., Aster et al., 2019, p. 103). Note that, in this case, the prior information is defined in the P-space for the original parameter vector $\mathbf{p}$ and then transformed to the Q-space. Another characteristic of their method is that it is valid for processing irregularly-spaced data on an undulating surface.

Mendonça (2020) also proposed a reparameterization approach for the equivalent-layer technique. Their approach, however, consists in setting $\mathbf{H}$ as a truncated singular value decomposition (SVD) (e.g., Aster et al., 2019, p. 55) of the observed potential field. Differently from Oliveira Jr. et al. (2013), however, the method of Mendonça (2020) requires a regular grid of potential-field data on horizontal plane. Another difference is that these authors uses $\mathbf{W}_q = \mathbf{I}_Q$ (equation 13), which means that the regularization is defined directly in the Q-space.

Before Oliveira Jr. et al. (2013) and Mendonça (2020), Barnes and Lumley (2011) also proposed a computationally efficient method for equivalent-layer technique based on reparameterization. A key difference, however, is that Barnes and Lumley (2011) did not set a $P \times Q$ reparameterization matrix $\mathbf{H}$ (equation 9) with $Q << P$. Instead, they used a matrix $\mathbf{H}$ with $Q \approx 1.7\, P$. Their central idea is setting a reparameterization scheme that groups distant equivalent sources into blocks by using a bisection process. This scheme leads to a quadtree representation of the physical-property distribution on the equivalent layer, so that matrix $\mathbf{GH}$ (equation 10) is notably sparse. Barnes and Lumley (2011) explore this sparsity in solving the overdetermined problem for $\tilde{\mathbf{q}}$ (equation 32) via conjugate-gradient method (e.g., Golub and Van Loan, 2013, sec. 11.3).

3.5 Wavelet compression

Previously to Barnes and Lumley (2011), the idea of transforming the dense matrix $\mathbf{G}$ (equation 3) into a sparse one has already been used in the context of equivalent-layer technique. Li and Oldenburg (2010) proposed a method that applies the discrete wavelet transform to introduce sparsity into the original dense matrix $\mathbf{G}$. Those authors approximate a planar grid of potential-field data by a regularly-spaced grid of equivalent sources, so that the number of data D and sources P is the same , i.e., $D = P$. Specifically, Li

and Oldenburg (2010) proposed a method that applies the wavelet transform to the original dense matrix $\mathbf{G}$ and sets to zero the small coefficients that are below a given threshold, which results in an approximating sparse representation of $\mathbf{G}$ in the wavelet domain. They first consider the following approximation

$$\mathbf{d}_w \approx \mathbf{G}_s \, \mathbf{p}_w \, , \tag{33}$$

where

$$\mathbf{d}_w = \boldsymbol{\mathcal{W}} \, \mathbf{d} \, , \quad \mathbf{p}_w = \boldsymbol{\mathcal{W}} \, \mathbf{p} \, , \tag{34}$$

are the observed data and parameter vector in the wavelet domain; $\boldsymbol{\mathcal{W}}$ is a $D \times D$ orthogonal matrix defining a discrete wavelet transform; and $\mathbf{G}_s$ is a sparse matrix obtained by setting to zero the elements of

$$\mathbf{G}_w = \boldsymbol{\mathcal{W}} \, \mathbf{G} \, \boldsymbol{\mathcal{W}}^\top \tag{35}$$

with absolute value smaller than a given threshold.

Li and Oldenburg (2010) solve a normalized inverse problem in the wavelet domain. Specifically, they first define a matrix

$$\mathbf{G}_L = \mathbf{G}_s \, \mathbf{L}^{-1} \tag{36}$$

and a normalized parameter vector

$$\mathbf{p}_L = \mathbf{L} \, \mathbf{p}_w \, , \tag{37}$$

where $\mathbf{L}$ is a diagonal and invertible matrix representing an approximation of the first-order Tikhonov regularization in the wavelet domain. Then they solve an overdetermined problem (equation 22) to obtain an estimate $\tilde{\mathbf{p}}_L$ for $\mathbf{p}_L$ (equation 37), with $\mathbf{G}_L$ (equation 36), $\mathbf{H} = \mathbf{I}_P$ (equations 9), $\mu = 0$ (equation 11), $\mathbf{W}_d = \mathbf{I}_D$ (equation 12) and $\bar{p} = \mathbf{0}$ (equation 14) via conjugate-gradient method (e.g., Golub and Van Loan, 2013, sec. 11.3). Finally, Li and Oldenburg (2010) compute an estimate $\tilde{\mathbf{p}}$ for the original parameter vector given by

$$\tilde{\mathbf{p}} = \boldsymbol{\mathcal{W}}^\top \left(\mathbf{L}^{-1} \, \tilde{\mathbf{p}}_L \right) \, , \tag{38}$$

where the term within parenthesis is an estimate $\tilde{\mathbf{p}}_w$ of the parameter vector $\mathbf{p}_w$ (equation 34) in the wavelet domain and matrix $\boldsymbol{\mathcal{W}}^\top$ represents an inverse wavelet transform.

3.6 Iterative methods using the full matrix G

Xia and Sprowl (1991) introduced an iterative method for estimating the parameter vector $\tilde{\mathbf{p}}$ (equation 3), which was subsequently adapted to the Fourier domain by Xia et al. (1993). Their method uses the full and dense sensitivity matrix $\mathbf{G}$ (equation 3) (without applying any compression or reparameterization, for example) to compute the predicted data at all observation points per iteration. More than two decades later, Siqueira et al. (2017) have proposed an iterative method similar to that presented by Xia and Sprowl (1991). The difference is that Siqueira et al.'s algorithm was deduced from the *Gauss' theorem* (e.g., Kellogg, 1967, p. 43) and the *total excess of mass* (e.g., Blakely, 1996, p. 60). Besides, Siqueira et al. (2017) have included a numerical analysis showing that their method produces very stable solutions, even for noise-corrupted potential-field data.

The iterative method proposed by Siqueira et al. (2017) is outlined in Algorithm 5, presumes an equivalent layer formed by monopoles (point masses) and can be applied to irregularly-spaced data on an undulating surface. Note that the residuals $\mathbf{r}$ are used to compute a correction $\boldsymbol{\Delta}\mathbf{p}$ for the parameter vector at each iteration (line 11), which requires a matrix-vector product involving the full matrix $\mathbf{G}$. Interestingly, this

approach for estimating the physical property distribution on an equivalent layer is the same originally proposed by Bott (1960) for estimating the basement relief under sedimentary basins. The methods of Xia and Sprowl (1991) and Siqueira et al. (2017) were originally proposed for processing gravity data, but can be potentially applied to any harmonic function because they actually represent iterative solutions of the classical *Dirichlet's problem* or the *first boundary value problem of potential theory* (Kellogg, 1967, p. 236) on a plane.

Recently, Jirigalatu and Ebbing (2019) presented another iterative method for estimating a parameter vector $\tilde{\mathbf{p}}$ (equation 3). With the purpose of combining different potential-field data, their method basically modifies that shown in Algorithm 5 by changing the initial approximation and the iterative correction for the parameter vector. Specifically, Jirigalatu and Ebbing (2019) replace line 4 by $\tilde{\mathbf{p}} = \mathbf{0}$, where $\mathbf{0}$ is a vector of zeros, and line 9 by $\boldsymbol{\Delta p} = \omega\, \mathbf{G}^\top \mathbf{r}$, where ω is a positive scalar defined by trial and error. Note that this modified approach requires two matrix-vector products involving the full matrix $\mathbf{G}$ per iteration. To overcome the high computational cost of these two products, Jirigalatu and Ebbing (2019) set an equivalent layer formed by prisms and compute their predicted potential field in the wavenumber domain by using the Gauss-FFT technique Zhao et al. (2018).

Algorithm 5: Generic pseudo-code for the iterative method proposed by Siqueira et al. (2017). The symbol "$\circ$" denotes the entrywise or Hadamard product (e.g., Horn and Johnson, 1991, p. 298) and $\boldsymbol{\sigma}$ is a $P \times 1$ vector whose j-th element is the ratio of a predefined element of area centered at the j-th equivalent source and the term $2\pi\gamma$, where γ is the gravitational constant.

Initialization :
1 Set P equivalent sources on a horizontal plane z_0 ;
2 Set a tolerance ϵ ;
3 Set an auxiliary vector $\boldsymbol{\sigma}$;
4 Compute $\tilde{\mathbf{p}} = \boldsymbol{\sigma} \circ \mathbf{d}$;
5 Compute $\mathbf{G}$ (equation 3) ;
6 Compute $\mathbf{r} = \mathbf{d} - \mathbf{G}\,\tilde{\mathbf{p}}$;
7 Compute $\delta = \|\mathbf{r}\|/D$;
8 **while** $(\delta > \epsilon)$ **do**
9 Compute $\boldsymbol{\Delta p} = \boldsymbol{\sigma} \circ \mathbf{r}$;
10 Update $\tilde{\mathbf{p}} \leftarrow \tilde{\mathbf{p}} + \boldsymbol{\Delta p}$;
11 Compute $\boldsymbol{\nu} = \mathbf{G}\,\boldsymbol{\Delta p}$;
12 Update $\mathbf{r} \leftarrow \mathbf{r} - \boldsymbol{\nu}$;
13 Compute $\delta = \|\boldsymbol{\nu}\|/D$;
14 **end**

3.7 Iterative deconvolution

Recently, Takahashi et al. (2020, 2022) proposed the *convolutional equivalent-layer method*, which explores the structure of the sensitivity matrix $\mathbf{G}$ (equation 3) for the particular case in which (i) there is a single equivalent source right below each potential-field datum and (ii) both data and sources rely on planar and regularly spaced grids. Specifically, they consider a regular grid of D potential-field data at points (x_i, y_i, z_0), $i \in \{1 : D\}$, on a horizontal plane z_0. The data indices i may be ordered along the $x-$ or $y-$direction, which results in an $x-$ or $y-$oriented grid, respectively. They also consider a single equivalent source located right below each datum, at a constant vertical coordinate $z_0 + \Delta z$, $\Delta z > 0$. In this case, the number of data and equivalent sources are equal to each other (i.e., $D = P$) and $\mathbf{G}$ (equation 3) assumes a *doubly block Toeplitz* (Jain, 1989, p. 28) or *block-Toeplitz Toeplitz-block* (BTTB) (Chan and

Jin, 2007, p. 67) structure formed by $N_B \times N_B$ blocks, where each block has $N_b \times N_b$ elements, with $D = N_B\, N_b$. This particular structure allows formulating the product of $\mathbf{G}$ and an arbitrary vector as a *fast 2D discrete convolution* via *Fast Fourier Transform* (FFT) (Van Loan, 1992, section 4.2).

Consider, for example, the particular case in which $N_B = 4$, $N_b = 3$ and $D = 12$. In this case, $\mathbf{G}$ (equation 3) is a 12×12 block matrix given by

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}^0 & \mathbf{G}^1 & \mathbf{G}^2 & \mathbf{G}^3 \\ \mathbf{G}^{-1} & \mathbf{G}^0 & \mathbf{G}^1 & \mathbf{G}^2 \\ \mathbf{G}^{-2} & \mathbf{G}^{-1} & \mathbf{G}^0 & \mathbf{G}^1 \\ \mathbf{G}^{-3} & \mathbf{G}^{-2} & \mathbf{G}^{-1} & \mathbf{G}^0 \end{bmatrix}_{D \times D} , \tag{39}$$

where each block $\mathbf{G}^n$, $n \in \{(1 - N_B) : (N_B - 1)\}$, is a 3×3 Toeplitz matrix. Takahashi et al. (2020, 2022) have deduced the specific relationship between blocks $\mathbf{G}^n$ and $\mathbf{G}^{-n}$ and also between a given block $\mathbf{G}^n$ and its transposed $(\mathbf{G}^n)^\top$ according to the harmonic function g_{ij} (equation 2) defining the element ij of the sensitivity matrix $\mathbf{G}$ (equation 3) and the orientation of the data grid.

Consider the matrix-vector products

$$\mathbf{G}\,\mathbf{v} = \mathbf{w} \tag{40}$$

and

$$\mathbf{G}^\top\,\mathbf{v} = \mathbf{w}\,, \tag{41}$$

involving a $D \times D$ sensitivity matrix $\mathbf{G}$ (equation 3) defined in terms of a given harmonic function g_{ij} (equation 2), where

$$\mathbf{v} = \begin{bmatrix} \mathbf{v}^0 \\ \vdots \\ \mathbf{v}^{N_B-1} \end{bmatrix}_{D \times 1} , \quad \mathbf{w} = \begin{bmatrix} \mathbf{w}^0 \\ \vdots \\ \mathbf{w}^{N_B-1} \end{bmatrix}_{D \times 1} , \tag{42}$$

are arbitrary partitioned vectors formed by N_B sub-vectors $\mathbf{v}^n$ and $\mathbf{w}^n$, $n \in \{0 : (N_B - 1)\}$, all of them having N_b elements. Equations 40 and 41 can be computed in terms of an auxiliary matrix-vector product

$$\mathbf{G}_c\,\mathbf{v}_c = \mathbf{w}_c\,, \tag{43}$$

where

$$\mathbf{v}_c = \begin{bmatrix} \mathbf{v}_c^0 \\ \vdots \\ \mathbf{v}_c^{N_B-1} \\ \mathbf{0} \end{bmatrix}_{4D \times 1} , \quad \mathbf{w}_c = \begin{bmatrix} \mathbf{w}_c^0 \\ \vdots \\ \mathbf{w}_c^{N_B-1} \\ \mathbf{0} \end{bmatrix}_{4D \times 1} , \tag{44}$$

are partitioned vectors formed by $2N_b \times 1$ sub-vectors

$$\mathbf{v}_c^n = \begin{bmatrix} \mathbf{v}^n \\ \mathbf{0} \end{bmatrix}_{2N_b \times 1} , \quad \mathbf{w}_c^n = \begin{bmatrix} \mathbf{w}^n \\ \mathbf{0} \end{bmatrix}_{2N_b \times 1} , \tag{45}$$

and $\mathbf{G}_c$ is a $4D \times 4D$ *doubly block circulant* (Jain, 1989, p. 28) or *block-circulant circulant-block* (BCCB) (Chan and Jin, 2007, p. 76) matrix. What follows aims at explaining how the original matrix-vector products defined by equations 40 and 41, involving a $D \times D$ BTTB matrix $\mathbf{G}$ exemplified by equation 39, can be

efficiently computed in terms of the auxiliary matrix-vector product given by equation 43, which has a $4D \times 4D$ BCCB matrix $\mathbf{G}_c$.

Matrix $\mathbf{G}_c$ (equation 43) is formed by $2N_B \times 2N_B$ blocks, where each block $\mathbf{G}_c^n$, $n \in \{(1 - N_B) : (N_B - 1)\}$ is a $2N_b \times 2N_b$ circulant matrix. For the case in which the original matrix-vector product is that defined by equation 40, the first column of blocks forming the BCCB matrix $\mathbf{G}_c$ is given by

$$\mathbf{G}_c[:, : 2N_b] = \begin{bmatrix} \mathbf{G}_c^0 \\ \mathbf{G}_c^{-1} \\ \vdots \\ \mathbf{G}_c^{1-N_B} \\ \mathbf{0} \\ \mathbf{G}_c^{N_B-1} \\ \vdots \\ \mathbf{G}_c^1 \end{bmatrix}_{4D \times 2N_b} , \tag{46}$$

with blocks $\mathbf{G}_c^n$ having the first column given by

$$\mathbf{G}_c^n[:, 1] = \begin{bmatrix} \mathbf{G}^n[:, 1] \\ 0 \\ (\mathbf{G}^n[1, N_b : 2])^\top \end{bmatrix}_{2N_b \times 2N_b} , \quad n \in \{(1 - N_B) : (N_B - 1)\} , \tag{47}$$

where $\mathbf{G}^n$ are the blocks forming the BTTB matrix $\mathbf{G}$ (equation 39). For the case in which the original matrix-vector product is that defined by equation 41, the first column of blocks forming the BCCB matrix $\mathbf{G}_c$ is given by

$$\mathbf{G}_c[:, : 2N_b] = \begin{bmatrix} \mathbf{G}_c^0 \\ \mathbf{G}_c^1 \\ \vdots \\ \mathbf{G}_c^{N_B-1} \\ \mathbf{0} \\ \mathbf{G}_c^{1-N_B} \\ \vdots \\ \mathbf{G}_c^{-1} \end{bmatrix}_{4D \times 2N_b} , \tag{48}$$

with blocks $\mathbf{G}_c^n$ having the first column given by

$$\mathbf{G}_c^n[:, 1] = \begin{bmatrix} (\mathbf{G}^n[1, :])^\top \\ 0 \\ \mathbf{G}^n[N_b : 2, 1] \end{bmatrix}_{2N_b \times 2N_b} , \quad n \in \{(1 - N_B) : (N_B - 1)\} . \tag{49}$$

The complete matrix $\mathbf{G}_c$ (equation 43) is obtained by properly downshifting the block columns $\mathbf{G}_c[:, : 2N_b]$ defined by equation 46 or 48. Similarly, the n-th block $\mathbf{G}_c^n$ of $\mathbf{G}_c$ is obtained by properly downshifting the first columns $\mathbf{G}_c^\ell[:, 1]$ defined by equation 47 or 49.

Note that $\mathbf{G}_c$ (equation 43) is a $4D \times 4D$ matrix and $\mathbf{G}$ (equation 39) is a $D \times D$ matrix. It seems weird to say that computing $\mathbf{G}_c\mathbf{v}_c$ is more efficient than directly computing $\mathbf{G}\mathbf{v}$. To understand this, we need first

to use the fact that BCCB matrices are diagonalized by the 2D unitary discrete Fourier transform (DFT) (e.g., Davis, 1979, p. 31). Because of that, $\mathbf{G}_c$ can be written as

$$\mathbf{G}_c = \left(\boldsymbol{\mathcal{F}}_{2N_B} \otimes \boldsymbol{\mathcal{F}}_{2N_b}\right)^* \boldsymbol{\Lambda} \left(\boldsymbol{\mathcal{F}}_{2N_B} \otimes \boldsymbol{\mathcal{F}}_{2N_b}\right) \,, \tag{50}$$

where the symbol "$\otimes$" denotes the Kronecker product (e.g., Horn and Johnson, 1991, p. 243), $\boldsymbol{\mathcal{F}}_{2N_B}$ and $\boldsymbol{\mathcal{F}}_{2N_b}$ are the $2N_B \times 2N_B$ and $2N_b \times 2N_b$ unitary DFT matrices (e.g., Davis, 1979, p. 31), respectively, the superscritpt "$*$" denotes the complex conjugate and $\boldsymbol{\Lambda}$ is a $4D \times 4D$ diagonal matrix containing the eigenvalues of $\mathbf{G}_c$. Due to the diagonalization of the matrix $\mathbf{G}_c$, equation 43 can be rewritten by using equation 50 and premultiplying both sides of the result by $\left(\boldsymbol{\mathcal{F}}_{2N_B} \otimes \boldsymbol{\mathcal{F}}_{2N_b}\right)$, i.e.,

$$\boldsymbol{\Lambda} \left(\boldsymbol{\mathcal{F}}_{2N_B} \otimes \boldsymbol{\mathcal{F}}_{2N_b}\right) \mathbf{v}_c = \left(\boldsymbol{\mathcal{F}}_{2N_B} \otimes \boldsymbol{\mathcal{F}}_{2N_b}\right) \mathbf{w}_c \,. \tag{51}$$

By following Takahashi et al. (2020), we rearrange equation 51 as follows

$$\boldsymbol{\mathcal{L}} \circ \left(\boldsymbol{\mathcal{F}}_{2N_B} \boldsymbol{\mathcal{V}}_c \boldsymbol{\mathcal{F}}_{2N_b}\right) = \boldsymbol{\mathcal{F}}_{2N_B} \boldsymbol{\mathcal{W}}_c \boldsymbol{\mathcal{F}}_{2N_b} \tag{52}$$

where "$\circ$" denotes the Hadamard product (e.g., Horn and Johnson, 1991, p. 298) and $\boldsymbol{\mathcal{L}}$, $\boldsymbol{\mathcal{V}}_c$ and $\boldsymbol{\mathcal{W}}_c$ are $2N_B \times 2N_b$ matrices obtained by rearranging, along their rows, the elements forming the diagonal of $\boldsymbol{\Lambda}$ (equation 50), vector $\mathbf{v}_c$ and vector $\mathbf{w}_c$ (equation 44), respectively. Then, by premultiplying both sides of equation 52 by $\boldsymbol{\mathcal{F}}^*_{2N_B}$ and then postmultiplying both sides by $\boldsymbol{\mathcal{F}}^*_{2N_b}$, we obtain

$$\boldsymbol{\mathcal{F}}^*_{2N_B} \left[\boldsymbol{\mathcal{L}} \circ \left(\boldsymbol{\mathcal{F}}_{2N_B} \boldsymbol{\mathcal{V}}_c \boldsymbol{\mathcal{F}}_{2N_b}\right)\right] \boldsymbol{\mathcal{F}}^*_{2N_b} = \boldsymbol{\mathcal{W}}_c \,. \tag{53}$$

Finally, we get from equation 50 that matrix $\boldsymbol{\mathcal{L}}$ can be computed by using only the first column $\mathbf{G}_c[:, 1]$ of the BCCB matrix $\mathbf{G}_c$ (equation 43) according to (Takahashi et al., 2020)

$$\boldsymbol{\mathcal{L}} = \sqrt{4D}\, \boldsymbol{\mathcal{F}}_{2N_B} \boldsymbol{\mathcal{C}} \boldsymbol{\mathcal{F}}_{2N_b} \,, \tag{54}$$

where $\boldsymbol{\mathcal{C}}$ is a $2N_B \times 2N_b$ matrix obtained by rearranging, along its rows, the elements of $\mathbf{G}_c[:, 1]$ (equation 43). It is important noting that the matrices $\boldsymbol{\mathcal{C}}$ and $\boldsymbol{\mathcal{L}}$ (equation 54) associated with the BTTB matrix $\mathbf{G}$ (equation 39) are different from those associated with $\mathbf{G}^\top$.

The whole procedure to compute the original matrix-vector products $\mathbf{G}\mathbf{v}$ (equation 40) and $\mathbf{G}^\top\mathbf{v}$ (equation 41) consists in (i) rearranging the elements of the vector $\mathbf{v}$ and the first column $\mathbf{G}[:, 1]$ of matrix $\mathbf{G}$ into the matrices $\boldsymbol{\mathcal{V}}_c$ and $\boldsymbol{\mathcal{C}}$ (equations 53 and 54), respectively; (ii) computing terms $\boldsymbol{\mathcal{F}}_{2N_B} \boldsymbol{\mathcal{A}} \boldsymbol{\mathcal{F}}_{2N_b}$ and $\boldsymbol{\mathcal{F}}^*_{2N_B} \boldsymbol{\mathcal{A}} \boldsymbol{\mathcal{F}}^*_{2N_b}$, where $\boldsymbol{\mathcal{A}}$ is a given matrix, and a Hadamard product to obtain $\boldsymbol{\mathcal{W}}_c$ (equation 53); and (iii) retrieve the elements of vector $\mathbf{w}$ (equation 40) from $\boldsymbol{\mathcal{W}}_c$ (equation 53). It is important noting that the steps (i) and (iii) do not have any computational cost because they involve only reorganizing elements of vectors and matrices. Besides, the terms $\boldsymbol{\mathcal{F}}_{2N_B} \boldsymbol{\mathcal{A}} \boldsymbol{\mathcal{F}}_{2N_b}$ and $\boldsymbol{\mathcal{F}}^*_{2N_B} \boldsymbol{\mathcal{A}} \boldsymbol{\mathcal{F}}^*_{2N_b}$ in step (ii) represent, respectively, the 2D Discrete Fourier Transform (2D-DFT) and the 2D Inverse Discrete Fourier Transform (2D-IDFT) of $\boldsymbol{\mathcal{A}}$. These transforms can be efficiently computed by using the 2D Fast Fourier Transform (2D-FFT). Hence, the original matrix-vector products $\mathbf{G}\mathbf{v}$ (equation 40) and $\mathbf{G}^\top\mathbf{v}$ (equation 41) can be efficiently computed by using the 2D-FFT.

Algorithms 6 and 7 show pseudo-codes for the convolutional equivalent-layer method proposed by Takahashi et al. (2020, 2022). Note that those authors formulate the overdetermined problem (equation 22) of obtaining an estimate $\tilde{\mathbf{p}}$ for the parameter vector $\mathbf{p}$ (equation 3) as an *iterative deconvolution* via

conjugate gradient normal equation residual (CGNR) Golub and Van Loan (2013, sec. 11.3) or *conjugate gradient least squares* (CGLS) (Aster et al., 2019, p. 165) method. They consider $\mathbf{H} = \mathbf{I}_P$ (equation 9), $\mu = 0$ (equation 11), $\mathbf{W}_d = \mathbf{W}_q = \mathbf{I}_P$ (equations 12 and 13) and $\bar{\mathbf{p}} = \mathbf{0}$ (equation 14). As shown by Takahashi et al. (2020, 2022), the CGLS produces stable estimates $\tilde{\mathbf{p}}$ for the parameter vector $\mathbf{p}$ (equation 3) in the presence of noisy potential-field data $\mathbf{d}$. This is a well-known property of the CGLS method (e.g., Aster et al., 2019, p. 166).

Algorithm 6: Generic pseudo-code for the convolutional equivalent-layer method proposed by Takahashi et al. (2020, 2022).

Initialization :

Set the regular grid of P equivalent sources on a horizontal plane z_0 ;
Set a tolerance ϵ and a maximum number of iterations `ITMAX` ;
Compute the first column $\mathbf{G}[:, 1]$ and row $\mathbf{G}[1, :]$ of the sensitivity matrix $\mathbf{G}$ (equation 3) for the particular case in which it has a BTTB structure (equation 39);
Rearrange the elements of $\mathbf{G}[:, 1]$ into matrix $\boldsymbol{\mathcal{C}}$, compute its 2D-DFT via 2D-FFT and multiply by $\sqrt{4D}$ to obtain a matrix $\boldsymbol{\mathcal{L}}'$ (equation 54);
Rearrange the elements of $\mathbf{G}[1, :]$ into matrix $\boldsymbol{\mathcal{C}}$, compute its 2D-DFT via 2D-FFT and multiply by $\sqrt{4D}$ to obtain a matrix $\boldsymbol{\mathcal{L}}''$ (equation 54);
Set $\tilde{\mathbf{p}} = \mathbf{0}$;
Set $\mathbf{r} = \mathbf{d}$ and compute $\delta = \|\mathbf{r}\|/D$;
Compute $\boldsymbol{\vartheta} = \mathbf{G}^\top \mathbf{r}$ (Algorithm 7) and $\rho_0 = \boldsymbol{\vartheta}^\top \boldsymbol{\vartheta}$;
Set $\tau = 0$ and $\boldsymbol{\eta} = \mathbf{0}$;
$m = 1$;
while $(\delta > \epsilon)$ **and** $(m <$ `ITMAX`$)$ **do**
    Update $\boldsymbol{\eta} \leftarrow \boldsymbol{\vartheta} + \tau\, \boldsymbol{\eta}$;
    Compute $\boldsymbol{\nu} = \mathbf{G}\, \boldsymbol{\eta}$ (Algorithm 7);
    Compute $\upsilon = \rho_0/(\boldsymbol{\nu}^\top \boldsymbol{\nu})$;
    Update $\tilde{\mathbf{p}} \leftarrow \tilde{\mathbf{p}} + \upsilon\, \boldsymbol{\eta}$;
    Update $\mathbf{r} \leftarrow \mathbf{r} - \upsilon\, \boldsymbol{\nu}$ and compute $\delta \leftarrow \|\upsilon\, \boldsymbol{\nu}\|/D$;
    Compute $\boldsymbol{\vartheta} = \mathbf{G}^\top \mathbf{r}$ (Algorithm 7) and $\rho = \boldsymbol{\vartheta}^\top \boldsymbol{\vartheta}$;
    Compute $\tau = \rho/\rho_0$;
    Update $\rho_0 \leftarrow \rho$;
    $m \leftarrow m + 1$;
end

Algorithm 7: Pseudo-code for computing the generic matrix-vector products given by equations 40 and 41 via fast 2D discrete convolution for a given vector $\mathbf{v}$ (equation 42) and matrix $\boldsymbol{\mathcal{L}}$ (equation 54).

Rearrange the elements of $\mathbf{v}$ (equations 40 and 42) into the matrix $\boldsymbol{\mathcal{V}}_c$ (equation 53);
Compute $\boldsymbol{\mathcal{F}}_{2N_B}\, \boldsymbol{\mathcal{V}}_c\, \boldsymbol{\mathcal{F}}_{2N_b}$ via 2D-FFT;
Compute the Hadamard product with matrix $\boldsymbol{\mathcal{L}}$ (equation 54);
Compute 2D-IDFT via 2D-FFT to obtain matrix $\boldsymbol{\mathcal{W}}_c$ (53);
Retrieve $\mathbf{w}$ (equations 40 and 42) from $\mathbf{w}_c$ (equations 43–45);

3.8 Direct deconvolution

The method proposed by Takahashi et al. (2020, 2022) can be reformulated to avoid the iterations of the conjugate gradient method. This alternative formulation consists in considering that $\mathbf{v} = \mathbf{p}$ and $\mathbf{w} = \mathbf{d}$ in

equation 40, where $\mathbf{p}$ is the parameter vector (equation 3) and $\mathbf{d}$ the observed data vector. In this case, the equality "=" in equation 40 becomes an approximation "≈". Then, equation 52 is manipulated to obtain

$$\boldsymbol{\mathcal{V}}_c \approx \boldsymbol{\mathcal{F}}^*_{2N_B} \left[\left(\boldsymbol{\mathcal{F}}_{2N_B} \, \boldsymbol{\mathcal{W}}_c \, \boldsymbol{\mathcal{F}}_{2N_b} \right) \circ \breve{\boldsymbol{\mathcal{L}}} \right] \boldsymbol{\mathcal{F}}^*_{2N_b} \, , \tag{55}$$

where

$$\breve{\boldsymbol{\mathcal{L}}} = \boldsymbol{\mathcal{L}}^* \oslash \left(\boldsymbol{\mathcal{L}} \boldsymbol{\mathcal{L}}^* + \zeta \mathbf{1} \right) \, , \tag{56}$$

$\mathbf{1}$ is a $4D \times 4D$ matrix of ones, "$\oslash$" denotes entrywise division and ζ is a positive scalar. Note that $\zeta = 0$ leads to $\mathbf{1} \oslash \boldsymbol{\mathcal{L}}$. In this case, the entrywise division may be problematic due to the elements of $\boldsymbol{\mathcal{L}}$ having absolute value equal or close to zero. So, a small ζ is set to avoid this problem in equation 56. Next, we use $\breve{\boldsymbol{\mathcal{L}}}$ to obtain a matrix $\boldsymbol{\mathcal{V}}_c$ from equation 55. Finally, the elements of the estimated parameter vector $\tilde{\mathbf{p}}$ are retrieved from the first quadrant of $\boldsymbol{\mathcal{V}}_c$. This procedure represents a *direct deconvolution* (e.g., Aster et al., 2019, p. 220) using a *Wiener filter* (e.g., Gonzalez and Woods, 2002, p. 263).

Differently from the convolutional equivalent-layer method proposed by Takahashi et al. (2020, 2022), the alternative direct deconvolution presented here produces an estimated parameter vector $\tilde{\mathbf{p}}$ directly from the observed data $\mathbf{d}$, in a single step, avoiding the conjugate gradient iterations. On the other hand, the alternative method presented here requires estimating a set of tentative parameter vectors $\tilde{\mathbf{p}}$ for different predefined ζ. Besides, there must be criterion to chose the best $\tilde{\mathbf{p}}$ from this tentative set. This can be made, for example, by using the well-known *L-curve* (Hansen, 1992). From a computational point of view, the number of CGLS iterations in the method proposed by Takahashi et al. (2020, 2022) is equivalent to the number of tentative estimated parameter vectors required to form the L-curve in the proposed direct deconvolution.

4 NUMERICAL STABILITY

All equivalent-layer methods aims at obtaining an estimate $\tilde{\mathbf{p}}$ for the parameter vector $\mathbf{p}$ (equation 3), which contains the physical property of the equivalent sources. Some methods do it by first obtaining an estimate $\tilde{\mathbf{q}}$ for the reparameterized parameter vector $\mathbf{q}$ (equation 9) and then using it to obtain $\tilde{\mathbf{p}}$ (equation 15). The stability of a solution $\tilde{\mathbf{p}}$ against noise in the observed data is rarely addressed. Here, we follow the numerical stability analysis presented in Siqueira et al. (2017).

For a given equivalent-layer method, we obtain an estimate $\tilde{\mathbf{p}}$ assuming noise-free potential-field data $\mathbf{d}$. Then, we create L different noise-corrupted data $\mathbf{d}^\ell$, $\ell \in \{1 : L\}$, by adding L different sequences of pseudorandom Gaussian noise to $\mathbf{d}$, all of them having zero mean. From each $\mathbf{d}^\ell$, we obtain an estimate $\tilde{\mathbf{p}}^\ell$. Regardless of the particular equivalent-layer method used, the following inequality (Aster et al., 2019, p. 66) holds true:

$$\Delta p^\ell \leq \kappa \, \Delta d^\ell \, , \quad \ell \in \{1 : L\} \, , \tag{57}$$

where κ is the constant of proportionality between the model perturbation

$$\Delta p^\ell = \frac{\| \tilde{\mathbf{p}}^\ell - \tilde{\mathbf{p}} \|}{\| \tilde{\mathbf{p}} \|} \, , \quad \ell \in \{1 : L\} \, , \tag{58}$$

and the data perturbation

$$\Delta d^\ell = \frac{\| \mathbf{d}^\ell - \mathbf{d} \|}{\| \mathbf{d} \|} \, , \quad \ell \in \{1 : L\} \, , \tag{59}$$

with $\|\cdot\|$ representing the Euclidean norm. The constant κ acts as the condition number associated with the pseudo-inverse in a given linear inversion. The larger (smaller) the value of κ, the more unstable (stable) is the estimated solution. Equation 57 shows a linear relationship between the model perturbation Δp^{ℓ} and the data perturbation Δd^{ℓ} (equations 58 and 59). We estimate the κ (equation 57) associated with a given equivalent-layer method as the slope of the straight line fitted to the L points $\left(\Delta p^{\ell}\ ,\ \Delta d^{\ell}\right)$.

PAREI AQUI

Here, the analysis of solution stability is numerically conducted by applying the classical equivalent-layer technique with zeroth-order Tikhonov regularization, the convolutional method for gravimetric and magnetic data, the deconvolutional method (equation **??**) and the deconvolutional method with different values for the Wiener stabilization (equation **??**).

5 NUMERICAL SIMULATIONS

5.1 Floating-point operations calculation

To measure the computational effort of the different algorithms to solve the equivalent layer linear system, a non-hardware dependent method can be useful because allow us to do direct comparison between them. Counting the floating-point operations (*flops*), i.e., additions, subtractions, multiplications and divisions is a good way to quantify the amount of work of a given algorithm (Golub and Van Loan, 2013). For example, the number of *flops* necessary to multiply two vectors $\mathbb{R}^D$ is $2D$. A common matrix-vector multiplication with dimension $\mathbb{R}^{D\times D}$ and $\mathbb{R}^D$, respectively, is $2D^2$ and a multiplication of two matrices $\mathbb{R}^{D\times D}$ is $2D^3$. Figure **??** shows the total flops count for the different methods presented in this review with a crescent number of data, ranging from $10,000$ to $1,000,000$.

5.1.1 Normal equations using Cholesky decomposition

The equivalent sources can be estimated directly from solving the normal equations 18. In this work we will use the Cholesky decompositions method to calculate the necessary *flops* for a overdetermined problem (equation 20). In this method it is necessary to calculate the lower triangule matrix of the left side equation ($1/2D^3$), the Cholesky factor ($1/3D^3$), a matrix-vector multiplication ($2D^2$) and finally solving the triangular system ($2D^2$), totalizing

$$f_{classical} = \frac{5}{6}D^3 + 4D^2 \tag{60}$$

5.1.2 Window method (Leão and Silva, 1989)

The moving data-window scheme (Leão and Silva, 1989) solve M linear systems with much smaller sizes (equation **??**) in comparison to the original $D \times D$ system. For our results we are considering a data-window of the same size of wich the authors presented in theirs work ($D' = 49$) and the same number of equivalent sources ($P' = 225$). Using the algorithm 1 as a guide, we have a matrix-matrix multiplication ($2D'^2P'$), a scalar multiplication and a sum with diagonal matrices (D' each), a matrix inverse (D'^3), another matrix-matrix product ($2D'P'^2$), a matrix-vetor product ($2P'D'$) and finally a iteration with a vector-vector multiplication ($2D'$). The *flops* are

$$f_{window} = M2D' + 2P'D' + \frac{2D'^3}{3} + 2D' + 2D'P'(D' + P') \tag{61}$$

Here we are considering a $2D'^3/3$ *flops* count for the Gauss-Jordan inverse matrix algorithm and $M = D$ as we want to calculate the same number of transformations as observation points. Notice that this algorithm takes advantage of a regular grid to calculate only once the inverse matrix and the harmonic functions of $\mathbf{a}'$, with a irregular grid these calculations would be necessary at each iteration. Also this method does not store the equivalent sources estimatives saving computer memory, however, any other potential field transformation would require to run the algorithm again with the compatible harmonic function $\mathbf{a}'$ (equation 5).

5.1.3 PEL method (Oliveira Jr. et al., 2013)

The polynomial equivalent layer uses a similiar approach of moving windows from Leão and Silva (1989). For this operations calculation (equation 32) we used a first degree polynomial (two variables) and each

window contains $D' = 1,000$ observed data and $P' = 1,000$ equivalent sources. Following the steps given in (Oliveira Jr. et al., 2013) the total *flops* becomes

$$f_{pel} = \frac{1}{3}N_c^3 + 2N_c^2 + 2DP'N_c + N_c^2D + 2N_cD + 2DC \tag{62}$$

where N_c is the number of constant coefficients for the first degree polynomial ($C = 3$) times the number of windows ($C \times M$).

5.1.4 Conjugate gradient least square (CGLS)

The CGLS method is a very stable and fast algorithm for solving linear systems iteratively. Its computational complexity envolves a matrix-vector product outside the loop ($2D^2$), two matrix-vector products inside the loop ($4D^2$) and six vector products inside the loop ($12D$) (**?**)

$$f_{cgls} = 2D^2 + it(4D^2 + 12D) \tag{63}$$

5.1.5 Wavelet compression method with CGLS (**?**)

For the wavelet method (equation **??**) we have calculated a compression rate of 98% ($C_r = 0.02$) as the authors used in **?** and the wavelet transformation requiring $\log_2(D)$ *flops* each (equations 34 and 35), with its inverse also using the same number of operations (equation 38). The normalization using diagonal matrix $\mathbf{L}$ in equations 36, 37 and 38 can be simplified to a matrix-vector product ($2DP$) and two vector-vector products($2Peach$). Combined with the conjugate gradient least square necessary steps and iterations, the number of *flops* are

$$f_{wavelet} = 2DP + 4P + 2DC_r + 4D\log_2(D) + it(4D\log_2(D) + 4DC_r + 12C_r) \tag{64}$$

5.1.6 Fast equivalent layer for gravity data (Siqueira et al., 2017)

The fast equivalent layer from Siqueira et al. (2017) solves the linear system in *it* iterations. The main cost of this method (algorithm 5) is the matrix-vector multiplication to assess the predicted data ($2D^2$) and three simply element by element vector sum, subtraction and division ($3D$ total)

$$f_{siqueira} = it(3D + 2D^2) \tag{65}$$

5.1.7 Convolutional equivalent layer for gravity data (**?**)

This methods replaces the matrix-vector multiplication of the iterative fast-equivalent technique (Siqueira et al., 2017) by three steps, involving a Fourier transform, an inverse Fourier transform, and a Hadamard product of matrices (equation 53). Considering that the first column of our BCCB matrix has $4D$ elements, the flops count of this method is a combination of agorithms 5 and 7

$$f_{convgrav} = \kappa 4D\log_2(4D) + it(27D + \kappa 8D\log_2(4D)) \tag{66}$$

In the resultant count we considered a *radix-2* algorithm for the fast Fourier transform and its inverse, which has a κ equals to 5 and requires $\kappa 4D\log_2(4D)$ flops each. The Hadarmard product of two matrices of $4D$ elements with complex numbers takes $24D$ flops. Note that equation 66 is different from the one

presented in **?** because we also added the flops necessary to calculate the eigenvalues in this form. It does not differentiate much in order of magnitude because the iterative part is the most costful.

5.1.8 Convolutional equivalent layer for magnetic data (**?**)

The convolutional equivalent layer for magnetic (algorithm 6) data uses the same flops count of the main operations as in the gravimetric case (equation 53), the difference is the use of the conjugate gradient algorithm to solve the inverse problem. It requires a Hadamard product outside of the iterative loop and the matrix-vector and vector-vector multiplications inside the loop as seem in equation 63.

$$f_{convmag} = \kappa 16D \log_2(4D) + 24D + it(\kappa 16D \log_2(4D) + 60D) \tag{67}$$

5.1.9 Deconvolutional method

The deconvolution method does not require an iterative algorithm, rather it solves the estimative of the physical properties in a single step using the $4D$ eigenvalues of the BCCB matrix as in the convolutional method. From equation **??** it is possible to deduce this method requires two fast Fourier transform ($\kappa 4D \log_2(4D)$), one for the eigenvalues and another for the data transformation, an element by element division ($24D$) and finally, a fast inverse Fourier transform for the final estimative ($\kappa 4D \log_2(4D)$).

$$f_{deconv} = \kappa 12D \log_2(4D) + 24D \tag{68}$$

Using the deconvolutional method with a Wiener stabilization adds two multiplications of complex elements of the conjugates eigenvalues ($24D$ each) and the sum of $4D$ elements with the stabilization parameter μ as shown in equation **??**

$$f_{deconvwiener} = \kappa 12D \log_2(4D) + 76D \tag{69}$$

CONFLICT OF INTEREST STATEMENT

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

AUTHOR CONTRIBUTIONS

The Author Contributions section is mandatory for all articles, including articles by sole authors. If an appropriate statement is not provided on submission, a standard one will be inserted during the production process. The Author Contributions statement must describe the contributions of individual authors referred to by their initials and, in doing so, all authors agree to be accountable for the content of the work. Please see here for full authorship criteria.

FUNDING

Diego Takahashi was supported by a Post-doctoral scholarship from CNPq (grant 300809/2022-0) Valéria C.F. Barbosa was supported by fellowships from CNPq (grant 309624/2021-5) and FAPERJ (grant 26/202.582/2019). Vanderlei C. Oliveira Jr. was supported by fellowships from CNPq (grant 315768/2020-7) and FAPERJ (grant E-26/202.729/2018).

ACKNOWLEDGMENTS

We thank the brazillian federal agencies CAPES, CNPq, state agency FAPERJ and Observatório Nacional research institute and Universidade do Estado do Rio de Janeiro.

DATA AVAILABILITY STATEMENT

The datasets generated for this study can be found in the frontiers-paper Github repository link: https://github.com/DiegoTaka/frontiers-paper.

REFERENCES

Aster, R. C., Borchers, B., and Thurber, C. H. (2019). *Parameter Estimation and Inverse Problems* (Elsevier), 3 edn.

Barbosa, V. C. F., Silva, J. B., and Medeiros, W. E. (1997). Gravity inversion of basement relief using approximate equality constraints on depths. *Geophysics* 62, 1745–1757

Barnes, G. and Lumley, J. (2011). Processing gravity gradient data. *GEOPHYSICS* 76, I33–I47. doi:10.1190/1.3548548

Blakely, R. J. (1996). *Potential Theory in Gravity and Magnetic Applications* (Cambridge University press)

Bott, M. H. P. (1960). The use of Rapid Digital Computing Methods for Direct Gravity Interpretation of Sedimentary Basins. *Geophysical Journal International* 3, 63–67. doi:10.1111/j.1365-246X.1960.tb00065.x

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

Cordell, L. (1992). A scattered equivalent-source method for interpolation and gridding of potential-field data in three dimensions. *Geophysics* 57, 629–636

Dampney, C. N. G. (1969). The equivalent source technique. *GEOPHYSICS* 34, 39–53. doi:10.1190/1.1439996

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* 38, 339–348. doi:10.1190/1.1440344

Golub, G. H. and Van Loan, C. F. (2013). *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences (Johns Hopkins University Press), 4 edn.

Gonzalez, R. C. and Woods, R. E. (2002). *Digital Image Processing* (Prentice Hall), 2 edn.

Gonzalez, S. P., Barbosa, V. C. F., and Oliveira Jr., V. C. (2022). Analyzing the ambiguity of the remanent-magnetization direction separated into induced and remanent magnetic sources. *Journal of Geophysical Research: Solid Earth* 127, 1–24. doi:10.1029/2022JB024151

Guspí, F., Introcaso, A., and Introcaso, B. (2004). Gravity-enhanced representation of measured geoid undulations using equivalent sources. *Geophysical Journal International* 159, 1–8. doi:10.1111/j.1365-246X.2004.02364.x

Guspí, F. and Novara, I. (2009). Reduction to the pole and transformations of scattered magnetic data using Newtonian equivalent sources. *GEOPHYSICS* 74, L67–L73. doi:10.1190/1.3170690

Hansen, P. C. (1992). Analysis of discrete ill-posed problems by means of the l-curve. *SIAM Review* 34, 561–580. doi:10.1137/1034115

Hansen, R. O. and Miyazaki, Y. (1984). Continuation of potential fields between arbitrary surfaces. *GEOPHYSICS* 49, 787–795. doi:10.1190/1.1441707

Horn, R. A. and Johnson, C. R. (1991). *Topics in Matrix Analysis* (Cambridge University Press), 1 edn.

Jain, A. K. (1989). *Fundamentals of Digital Image Processing* (Pearson), 1 edn.

Jirigalatu, J. and Ebbing (2019). A fast equivalent source method for airborne gravity gradient data. *Geophysics* 84, G75–G82. doi:10.1190/GEO2018-0366.1

Kellogg, O. D. (1967). *Foundations of Potential Theory* (Springer-Verlag), reprint from the first edition of 1929 edn.

Kennett, B., Sambridge, M., and Williamson, P. (1988). Subspace methods for large inverse problems with multiple parameter classes. *Geophysical Journal International* 94, 237–247

Leão, J. W. D. and Silva, J. B. C. (1989). Discrete linear transformations of potential field data. *Geophysics* 54, 497–507. doi:10.1190/1.1442676

Li, Y., Nabighian, M., and Oldenburg, D. W. (2014). Using an equivalent source with positivity for low-latitude reduction to the pole without striation. *GEOPHYSICS* 79, J81–J90. doi:10.1190/geo2014-0134.1

Li, Y. and Oldenburg, D. W. (2010). Rapid construction of equivalent sources using wavelets. *GEOPHYSICS* 75, L51–L59. doi:10.1190/1.3378764

Mendonça, C. A. (2020). Subspace method for solving large-scale equivalent layer and density mapping problems. *GEOPHYSICS* 85, G57–G68. doi:10.1190/geo2019-0302.1

Mendonça, C. A. and Silva, J. B. C. (1994). The equivalent data concept applied to the interpolation of potential field data. *Geophysics* 59, 722–732. doi:10.1190/1.1443630

Menke, W. (2018). *Geophysical data analysis: Discrete inverse theory* (Elsevier), 4 edn.

Oldenburg, D., McGillivray, P., and Ellis, R. (1993). Generalized subspace methods for large-scale inverse problems. *Geophysical Journal International* 114, 12–20

Oliveira Jr., V. C., Barbosa, V. C. F., and Uieda, L. (2013). Polynomial equivalent layer. *GEOPHYSICS* 78, G1–G13. doi:10.1190/geo2012-0196.1

Reis, A. L. A., Oliveira Jr., V. C., and Barbosa, V. C. F. (2020). Generalized positivity constraint on magnetic equivalent layers. *Geophysics* 85, 1–45. doi:10.1190/geo2019-0706.1

Roy, A. (1962). Ambiguity in geophysical interpretation. *GEOPHYSICS* 27, 90–99. doi:10.1190/1.1438985

Silva, J. B. C. (1986). Reduction to the pole as an inverse problem and its application to low-latitude anomalies. *GEOPHYSICS* 51, 369–382. doi:10.1190/1.1442096

Siqueira, F., Oliveira Jr., V. C., and Barbosa, V. C. F. (2017). Fast iterative equivalent-layer technique for gravity data processing: A method grounded on excess mass constraint. *GEOPHYSICS* 82, G57–G69. doi:10.1190/GEO2016-0332.1

Skilling, J. and Bryan, R. (1984). Maximum entropy image reconstruction-general algorithm. *Monthly Notices of the Royal Astronomical Society, Vol. 211, NO. 1, P. 111, 1984* 211, 111

Soler, S. R. and Uieda, L. (2021). Gradient-boosted equivalent sources. *Geophysical Journal International* 227, 1768–1783. doi:10.1093/gji/ggab297

Takahashi, D., Oliveira Jr., V. C., and Barbosa, V. C. (2022). Convolutional equivalent layer for magnetic data processing. *Geophysics* 87, 1–59

Takahashi, D., Oliveira Jr., V. C., and Barbosa, V. C. F. (2020). Convolutional equivalent layer for gravity data processing. *GEOPHYSICS* 85, G129–G141. doi:10.1190/geo2019-0826.1

van der Sluis, A. and van der Vorst, H. A. (1987). Numerical solution of large, sparse linear algebraic systems arising from tomographic problems. In *Seismic tomography with applications in global seismology and exploration geophysics*, ed. G. Nolet (D. Reidel Publishing Company), chap. 3. 49–83

Van Loan, C. F. (1992). *Computational Frameworks for the fast Fourier transform*. Frontiers in Applied Mathematics (SIAM)

Xia, J. and Sprowl, D. R. (1991). Correction of topographic distortion in gravity data. *Geophysics* 56, 537–541

Xia, J., Sprowl, D. R., and Adkins-Heljeson, D. (1993). Correction of topographic distortions in potential-field data; a fast and accurate approach. *Geophysics* 58, 515–523. doi:10.1190/1.1443434

Zhao, G., Chen, B., Chen, L., Liu, J., and Ren, Z. (2018). High-accuracy 3D Fourier forward modeling of gravity field based on the Gauss-FFT technique. *Journal of Applied Geophysics* 150, 294–303. doi:10.1016/j.jappgeo.2018.01.002

Zidarov, D. (1965). Solution of some inverse problems of applied geophysics. *Geophysical Prospecting* 13, 240–246. doi:10.1111/j.1365-2478.1965.tb01932.x