Convolutional equivalent layer for gravity data processing

(May 5, 2020)

GEO-XXXX

Running head: **Convolutional equivalent layer**

ABSTRACT

We have developed an efficient and very fast equivalent-layer technique for gravity data processing by modifying an iterative method grounded on excess mass constraint that does not require the solution of linear systems. Taking advantage of the symmetric Block-Toeplitz Toeplitz-block (BTTB) structure of the sensitivity matrix, that raises when regular grids of observation points and equivalent sources (point masses) are used to set up a fictitious equivalent layer, we have developed an algorithm which greatly reduces the number o f flops and RAM memory necessary to estimate a 2D mass distribution over the equivalent layer. The structure of symmetric BTTB matrix consists of the elements of the first column of the sensitivity matrix, which in turn can be embedded into a symmetric Block-Circulant Circulant-Block (BCCB) matrix. Likewise, only the first column of the BCCB matrix is needed to reconstruct the full sensitivity matrix completely. From the first column of BCCB matrix, its eigenvalues can be calculated using the 2D Fast Fourier Transform (2D FFT), which can be used to readily compute the matrix-vector product of the forward modeling in the fast equivalent-layer technique. As a result, our method is efficient to process very large datasets using either fine- or mid-grid meshes. The larger the dataset, the faster and more efficient our method becomes compared to the available equivalent-layer techniques. Tests with synthetic data demonstrate the ability of our method to satisfactorily

upward- and downward-continuing the gravity data. Our results show very small border effects and noise amplification compared to those produced by the classical approach in the Fourier domain. Besides, they show that while the running time of our method is ≈ 30.9 seconds for processing $N = 1,000,000$ observations, the fast equivalent-layer technique spent ≈ 46.8 seconds with $N = 22,500$. A test with field data from Carajás Province, Brazil, illustrates the low computational cost of our method to process a large data set composed of $N = 250,000$ observations.

INTRODUCTION

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

Many previous works have used the equivalent layer to perform different potential-field data transformations such as gridding (e.g., Dampney, 1969; Cordell, 1992; Mendonça and Silva, 1994), upward/downward continuation (e.g., Emilia, 1973; Hansen and Miyazaki, 1984; Li and Oldenburg, 2010), reduction to the pole (e.g., Silva, 1986; Leão and Silva, 1989; Guspí and Novara, 2009; Oliveira Jr. et al., 2013), combining multiple data sets (e.g., Boggs and Dransfield, 2004) and gradient data processing (e.g., Barnes and Lumley, 2011).

Although the use of the equivalent-layer technique increased over the last decades, one of the biggest problems is still its high computational cost for processing large-data sets. This problem propelled several studies to improve the computational efficiency of the equivalent layer technique. Leão and Silva (1989) developed a fast method for processing a regular grid of potential-field data. The method consists in estimating an equivalent layer which exactly reproduces the potential-field data within a small data window. The data window is shifted

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

By following a different strategy, Li and Oldenburg (2010) developed a rapid method that transforms the dense sensitivity matrix associated with the linear system into a sparse

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

Oliveira Jr. et al. (2013) developed a fast method based on the reparameterization of the physical-property distribution within the equivalent layer. Those authors divided the equivalent layer into a regular grid of equivalent-source windows inside which the physical-property distribution is described by bivariate polynomial functions. By using this polynomial representation, the inverse problem for estimating the equivalent layer is posed in the space of the total number of polynomial coefficients within all equivalent-source windows instead of in the space of the total number of equivalent sources. According to Oliveira Jr. et al. (2013), the computational efficiency of their method relies on the fact that the total number of polynomial coefficients needed to describe the physical-property distribution within the equivalent layer is generally much smaller than the number of equivalent

sources, leading to a very smaller linear system to be solved. Those authors could verify that the total number of *flops* needed for building and solving the linear inverse problem of estimating the total number of polynomial coefficients can be reduced by as many as three and four orders of magnitude, respectively, if compared with the same inverse problem of estimating the physical property of each equivalent source via Cholesky decomposition.

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

In the present work, we show that the sensitivity matrix associated with a planar equivalent layer of point masses has a very well-defined structure called Block-Toeplitz Toeplitz-Block (BTTB) for the case in which (i) the observed gravity data is located on a regularly

spaced grid at constant height and (ii) there is one point mass directly beneath each observation point. This technique have been successfully used in potential-field methods for 3D gravity inversion (Zhang and Wong, 2015), downward continuation (Zhang et al., 2016) and 3D magnetic modeling (Qiang et al., 2019). By using this property, we propose an efficient algorithm based on FFT convolution (e.g., Van Loan, 1992, p. 207) for computing the forward problem at each iteration of the fast equivalent-layer technique proposed by Siqueira et al. (2017). Our method uses the gravitational effect produced by a single point mass to compute the effect produced by the whole equivalent layer, which results in a drastic reduction not only in the number of flops, but also in the RAM memory usage of the fast equivalent-layer technique. Tests with synthetic and field data illustrate the good performance of our method in processing large gravity data sets.

METHODOLOGY

Equivalent-layer technique for gravity data processing

Let d_i^o be the observed gravity data at the point (x_i, y_i, z_i), $i = 1, ..., N$, of a local Cartesian system with x-axis pointing to north, the y-axis pointing to east and the z-axis pointing downward. Let us consider an equivalent layer composed by a set of N point masses (equivalent sources) over a layer located at depth z_0 ($z_0 > z_i$) and whose x- and y- coordinates of each point mass coincides with the corresponding coordinates of the observation directly above. There is a linear relationship that maps the unknown mass distribution onto the gravity data given by

$$\mathbf{d}(\mathbf{p}) = \mathbf{A}\mathbf{p}\,, \tag{1}$$

where $\mathbf{d}$ is an $N \times 1$ vector whose ith element is the predicted gravity data at the ith point (x_i,y_i,z_i), $\mathbf{p}$ is the unknown $N \times 1$ parameter vector whose jth element p_j is the mass of the jth equivalent source (point mass) at the jth Cartesian coordinates (x_j,y_j,z_0) and $\mathbf{A}$ is an $N \times N$ sensitivity matrix whose ijth element is given by

$$a_{ij} = \frac{c_g\, G\,(z_0 - z_i)}{[(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_0)^2]^{\frac{3}{2}}}\,, \tag{2}$$

where G is the Newton's gravitational constant and $c_g = 10^5$ transforms from m/s^2 to mGal. Notice that the sensitivity matrix depends on the ith coordinate of the observation and the jth coordinate of the equivalent source. For convenience, we designate these coordinates as *matrix coordinates* and the indices i and j as *matrix indices*. In the classical equivalent-layer technique, we estimate the regularized parameter vector from the observed gravity data $\mathbf{d}^o$ by

$$\hat{\mathbf{p}} = \left(\mathbf{A}^\top \mathbf{A} + \mu\, \mathbf{I}\right)^{-1} \mathbf{A}^\top \mathbf{d}^o\,. \tag{3}$$

Fast equivalent-layer technique

Siqueira et al. (2017) developed an iterative least-squares method to estimate the mass distribution over the equivalent layer based on the excess of mass and the positive correlation between the observed gravity data and the masses on the equivalent layer. They showed that the fast equivalent-layer technique has a better computational efficiency than the classical equivalent layer approach (equation 3) if the dataset is greater than at least 200 observation points, even using a large number of iterations.

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

$$\hat{\mathbf{p}}^0 = \tilde{\mathbf{A}}^{-1}\mathbf{d}^o \,, \tag{4}$$

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

$$\tilde{a}_{ii}^{-1} = \frac{\Delta s_i}{(2\pi\, G\, c_g)} \,, \tag{5}$$

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

$$\hat{\mathbf{p}}^{k+1} = \hat{\mathbf{p}}^k + \boldsymbol{\Delta}\hat{\mathbf{p}}^k \,, \tag{6}$$

where the mass correction is given by

$$\boldsymbol{\Delta}\hat{\mathbf{p}}^{k+1} = \tilde{\mathbf{A}}^{-1}(\mathbf{d}^o - \mathbf{A}\hat{\mathbf{p}}^k) \,. \tag{7}$$

At the kth iteration of Siqueira et al.'s (2017) method, the matrix-vector product $\mathbf{A}\hat{\mathbf{p}}^k = \mathbf{d}(\hat{\mathbf{p}}^k)$ must be calculated to get a new residual $\mathbf{d^0} - \mathbf{A}\hat{\mathbf{p}}^k$, which represents a bottleneck.

Considering the limitation of 16 Gb of RAM memory in our system, we could run the Siqueira et al.'s (2017) method only up to $22,500$ observation points; Otherwise, it is costly and can be prohibitive in terms of RAM memory to maintain such operation.

Structure of matrix A for regular grids

Consider that the observed data are located on an $N_x \times N_y$ regular grid of points regularly spaced by Δx and Δy along the x- and y-directions, respectively, on a horizontal plane defined by the constant vertical coordinate $z_1 < z_0$. As a consequence, a given pair of matrix coordinates (x_i, y_i), defined by the matrix index i, $i = 1, \ldots, N = N_x N_y$, is equivalent to a pair of coordinates (x_k, y_l) given by:

$$x_i \equiv x_k = x_1 + [k(i) - 1]\ \Delta x\ , \tag{8}$$

and

$$y_i \equiv y_l = y_1 + [l(i) - 1]\ \Delta y\ , \tag{9}$$

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

The integer functions assume different forms depending on the orientation of the regular grid of data. Consider the case in which the grid is oriented along the x-axis (Figure 1a). For convenience, we designate these grids as *x-oriented grids*. For them, we have the following integer functions:

$$i(k, l) = (l - 1)\, N_x + k \quad , \tag{10}$$

$$l(i) = \left\lceil \frac{i}{N_x} \right\rceil \tag{11}$$

and

$$k(i) = i - \left\lceil \frac{i}{N_x} \right\rceil N_x + N_x \quad , \tag{12}$$

where $\lceil \cdot \rceil$ denotes the ceiling function (Graham et al., 1994, p. 67). These integer functions are defined in terms of the matrix index i, but they can be defined in the same way by using the index j. Figure 1a illustrates an x-oriented grid defined by $N_x = 4$ and $N_y = 3$. In this example, the matrix coordinates x_7 and y_7, defined by the matrix index $i = 7$ (or $j = 7$), are equivalent to the grid coordinates x_3 and y_2, which are defined by the grid indices $k = 3$ and $l = 2$, respectively. These indices are computed with equations 11 and 12, by using the matrix index $i = 7$ (or $j = 7$).

Now, consider the case in which the regular grid of data is oriented along the y-axis (Figure 1b). For convenience, we call them *y-oriented grids*. Similarly to x-oriented grids, we have the following integer functions associated with y-oriented grids:

$$i(k, l) = (k - 1) N_y + l \quad , \tag{13}$$

$$k(i) = \left\lceil \frac{i}{N_y} \right\rceil \tag{14}$$

and

$$l(i) = i - \left\lceil \frac{i}{N_y} \right\rceil N_y + N_y \quad . \tag{15}$$

Figure 1b illustrates an y-oriented grid defined by $N_x = 4$ and $N_y = 3$. In this example, the matrix coordinates x_7 and y_7, defined by the matrix index $i = 7$ (or $j = 7$), are equivalent to the grid coordinates x_3 and y_1, which are defined by the grid indices $k = 3$ and $l = 1$, respectively. Differently from the example shown in Figure 1a, the grid indices of the present example are computed with equations 14 and 15, by using the matrix index $i = 7$ (or $j = 7$).

The element a_{ij} (equation 2) can be rewritten by using equations 8 and 9, giving rise to:

$$a_{ij} = \frac{c_g \, G \, \Delta z}{\left[\left(\Delta k_{ij} \, \Delta x \right)^2 + \left(\Delta l_{ij} \, \Delta y \right)^2 + \left(\Delta z \right)^2 \right]^{\frac{3}{2}}} \, , \tag{16}$$

where $\Delta z = z_0 - z_1$, $\Delta k_{ij} = k(i) - k(j)$ (equations 12 or 14) and $\Delta l_{ij} = l(i) - l(j)$ (equations 11 or 15). Notice that the structure of matrix $\mathbf{A}$ (equation 1) for the case in which its elements are given by a_{ij} (equation 16) is defined by the coefficients Δk_{ij} and Δl_{ij}.

For x-oriented grids, the coefficients Δk_{ij} and Δl_{ij} are computed by using equations 12 and 11, respectively. In this case, $\mathbf{A}$ (equation 1) is composed of $N_y \times N_y$ blocks, where each block is formed by $N_x \times N_x$ elements. For y-oriented grids, the coefficients Δk_{ij} and Δl_{ij} are computed by using equations 14 and 15, respectively. In this case, $\mathbf{A}$ (equation 1) is a composed of $N_x \times N_x$ blocks, where each block is formed by $N_y \times N_y$ elements. In both cases, $\mathbf{A}$ is Toeplitz blockwise, i.e., the blocks lying at the same block diagonal are equal to each other. Besides, the blocks located above the main diagonal are equal to those located below and each block is itself a Toeplitz matrix. These symmetries come from the fact that the coefficients Δk_{ij} and Δl_{ij} are squared at the denominator of a_{ij} (equation 16). Matrices with this well-defined pattern are called Doubly Block Toeplitz (Jain, 1989, p. 28) or symmetric Block-Toeplitz Toeplitz-Block (BTTB), for example. We opted for using the second term.

This well-defined pattern is better represented by using the *block indices* q and p. We

represent $\mathbf{A}$ (equation 1) as a grid of $Q \times Q$ blocks $\mathbf{A}_q$, $q = 0, \ldots, Q-1$, given by

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_0 & \mathbf{A}_1 & \cdots & \mathbf{A}_{Q-1} \\ \mathbf{A}_1 & \mathbf{A}_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{A}_1 \\ \mathbf{A}_{Q-1} & \cdots & \mathbf{A}_1 & \mathbf{A}_0 \end{bmatrix}_{N \times N} , \tag{17}$$

where each block has $P \times P$ elements conveniently represented by a_p^q, $p = 0, \ldots, P-1$, as follows:

$$\mathbf{A}_q = \begin{bmatrix} a_0^q & a_1^q & \cdots & a_{P-1}^q \\ a_1^q & a_0^q & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_1^q \\ a_{P-1}^q & \cdots & a_1^q & a_0^q \end{bmatrix}_{P \times P} , \tag{18}$$

with $N = QP$. The index q defines the block diagonal where $\mathbf{A}_q$ (equation 18) lies within the BTTB matrix $\mathbf{A}$ (equation 17). This index varies from 0, at the main diagonal, to $Q-1$, at the corners of $\mathbf{A}$. Similarly, the index p defines the diagonal where a_p^q lies within $\mathbf{A}_q$ (equation 18). This index varies from 0, at the main diagonal, to $P-1$, at the corners of $\mathbf{A}_q$. For x-oriented grids, $Q = N_y$, $P = N_x$ and the block indices q and p are defined, respectively, by the following integer functions of the matrix indices i and j:

$$q(i,j) = \mid l(i) - l(j) \mid \tag{19}$$

and

$$p(i,j) = \mid k(i) - k(j) \mid \quad , \tag{20}$$

where $l(i)$ and $l(j)$ are defined by equation 11 and $k(i)$ and $k(j)$ are defined by equation 12. 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:

$$q(i,j) = \mid k(i) - k(j) \mid \tag{21}$$

and

$$p(i,j) = \mid l(i) - l(j) \mid \quad , \tag{22}$$

where $k(i)$ and $k(j)$ are defined by equation 14 and $l(i)$ and $l(j)$ are defined by equation 15. Notice that, for each element a_{ij} (equation 16), defined by matrix indices i and j, there is a corresponding block element a_p^q, defined by block indices q (equations 19 or 21) and p (equations 20 or 22), so that

$$a_p^q \equiv a_{ij} \quad . \tag{23}$$

We also stress that matrix $\mathbf{A}$ (equation 1) defined by elements a_{ij} (equation 16) in terms of matrix indices i and j is strictly the same BTTB matrix $\mathbf{A}$ (equation 17) defined by the blocks $\mathbf{A}_q$ (equation 18) and block elements a_p^q (equation 23) in terms of the block indices q (equations 19 or 21) and p (equations 20 or 22).

It is important noting that different matrix indices i or j produce the same absolute values for the grid indices k (equations 12 or 14) and l (equations 11 or 15). As a consequence, different pairs of matrix indices i and j generate the same absolute values for the coefficients Δk_{ij} and Δl_{ij} that compose the denominator of a_{ij} (equation 16) and also the same values for the block indices q (equations 19 or 21) and p (equations 20 or 22), as well as the same block element a_p^q (equation 23). It means that elements a_{ij} defined by different matrix indices i and j have the same value. The key point for understanding the structure of BTTB matrix $\mathbf{A}$ (equation 17) is then, given a single element a_{ij} defined by matrix indices i and j, compute the grid indices k (equations 12 or 14) and l (equations 11 or 15). These grid indices are used to (1) compute the coefficients Δk_{ij} and Δl_{ij} and determine the value of a_{ij} with equation 16 and (2) compute the block indices q (equations 19 or 21) and p (equations 20 or 22) and determine the corresponding block element a_p^q (equation 23) forming $\mathbf{A}_q$ (equation 18).

Consider the x-oriented grid of $N_x \times N_y$ points shown in Figure 1a, with $N_x = 4$, $N_y = 3$ and $N = N_x\, N_y = 12$. To illustrate the relationship between the matrix indices (i and j) and the block indices (q and p), consider the element a_{ij} defined by $i = 2$ and $j = 10$, which is located in the 2nd line and 10th column of $\mathbf{A}$ (equation 1). By using equations 11 and 12, we obtain the grid indices $l(i) = 1$, $l(j) = 3$, $k(i) = 2$ and $k(j) = 2$. These grid indices result in the coefficients $\Delta k_{ij} = 0$ and $\Delta l_{ij} = -2$, which are used to compute the element a_{ij} (equation 16), as well as in the block indices $q = 2$ (equation 19) and $p = 0$ (equation 20). These block indices indicate that this element a_{ij} appears in the main diagonal of the blocks $\mathbf{A}_2$ (equation 18), which are located at the corners of $\mathbf{A}$ (equation 17). To verify this, let us take the matrix indices associated with these elements. They are $(i, j) = (1, 9)$, $(2, 10)$, $(3, 11)$, $(4, 12)$, $(9, 1)$, $(10, 2)$, $(11, 3)$ and $(12, 4)$. By using these matrix indices, it is easy to verify that all of them produce the same grid indices $l(i)$, $l(j)$, $k(i)$ and $k(j)$ (equations 11 and 12) as those associated with the element defined by $i = 2$ and $j = 10$ or vice versa ($i = 10$ and $j = 2$). Consequently, all of them produce elements a_{ij} (equation 16) having the same value. Besides, it is also easy to verify that all these matrix indices produce the same block indices $q = 2$ (equation 19) and $p = 0$ (equation 20) and the same block element a_p^q (equation 23). By repeating this procedure for all elements a_{ij}, $i = 1, \ldots, 12$, $j = 1, \ldots, 12$, forming the matrix $\mathbf{A}$ (equation 1) obtained from our x-oriented grid (Figure 1a), we can verify that

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_0 & \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{A}_1 & \mathbf{A}_0 & \mathbf{A}_1 \\ \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 \end{bmatrix}_{N \times N} , \tag{24}$$

where $\mathbf{A}_q$ (equation 18), $q = 0, \ldots, Q-1$, $Q = N_y$, are symmetric Toeplitz matrices given

by:

$$\mathbf{A}_q = \begin{bmatrix} a_0^q & a_1^q & a_2^q & a_3^q \\ a_1^q & a_0^q & a_1^q & a_2^q \\ a_2^q & a_1^q & a_0^q & a_1^q \\ a_3^q & a_2^q & a_1^q & a_0^q \end{bmatrix}_{N_x \times N_x} , \tag{25}$$

with elements a_p^q (equation 23) defined by $p = 0, \ldots, P-1$, $P = N_x$.

This procedure can also be used to verify that the matrix $\mathbf{A}$ (equation 1) obtained from the y-oriented grid illustrated in Figure 1b is given by

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_0 & \mathbf{A}_1 & \mathbf{A}_2 & \mathbf{A}_3 \\ \mathbf{A}_1 & \mathbf{A}_0 & \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 & \mathbf{A}_1 \\ \mathbf{A}_3 & \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 \end{bmatrix}_{N \times N} , \tag{26}$$

where $\mathbf{A}_q$ (equation 18), $q = 0, \ldots, Q-1$, $Q = N_x$, are symmetric Toeplitz matrices given by:

$$\mathbf{A}_q = \begin{bmatrix} a_0^q & a_1^q & a_2^q \\ a_1^q & a_0^q & a_1^q \\ a_2^q & a_1^q & a_0^q \end{bmatrix}_{N_y \times N_y} , \tag{27}$$

with elements a_p^q (equation 23) defined by $p = 0, \ldots, P-1$, $P = N_y$.

These examples (equations 24–27) show that the entire $N \times N$ BTTB matrix $\mathbf{A}$ (equations 1 and 17) can be defined by using only the elements forming its first column (or row). Notice that a column contains the gravitational effect produced by a single equivalent source at all N observation points.

BTTB matrix-vector product

The matrix-vector product $\mathbf{A}\hat{\mathbf{p}}^k$ (equation 7) required by the fast equivalent-layer technique (Siqueira et al., 2017) accounts for most of its total computation time and can cause RAM memory shortage when large data sets are used. This computational load can be drastically reduced by exploring the well-defined structure of matrix $\mathbf{A}$ (equation 1) for the particular case in which its elements a_{ij} are defined by equation 16. In this case, $\mathbf{A}$ is a symmetric BTTB matrix (equations 17 and 24–27) and the predicted data vector $\mathbf{d}(\mathbf{p})$ (equation 1) can be efficiently computed by using the 2D Discrete Fourier Transform (DFT). To do this, let us first rewrite $\mathbf{d}(\mathbf{p})$ and $\mathbf{p}$ (equation 1) as the following partitioned vectors:

$$\mathbf{d}(\mathbf{p}) = \begin{bmatrix} \mathbf{d}_0(\mathbf{p}) \\ \vdots \\ \mathbf{d}_{Q-1}(\mathbf{p}) \end{bmatrix}_{N\times 1} \tag{28}$$

and

$$\mathbf{p} = \begin{bmatrix} \mathbf{p}_0 \\ \vdots \\ \mathbf{p}_{Q-1} \end{bmatrix}_{N\times 1} , \tag{29}$$

where $\mathbf{d}_q(\mathbf{p})$ and $\mathbf{p}_q$, $q = 0, \ldots, Q-1$, are $P \times 1$ vectors. Notice that q is the block index defined by equations 19 and 21, Q defines the number of blocks $\mathbf{A}_q$ (equation 18) forming $\mathbf{A}$ (equation 17) and P defines the number of elements forming each block $\mathbf{A}_q$. Then, by using the partitioned vectors (equations 29 and 28) and remembering that $N = QP$, we define the auxiliary linear system

$$\mathbf{w} = \mathbf{C}\mathbf{v} , \tag{30}$$

where

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_0 \\ \vdots \\ \mathbf{w}_{Q-1} \\ \mathbf{0}_{2N\times 1} \end{bmatrix}_{4N\times 1} , \tag{31}$$

$$\mathbf{w}_q = \begin{bmatrix} \mathbf{d}_q(\mathbf{p}) \\ \mathbf{0}_{P\times 1} \end{bmatrix}_{2P\times 1} , \tag{32}$$

$$\mathbf{v} = \begin{bmatrix} \mathbf{v}_0 \\ \vdots \\ \mathbf{v}_{Q-1} \\ \mathbf{0}_{2N\times 1} \end{bmatrix}_{4N\times 1} , \tag{33}$$

and

$$\mathbf{v}_q = \begin{bmatrix} \mathbf{p}_q \\ \mathbf{0}_{P\times 1} \end{bmatrix}_{2P\times 1} , \tag{34}$$

with $\mathbf{d}_q(\mathbf{p})$ and $\mathbf{p}_q$ defined by equations 28 and 29, respectively. Finally $\mathbf{C}$ (equation 30) is a $4N \times 4N$ symmetric Block Circulant matrix with Circulant Blocks (BCCB) (Davis, 1979, p. 184). A detailed description of this matrix and some of its relevant properties are presented in Appendix A.

What follows shows a step-by-step description of how we use the auxiliary system (equation 30) to compute the matrix-vector product $\mathbf{A}\hat{\mathbf{p}}^k$ (equation 7) in a computationally efficient way by exploring the structure of matrix $\mathbf{C}$. By substituting equation A-5 in the auxiliary system (equation 30) and premultiplying both sides of the result by $(\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P})$ (see the details in Appendix A), we obtain

$$\mathbf{\Lambda}\,(\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P})\,\mathbf{v} = (\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P})\,\mathbf{w}\,. \tag{35}$$

Now, by applying the *vec*-operator to both sides of equation 35 (see the details in Appendix B), we obtain:

$$\mathbf{F}_{2Q}^{*}\left[\mathbf{L}\circ\left(\mathbf{F}_{2Q}\,\mathbf{V}\,\mathbf{F}_{2P}\right)\right]\mathbf{F}_{2P}^{*}=\mathbf{W}\;, \tag{36}$$

where "$\circ$" denotes the Hadamard product (Horn and Johnson, 1991, p. 298) and $\mathbf{L}$, $\mathbf{V}$ and $\mathbf{W}$ are $2Q\times 2P$ matrices obtained by rearranging, along their rows, the elements forming the diagonal of matrix $\mathbf{\Lambda}$, vector $\mathbf{v}$ and vector $\mathbf{w}$, respectively. The left side of equation 36 contains the 2D Inverse Discrete Fourier Transform (IDFT) of the term in brackets, which in turn represents the Hadamard product of matrix $\mathbf{L}$ (equation B-11) and the 2D DFT of matrix $\mathbf{V}$ (equation B-9). Matrix $\mathbf{L}$ contains the eigenvalues of $\mathbf{\Lambda}$ (equation A-5) and can be efficiently computed by using only the first column of the BCCB matrix $\mathbf{C}$ (equation 30) (see the details in Appendix C). Here, we evaluate equation 36 and compute matrix $\mathbf{L}$ by using the 2D Fast Fourier Transform (2D FFT). This approach, that have been used in potential-field methods (e.g., Zhang and Wong, 2015; Zhang et al., 2016; Qiang et al., 2019), is actually a fast 2D discrete convolution (e.g., Van Loan, 1992, p. 213).

At each iteration kth of the fast equivalent-layer technique, (equation 7), we efficiently compute $\mathbf{A}\hat{\mathbf{p}}^{k}=\mathbf{d}(\hat{\mathbf{p}}^{k})$ by following the steps below:

(1) Use equation 16 to compute the first column of each block $\mathbf{A}_q$ (equation 18), $q=0,\ldots,Q-1$, forming the BTTB matrix $\mathbf{A}$ (equation 17);

(2) Rearrange the first column of $\mathbf{A}$ according to equations A-1 and A-2 to obtain the first column $\mathbf{c}_0$ of the BCCB matrix $\mathbf{C}$ (equation 30);

(3) Rearrange $\mathbf{c}_0$ along the rows of matrix $\mathbf{G}$ (equation C-3) and use the 2D FFT to compute matrix $\mathbf{L}$ (equation C-3);

(4) Rearrange the parameter vector $\hat{\mathbf{p}}^k$ (equation 1) in its partitioned form (equation 29) to define the auxiliary vector $\mathbf{v}$ (equation 33);

(5) Rearrange $\mathbf{v}$ to obtain matrix $\mathbf{V}$, use the 2D FFT to compute its DFT and evaluate the left side of equation B-12;

(6) Use the 2D FFT to compute the IDFT of the result obtained in step (5) to obtain the matrix $\mathbf{W}$ (equation 36);

(7) Use the *vec*-operator (equation B-2) and equations 31 and 32 to rearrange $\mathbf{W}$ in order to obtain the predicted data vector $\mathbf{d}(\hat{\mathbf{p}}^k)$.

Computational performance

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

$$f_0 = N^{it}(3N + 2N^2) \;, \tag{37}$$

where N^{it} is the number of iterations. In this equation, the term $2N^2$ is associated with the matrix-vector product $\mathbf{A}\hat{\mathbf{p}}^k$ (equation 7) and accounts for most of the computational complexity of this method. Our method replace this matrix-vector product by three operations: one DFT, one Hadamard product and one IDFT involving $2Q \times 2P$ matrices (left side of equation 36). The Hadamard product requires $24N$ flops, $N = QP$, because the entries are complex numbers. We consider that a DFT/IDFT requires $\kappa\, 4N \log_2(4N)$ flops to be computed via 2D FFT, where κ is a constant depending on the algorithm. Then, the resultant flops count of our method is given by:

$$f_1 = N^{it}\left[27N + \kappa\, 8N \log_2(4N)\right] \;. \tag{38}$$

Figure 2 shows the flops counts f_0 and f_1 (equations 37 and 38) associated with the fast equivalent-layer technique (Siqueira et al., 2017) and our method, respectively, as a function of the number N of observation points. We considered a fixed number of $N^{it} = 50$ iterations and $\kappa = 5$ (equation 38), which is compatible with a radix-2 FFT (Van Loan, 1992, p. 16). As we can see, the number of flops is drastically decreased in our method.

Another advantage of our method is concerned with the real $N \times N$ matrix $\mathbf{A}$ (equations 1 and 17). In the fast equivalent-layer technique, the full matrix is computed once and stored during the entire iterative process. On the other hand, our method computes only one column of $\mathbf{A}$ and uses it to compute the complex $2Q \times 2P$ matrix $\mathbf{L}$ (equation C-3) via 2D FFT, which is stored during all iterations. Table 1 shows the RAM memory usage needed to store the full matrix $\mathbf{A}$, a single column of $\mathbf{A}$ and the full matrix $\mathbf{L}$. These quantities were computed for different numbers of observations N. Notice that $N = 1,000,000$ observations require nearly 7.6 Terabytes of RAM memory to store the whole matrix $\mathbf{A}$.

Figure 3 compares the running time of the fast equivalent-layer technique (Siqueira et al., 2017) and of our method, considering a constant number of iterations $N^{it} = 50$. We used a PC with an Intel Core i7 4790@3.6GHz processor and 16 GB of RAM memory. The computational efficiency of our approach is significantly superior to that of the fast equivalent-layer technique for a number of observations N greater than $10,000$. We could not perform this comparison with a number of observations greater than $22,500$ due to limitations of our PC in storing the full matrix $\mathbf{A}$. Figure 4 shows the running time of our method with a number of observations up to 25 millions. These results shows that, while the running time of our method is ≈ 30.9 seconds for $N = 1,000,000$, the fast equivalent-layer technique spends ≈ 46.8 seconds for $N = 22,500$.

APPLICATION TO SYNTHETIC DATA

We have simulated three sources whose horizontal projections are shown in Figure 5 as black lines. These sources are a sphere with density contrast $-1.25\,\mathrm{g/cm^3}$ and two rectangular prisms with density contrasts $1.00\,\mathrm{g/cm^3}$ (upper-left prism) and $1.30\,\mathrm{g/cm^3}$ (upper-right prism). Figure 5 shows the gravity disturbance (vertical component of gravitational attraction) produced by these sources. The synthetic data are contaminated with additive pseudorandom Gaussian noise with zero mean and standard deviation of 0.1 mGal. The data are computed at $N = 10,000$ observation points that are regularly spaced on a 100×100 grid, at $z_1 = -100$ m. We have set a grid of equivalent sources, each one directly beneath each observation point, at $z_0 = 300$ m.

Figure 6a and 6b show the data fits obtained, respectively, by the fast equivalent-layer technique (Siqueira et al., 2017) and by our method. They represent the differences between the simulated data (Figure 5) and the predicted data produced by both methods (not shown) after $N^{it} = 40$ iterations. As we can see, both methods produce virtually the same results. This excellent agreement is confirmed by Figure 6c, which shows the differences between the predicted data produced by both methods.

We performed the upward and downward continuations of the simulated gravity data (Figure 5) by using the fast equivalent-layer technique (Siqueira et al., 2017), our method and also the classical approach in the Fourier domain. This approach consists in performing the upward/downward continuation by directly computing the Fourier transform of the gravity data (e.g., Blakely, 1996, p. 317). Figure 7 shows the upward-continued gravity data obtained by the three methods. As we can see, the residuals between the true data at $z = -300$ m (Figure 7a) and the upward-continued data obtained by using our method

(Figure 7b) and the fast equivalent-layer technique (Figure 7c) are very similar to each other. Notice that the absolute values of the residuals produced by the classical Fourier approach (Figure 7d) are ≈ 10 times greater than those produced by our method and the fast equivalent-layer technique (Figure 7b and 7c), with maximum values concentrated at the border of the simulated area. Differently from the results yield by our method (Figure 7b) and the fast equivalent-layer technique (Figure 7c), that obtained with the classical Fourier approach exhibits a slight noise amplification (Figure 7d).

Figure 8 shows the results obtained by using all methods to compute downward continuation of gravity data. In this case, the maximum absolute values of the residuals produced by the classical Fourier approach (Figure 8d) are ≈ 20 times greater than those produced by our method and the fast equivalent-layer technique (Figure 8b and 8c). This noise amplification is a well-known problem of the downward continuation produced by the classical Fourier approach (e.g., Blakely, 1996, p. 320).

We opted for showing all the results (Figures 7 and 8) produced by all methods without removing the border effects in order to properly compare them to each other. We also stress that no padding function to expand the data was used in applying our method, the fast equivalent-layer technique or the Fourier approach. Another important aspect to be pointed out about these results is the computational times spent by our method and the fast equivalent-layer technique. The total computational time required by our method to estimate the physical-property distribution on the equivalent layer and to perform the upward/downward continuations is about two orders of magnitude lower than that spent by the fast equivalent-layer technique. This significant reduction in computational time was obtained by using a data set composed of $N = 10,000$ observation points. Considerably better results can be obtained with larger data sets.

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

APPLICATION TO FIELD DATA

We applied our method to airborne gravity data from Carajás, north of Brazil, which were provided by the Geological Survey of Brazil (CPRM). The data were collected along 131 north-south flight lines separated by 3 km and 29 east-west tie lines separated by 12 km. This data set was divided in two different areas, collected in different times, having samples spacing of 7.65 m and 15.21 m along the lines, totalizing $5,492,551$ observation points at a fixed height of 900 m ($z_1 = -900$ m). The gravity data were interpolated (Figure 9) into a regularly spaced grid of 500×500 observation points ($N = 250,000$) with a grid spacing of ≈ 717 m north-south and ≈ 782 m east-west.

To apply our method, we set an equivalent layer at $z_0 = 300$ m. Figure 10a shows the predicted data obtained with our method after $N^{it} = 50$ iterations. The residuals (Figure 10b), defined as the difference between the observed (Figure 9) and predicted (Figure 10a) data, show a very good data fit with mean close to zero (0.0003 mGal) and small standard deviation (0.1160 mGal), which corresponds to approximately 0.1 % of the maximum amplitude of the gravity data. By using the estimated mass distribution (not shown), we performed an upward-continuation of the observed gravity data to a horizontal plane located $5,000$ m above. Figure 11 shows a very consistent upward-continued gravity data, with a clear attenuation of the short wavelengths. By using our approach, the processing of the $250,000$ observations took only 0.216 seconds.

CONCLUSIONS

We show that the sensitivity matrix associated with the equivalent layer technique for processing gravity data has a BTTB structure when the following conditions are satisfied: (i) the observed data are disposed at a regular grid on a horizontal plane and (ii) the equivalent sources (point masses) are placed at a constant depth, one directly beneath each observation point. By exploring the BTTB structure of the sensitivity matrix, we formulate the forward modeling as a FFT convolution that requires only one column of the sensitivity matrix; hence, the forward modeling uses, in practice, a single equivalent source, which results in a drastic reduction of the required RAM memory. Next, we propose an efficient approach for optimizing the computational time of an iterative method called fast equivalent-layer technique. This iterative method is grounded on the excess mass constraint and does not demand the solution of linear systems. Compared to the fast equivalent-layer technique, our approach greatly reduces the number of flops and the RAM memory necessary to estimate a 2D mass distribution within a planar equivalent layer that fits the observed gravity data. For example, the number of flops is reduced by two orders of magnitude when processing one million of observations by using our approach. Traditionally, such amount of data impractically requires 7.6 Terabytes of RAM memory to handle the full sensitivity matrix whereas our method takes only 61.035 Megabytes. We have successfully applied the proposed method to compute the upward/downward continuations of synthetic gravity data. Our results show that our method produces considerably smaller border effects and noise amplification compared to the classical Fourier approach, which computes the Fourier transform of the observed data. Application to field data over the Carajás Province, north of Brazil, confirms the potential of our approach in processing an airborne gravity data set with $250,000$ observations in approximately 0.2 seconds. Our method can also be applied

to process large marine gravity data sets. Further studies need to be carried out in order to apply our method to process other potential-field data.

APPENDIX A

MATRIX C

This appendix illustrates the matrix $\mathbf{C}$ (equation 30) obtained with the x- and y-oriented grids illustrated in Figure 1 and also presents some of its relevant properties.

Matrix $\mathbf{C}$ (equation 30) is circulant blockwise, formed by $2Q \times 2Q$ blocks, where each block $\mathbf{C}_q$, $q = 0, \ldots, Q-1$, is a $2P \times 2P$ circulant matrix. Similarly to the BTTB matrix $\mathbf{A}$ (equations 17 and 24–27), the index q varies from 0 to $Q-1$. Additionally, the blocks lying above the main diagonal are equal to those located below.

It is well-known that a circulant matrix can be defined by properly downshifting its first column (Van Loan, 1992, p. 206). Hence, the BCCB matrix $\mathbf{C}$ (equation 30) can be obtained from its first column of blocks, which is given by

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

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

obtained by downshifting its first column

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

where a_p^q (equation 23), $p = 0, \ldots, P-1$, are the elements forming the block $\mathbf{A}_q$ (equations 18 and 24–27). The downshift can be thought off as permutation that pushes the components of a column vector down one notch with wraparound (Golub and Loan, 2013, p. 20). To illustrate this operation, consider our y-oriented grid illustrated in Figure 1b. In this case, the resulting BCCB matrix $\mathbf{C}$ (equation 30) is given by

$$\mathbf{C} = \begin{bmatrix} \mathbf{C_0} & \mathbf{C_1} & \mathbf{C_2} & \mathbf{C_3} & \mathbf{0} & \mathbf{C_3} & \mathbf{C_2} & \mathbf{C_1} \\ \mathbf{C_1} & \mathbf{C_0} & \mathbf{C_1} & \mathbf{C_2} & \mathbf{C_3} & \mathbf{0} & \mathbf{C_3} & \mathbf{C_2} \\ \mathbf{C_2} & \mathbf{C_1} & \mathbf{C_0} & \mathbf{C_1} & \mathbf{C_2} & \mathbf{C_3} & \mathbf{0} & \mathbf{C_3} \\ \mathbf{C_3} & \mathbf{C_2} & \mathbf{C_1} & \mathbf{C_0} & \mathbf{C_1} & \mathbf{C_2} & \mathbf{C_3} & \mathbf{0} \\ \mathbf{0} & \mathbf{C_3} & \mathbf{C_2} & \mathbf{C_1} & \mathbf{C_0} & \mathbf{C_1} & \mathbf{C_2} & \mathbf{C_3} \\ \mathbf{C_3} & \mathbf{0} & \mathbf{C_3} & \mathbf{C_2} & \mathbf{C_1} & \mathbf{C_0} & \mathbf{C_1} & \mathbf{C_2} \\ \mathbf{C_2} & \mathbf{C_3} & \mathbf{0} & \mathbf{C_3} & \mathbf{C_2} & \mathbf{C_1} & \mathbf{C_0} & \mathbf{C_1} \\ \mathbf{C_1} & \mathbf{C_2} & \mathbf{C_3} & \mathbf{0} & \mathbf{C_3} & \mathbf{C_2} & \mathbf{C_1} & \mathbf{C_0} \end{bmatrix}_{4N\times 4N}, \tag{A-3}$$

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

$$\mathbf{C}_q = \begin{bmatrix} a_0^q & a_1^q & a_2^q & 0 & a_2^q & a_1^q \\ a_1^q & a_0^q & a_1^q & a_2^q & 0 & a_2^q \\ a_2^q & a_1^q & a_0^q & a_1^q & a_2^q & 0 \\ 0 & a_2^q & a_1^q & a_0^q & a_1^q & a_2^q \\ a_2^q & 0 & a_2^q & a_0^q & a_0^q & a_1^q \\ a_1^q & a_2^q & 0 & a_2^q & a_1^q & a_0^q \end{bmatrix}_{2P \times 2P} \tag{A-4}$$

in terms of the block elements a_p^q (equation 23). Similar matrices are obtained for our x-oriented grid illustrated in Figure 1a.

BCCB matrices are diagonalized by the 2D unitary DFT (Davis, 1979, p. 185). It means that $\mathbf{C}$ (equation 30) satisfies

$$\mathbf{C} = (\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P})^* \, \mathbf{\Lambda} \, (\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P}) \ , \tag{A-5}$$

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

APPENDIX B

COMPUTATIONS WITH THE 2D DFT

In the present Appendix, we deduce equation 36 by using the row-ordered *vec*-operator (here designated simply as *vec*-operator). This equation can be efficiently computed by using the 2D fast Fourier Transform. This operator was implicitly used by Jain (1989, p. 31) to show the relationship between Kronecker products and separable transformations. The *vec*-operator defined here transforms a matrix into a column vector by stacking its rows.

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

$$\mathbf{M} = \begin{bmatrix} \mathbf{m}_1^\top \\ \vdots \\ \mathbf{m}_N^\top \end{bmatrix} , \tag{B-1}$$

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

$$vec\,(\mathbf{M}) = \begin{bmatrix} \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_N \end{bmatrix}_{NM \times 1} . \tag{B-2}$$

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

Two important properties of the *vec*-operator (equation B-2) are necessary to us. To define the first one, consider an $N \times M$ matrix $\mathbf{H}$ given by

$$\mathbf{H} = \mathbf{P} \circ \mathbf{Q} , \tag{B-3}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are arbitrary $N \times M$ matrices and "$\circ$" represents the Hadamard product (Horn and Johnson, 1991, p. 298). By applying the *vec*-operator to $\mathbf{H}$ (equation B-3), it can be shown that

$$vec\,(\mathbf{H}) = vec\,(\mathbf{P}) \circ vec\,(\mathbf{Q}) \ . \tag{B-4}$$

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

$$\mathbf{S} = \mathbf{P}\,\mathbf{M}\,\mathbf{Q}\,, \tag{B-5}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are arbitrary $N \times N$ and $M \times M$ matrices, respectively. By implicitly applying the *vec*-operator to the $\mathbf{S}$ (equation B-5), Jain (1989, p. 31) showed that:

$$vec\,(\mathbf{S}) = \left(\mathbf{P} \otimes \mathbf{Q}^{\top}\right) vec\,(\mathbf{M}) \ , \tag{B-6}$$

where "$\otimes$" denotes the Kronecker product (Neudecker, 1969). It is important to stress the difference between equation B-6 and that presented by Neudecker (1969), which is more commonly found in the literature. While that equation uses a *vec*-operator that transforms a matrix into a column vector by stacking its columns, equation B-6 uses the *vec*-operator defined by equation B-2, which transforms a matrix into a column vector by stacking its rows.

Now, let us deduce equation 36 by using the above-defined properties (equation B-4 and B-6). We start calling attention to the right side of equation 35. Consider that vector $\mathbf{w}$ (equation 35) is obtained by applying the *vec*-operator (equation B-2) to a matrix $\mathbf{W}$, whose 2D DFT $\tilde{\mathbf{W}}$ is represented by the following separable transformation (Jain, 1989, p. 146):

$$\tilde{\mathbf{W}} = \mathbf{F}_{2Q}\,\mathbf{W}\,\mathbf{F}_{2P}\,, \tag{B-7}$$

where $\mathbf{F}_{2Q}$ and $\mathbf{F}_{2P}$ are the $2Q \times 2Q$ and $2P \times 2P$ unitary DFT matrices. Using equation B-6 and the symmetry of unitary DFT matrices, we rewrite the right side of equation 35 as follows:

$$vec\left(\tilde{\mathbf{W}}\right) = \left(\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P}\right) vec\left(\mathbf{W}\right) \ . \tag{B-8}$$

Similarly, consider that $\mathbf{v}$ (equation 35) is obtained by applying the *vec*-operator (equation B-2) to a matrix $\mathbf{V}$, whose 2D DFT (equation B-7) is represented by $\tilde{\mathbf{V}}$. Using equation B-6 and the symmetry of unitary DFT matrices, we can rewrite the left side of equation 35 as follows:

$$\mathbf{\Lambda}\, vec\left(\tilde{\mathbf{V}}\right) = \mathbf{\Lambda}\left(\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P}\right) vec\left(\mathbf{V}\right) \ . \tag{B-9}$$

Note that both sides of equation B-9 are defined as the product of the diagonal matrix $\mathbf{\Lambda}$ (equation A-5) and a vector. In this case, the matrix-vector product can be conveniently replaced by

$$\boldsymbol{\lambda} \circ vec\left(\tilde{\mathbf{V}}\right) = \boldsymbol{\lambda} \circ \left(\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P}\right) vec\left(\mathbf{V}\right) \ , \tag{B-10}$$

where $\boldsymbol{\lambda}$ is a $4QP \times 1$ vector containing the diagonal of $\mathbf{\Lambda}$ (equation A-5). Then, consider that $\boldsymbol{\lambda}$ is obtained by applying the *vec*-operator (equation B-2) to a $2Q \times 2P$ matrix $\mathbf{L}$, we can use equations B-4 and B-6 to rewrite equation B-10 as follows:

$$vec\left(\mathbf{L} \circ \tilde{\mathbf{V}}\right) = vec\left[\mathbf{L} \circ \left(\mathbf{F}_{2Q}\, \mathbf{V}\, \mathbf{F}_{2P}\right)\right] \ . \tag{B-11}$$

Equations B-7, B-8 and B-11 show that equation 35 is obtained by applying the *vec*-operator to

$$\mathbf{L} \circ \left(\mathbf{F}_{2Q}\, \mathbf{V}\, \mathbf{F}_{2P}\right) = \mathbf{F}_{2Q}\, \mathbf{W}\, \mathbf{F}_{2P} \, . \tag{B-12}$$

Finally, we premultiply both sides of equation B-12 by $\mathbf{F}_{2Q}^{*}$ and then postmultiply both sides of the result by $\mathbf{F}_{2P}^{*}$ to deduce equation 36.

APPENDIX C

THE EIGENVALUES OF C

In the present Appendix, we show how to efficiently compute matrix $\mathbf{L}$ (equations B-11, B-12 and 36) by using only the first column of the BCCB matrix $\mathbf{C}$ (equation 30).

We need first premultiply both sides of equation A-5 by $(\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P})$ to obtain

$$(\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P})\, \mathbf{C} = \mathbf{\Lambda}\, (\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P}) \ . \tag{C-1}$$

From equation C-1, we can easily show that (Chan and Jin, 2007, p. 77):

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

where $\mathbf{c}_0$ is a $4QP \times 1$ vector representing the first column of $\mathbf{C}$ (equation 30) and $\boldsymbol{\lambda}$ (equation B-10) is the $4QP \times 1$ vector that contains the diagonal of matrix $\mathbf{\Lambda}$ (equation A-5) and is obtained by applying the *vec*-operator (equation B-2) to matrix $\mathbf{L}$. Now, let us conveniently consider that $\mathbf{c}_0$ is obtained by applying the *vec*-operator to a $2Q \times 2P$ matrix $\mathbf{G}$. Using this matrix, the property of the *vec*-operator for separable transformations (equation B-5) and the symmetry of unitary DFT matrices, equation C-2 can be rewritten as follows

$$\mathbf{F}_{2Q}\, \mathbf{G}\, \mathbf{F}_{2P} = \frac{1}{\sqrt{4QP}}\, \mathbf{L}\,. \tag{C-3}$$

This equation shows that the eigenvalues of the BCCB matrix $\mathbf{C}$ (equation 30), forming the rows of $\mathbf{L}$, are obtained by computing the 2D DFT of matrix $\mathbf{G}$, which contains the elements forming the first column of the BCCB matrix $\mathbf{C}$ (equation 30).

REFERENCES

Barnes, G., and J. Lumley, 2011, Processing gravity gradient data: GEOPHYSICS, **76**, I33–I47.

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

Boggs, D., and M. Dransfield, 2004, Analysis of errors in gravity derived from the falcon airborne gravity gradiometer: ASEG-PESA Airborne Gravity 2004 Workshop, Geoscience Australia Record, 135–141.

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

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.

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.

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

Graham, L., D. E. Knuth, and O. Patashnik, 1994, Concrete mathematics: a foundation for computer science, 2 ed.: Addison-Wesley Publishing Company.

Guspí, F., and I. Novara, 2009, Reduction to the pole and transformations of scattered magnetic data using newtonian equivalent sources: GEOPHYSICS, **74**, L67–L73.

Hansen, R. O., and Y. Miyazaki, 1984, Continuation of potential fields between arbitrary surfaces: GEOPHYSICS, **49**, 787–795.

Horn, R. A., and C. R. Johnson, 1991, Topics in matrix analysis, 1 ed.: Cambridge University Press.

Jain, A. K., 1989, Fundamentals of digital image processing, 1 ed.: Pearson.

Kellogg, O. D., 1929, Foundations of potential theory: Frederick Ungar Publishing Company.

Leão, J. W. D., and J. B. C. Silva, 1989, Discrete linear transformations of potential field data: GEOPHYSICS, **54**, 497–507.

Li, Y., and D. W. Oldenburg, 2010, Rapid construction of equivalent sources using wavelets: GEOPHYSICS, **75**, L51–L59.

Mendonça, C. A., and J. B. C. Silva, 1994, The equivalent data concept applied to the interpolation of potential field data: GEOPHYSICS, **59**, 722–732.

Neudecker, H., 1969, Some theorems on matrix differentiation with special reference to kronecker matrix products: Journal of the American Statistical Association, **64**, 953–963.

Oliveira Jr., V. C., V. C. F. Barbosa, and L. Uieda, 2013, Polynomial equivalent layer: GEOPHYSICS, **78**, G1–G13.

Qiang, J., W. Zhang, K. Lu, L. Chen, Y. Zhu, S. Hu, and X. Mao, 2019, A fast forward algorithm for three-dimensional magnetic anomaly on undulating terrain: Journal of Applied Geophysics, **166**, 33 – 41.

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

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**, G57–G69.

Van Loan, C., 1992, Computational frameworks for the fast fourier transform: SIAM.

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

Xia, J., D. R. Sprowl, and D. Adkins-Heljeson, 1993, Correction of topographic distortions in potential-field data; a fast and accurate approach: GEOPHYSICS, **58**, 515–523.

Zhang, Y., and Y. S. Wong, 2015, BTTB-based numerical schemes for three-dimensional gravity field inversion: Geophysical Journal International, **203**, 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

LIST OF FIGURES

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

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

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

10 Application to field data over the Carajás Province, Brazil. (a) Predicted data produced by our method. (b) Residuals between the observed (Figure 9) and the predicted data (panel a), with mean 0.000292 mGal and standard deviation of 0.105 mGal.

11 Application to field data over the Carajás Province, Brazil. The upward-continued gravity data obtained with our method $5,000$ m above the observed data (Figure 9). The

total computation time for processing of the $250,000$ observations was 0.216 seconds.

N	**Full matrix A**	**Single column of A**	**Full matrix L**
100	0.0763	0.000763	0.00610
400	1.22	0.0031	0.0248
$2,500$	48	0.0191	0.1528
$10,000$	763	0.0763	0.6104
$40,000$	12,207	0.305	2.4416
$250,000$	476,837	1.907	15.3
$500,000$	1,907,349	3.815	30.518
$1,000,000$	7,629,395	7.629	61.035

Table 1: Comparison between the system RAM memory usage needed to store the full matrix **A** (equations 1 and 17), a single column of **A** and the full matrix **L** (equation **??**). The quantities are represented in megabyte (MB), for different total number of observations N. We consider that the elements of **A** and **L** occupy, respectively, 8 and 16 bytes each in computer memory.

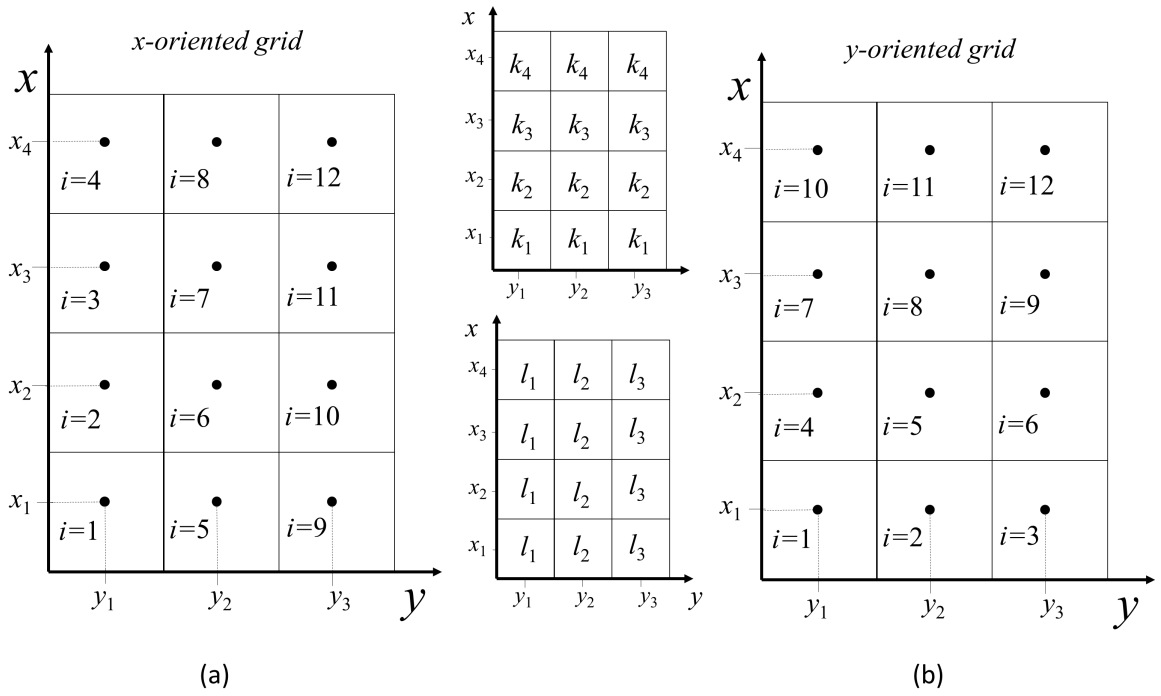

Figure 1: Schematic representation of an $N_x \times N_y$ regular grid of points (black dots) defined by $N_x = 4$ and $N_y = 3$. The grids are oriented along the (a) x-axis and (b) y-axis. The grid coordinates are x_k and y_k, where the $k = 1, \ldots, N_x$ and $l = 1, \ldots, N_y$ are called the grid indices. The insets show the grid indices k and l.

– **GEO-XXXX**

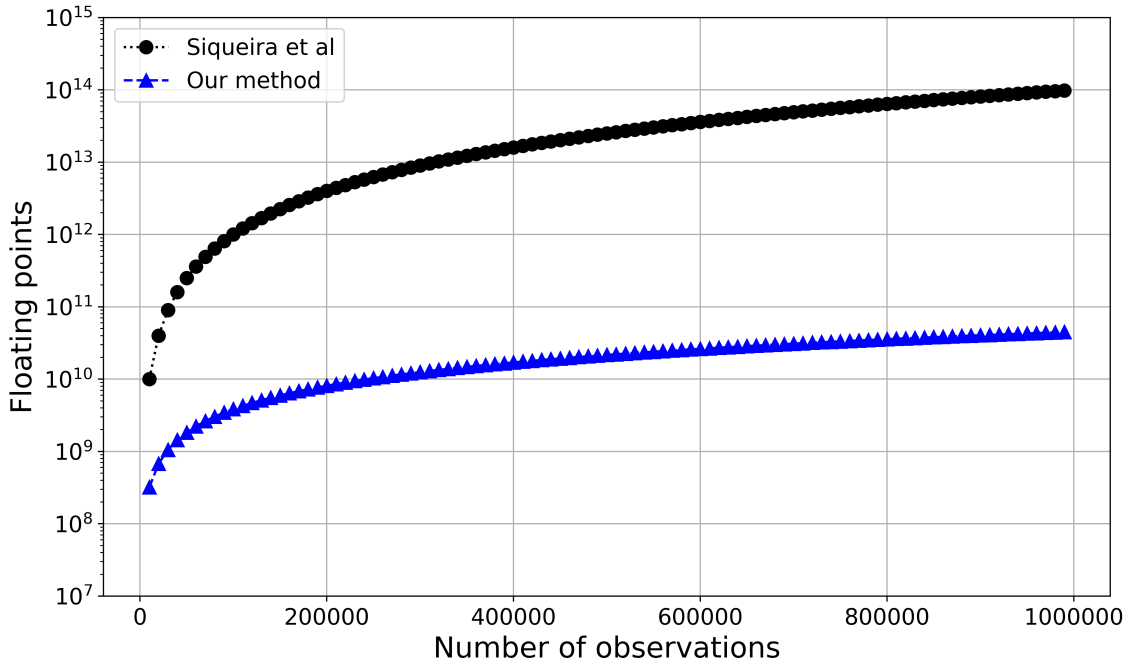

Figure 2: Comparison between the number of flops (equations 37 and 38) associated with the fast equivalent-layer technique (Siqueira et al., 2017) and our method, for N varying from $5,000$ to $1,000,000$. All values are computed with $N^{it} = 50$ iterations and $\kappa = 5$.

– **GEO-XXXX**

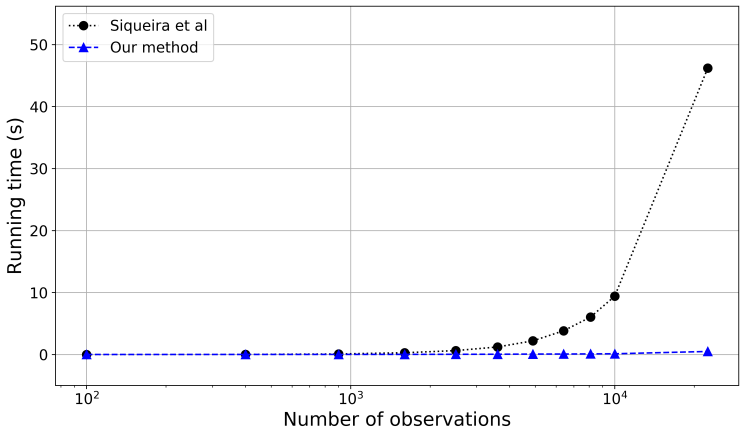

Figure 3: Comparison between the running time of the fast equivalent-layer technique (Siqueira et al., 2017) and our method. The values were obtained for $N^{it} = 50$ iterations.

– **GEO-XXXX**

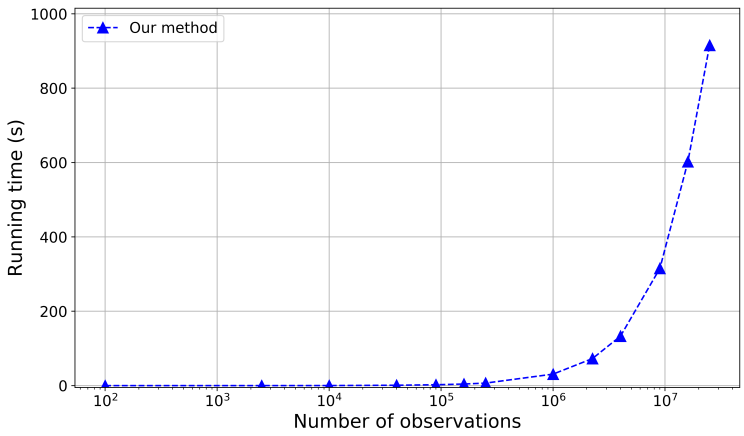

Figure 4: Running time of our method for a number of observations N up to 25 millions. The values were obtained for $N^{it} = 50$ iterations.

– **GEO-XXXX**

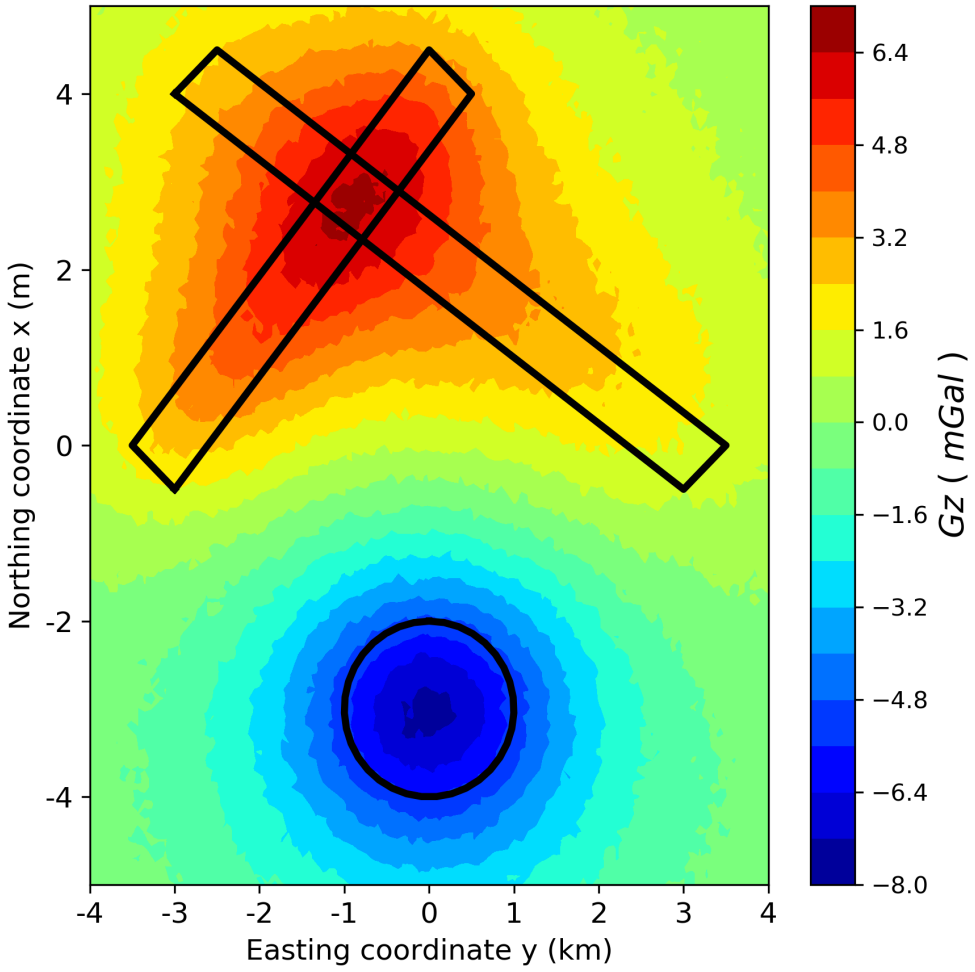

Figure 5: Application to synthetic data. Noise-corrupted gravity data (in color map) produced by three synthetic sources: a sphere with density contrast $-1.25\,\mathrm{g/cm^3}$ and two rectangular prisms with density contrasts $1.00\,\mathrm{g/cm^3}$ (upper-left body) and $1.30\,\mathrm{g/cm^3}$ (upper-right body). The black lines represent the horizontal projection of the sources.

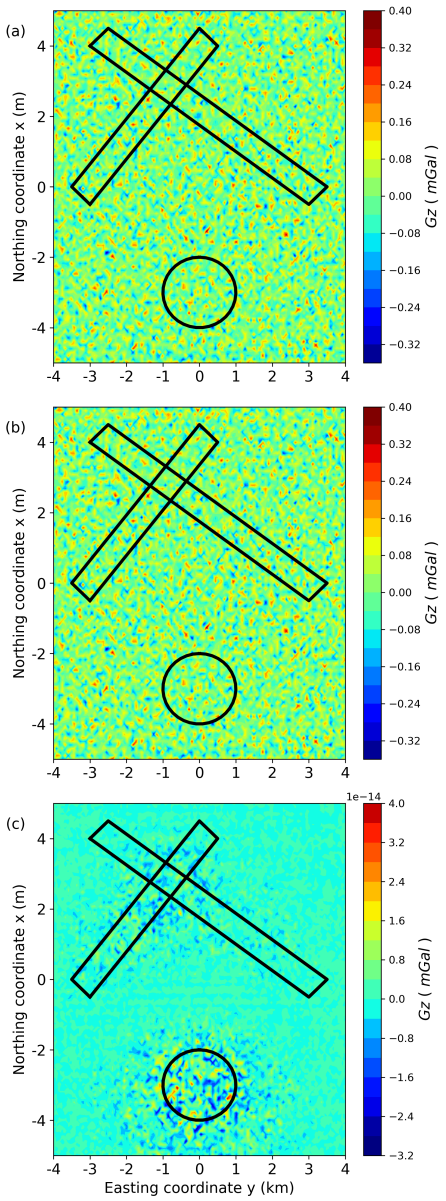

Figure 6: Application to synthetic data. Residuals between the simulated data (Figure 5) and predicted data produced by: (a) the fast equivalent-layer technique (Siqueira et al., 2017) and (b) our method. The mean (-4.493×10^{-5} mGal) and standard deviation (0.093 mGal) for residuals shown in a and b are exactly the same. (c) Difference between the predicted data produced by the fast equivalent-layer technique and our method. The computation times spent by the fast equivalent-layer technique and our method were 10.416 and 0.177 seconds, respectively.

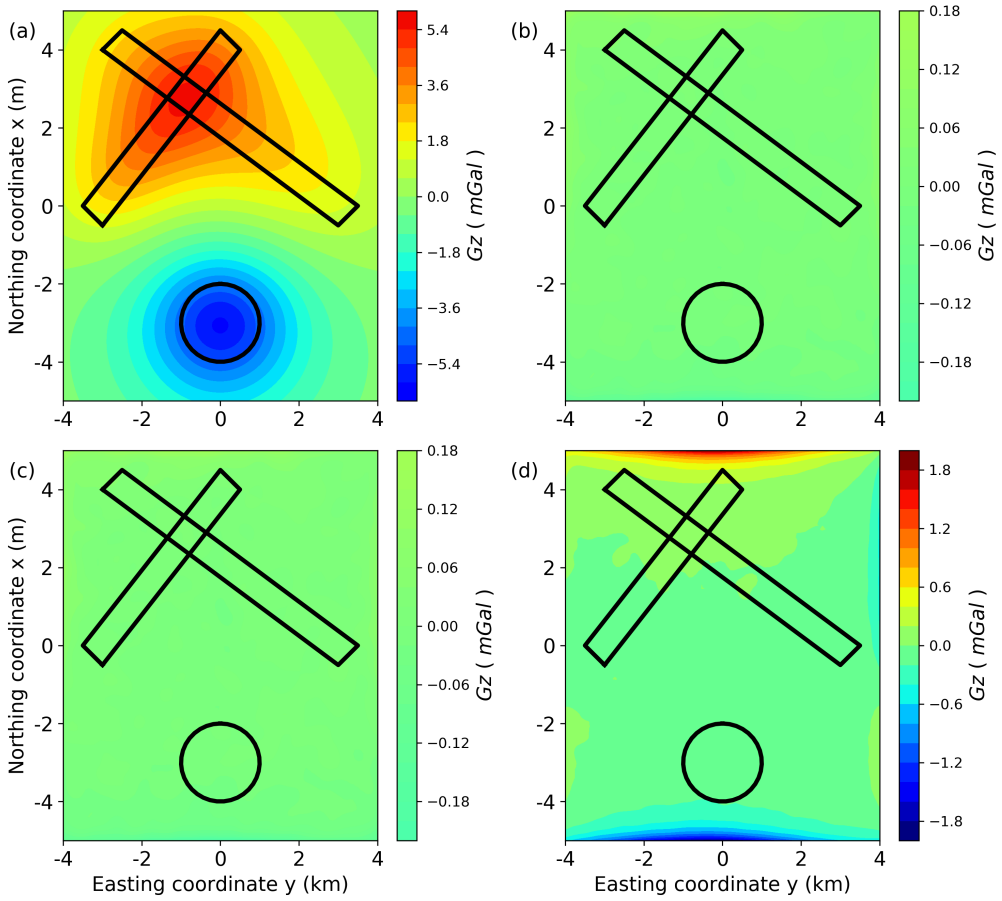

Figure 7: Application to synthetic data. (a) Noise-free gravity data produced by the synthetic sources at $z = -300$ m. Residuals between the data shown in a and the upward-continued data obtained by: (b) our method (not shown), with mean 0.003 mGal and standard deviation 0.034 mGal, (c) the fast equivalent-layer technique (not shown), with mean 0.003 mGal and standard deviation 0.034 mGal and (d) the classical Fourier approach (not shown), with mean −0.030 mGal and standard deviation 0.262 mGal. Values shown in b, c and d have the same scale. The computation times spent by the fast equivalent-layer technique and our method were 8.697 and 0.005 seconds, respectively.

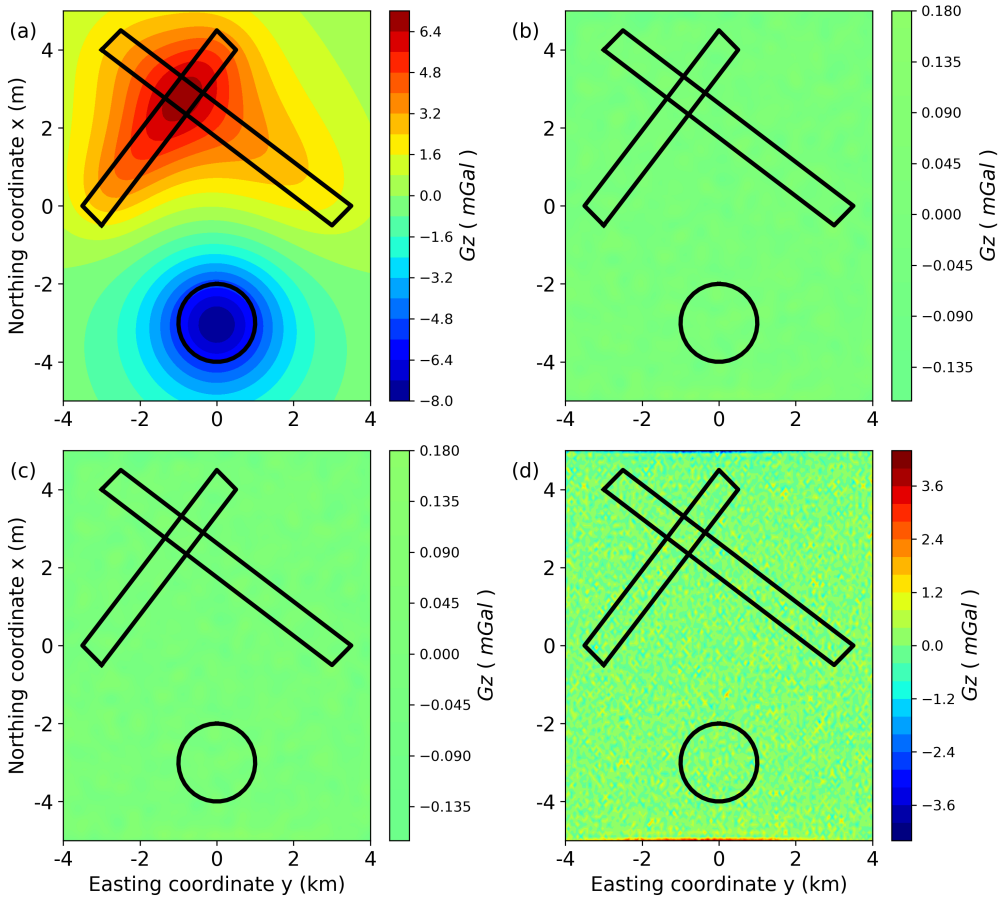

Figure 8: Application to synthetic data. (a) Noise-free gravity data produced by the synthetic sources at $z = -50$ m. Residuals between the data shown in a and the downward-continued data obtained by: (b) our method (not shown), with mean -0.001 mGal and standard deviation 0.038 mGal, (c) the fast equivalent-layer technique (not shown), with mean -0.001 mGal and standard deviation 0.038 mGal and (d) the classical Fourier approach (not shown), with mean -0.030 mGal and standard deviation 0.262 mGal. Values shown in b, c and d have the same scale. The computation times spent by the fast equivalent-layer technique and our method were 8.795 and 0.004 seconds, respectively.

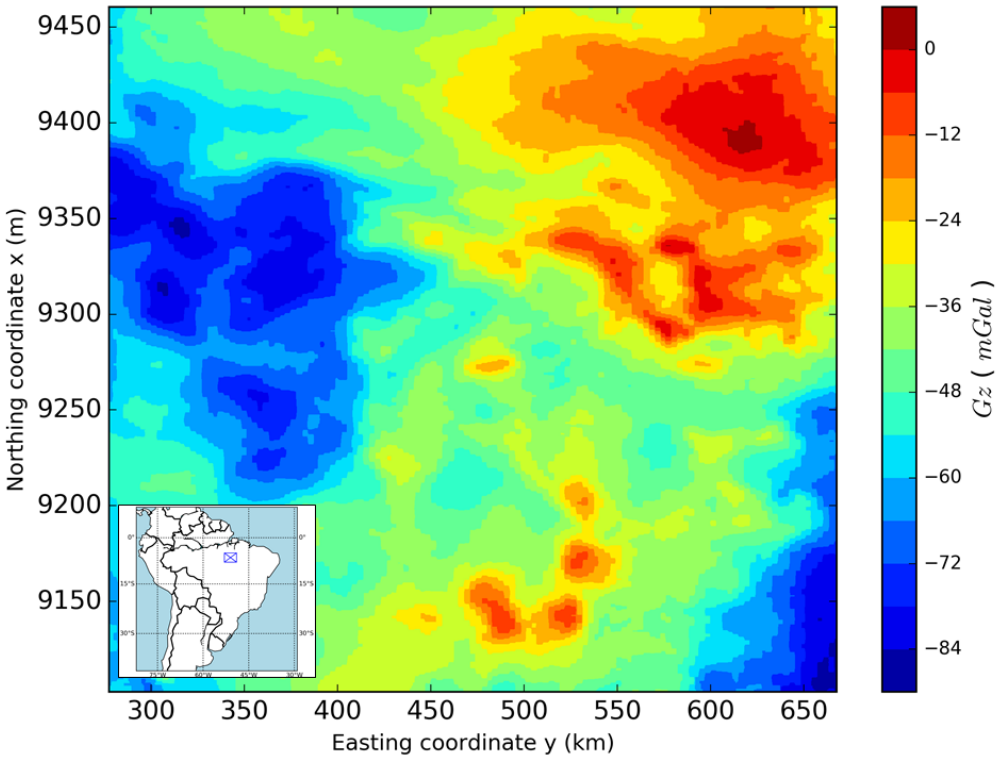

Figure 9: Application to field data over the Carajás Province, Brazil. Observed gravity data on a regular grid of 500×500 points, totaling $N = 250,000$ observations. The inset shows the study area (blue rectangle) which covers the southeast part of the state of Pará, north of Brazil.

– **GEO-XXXX**

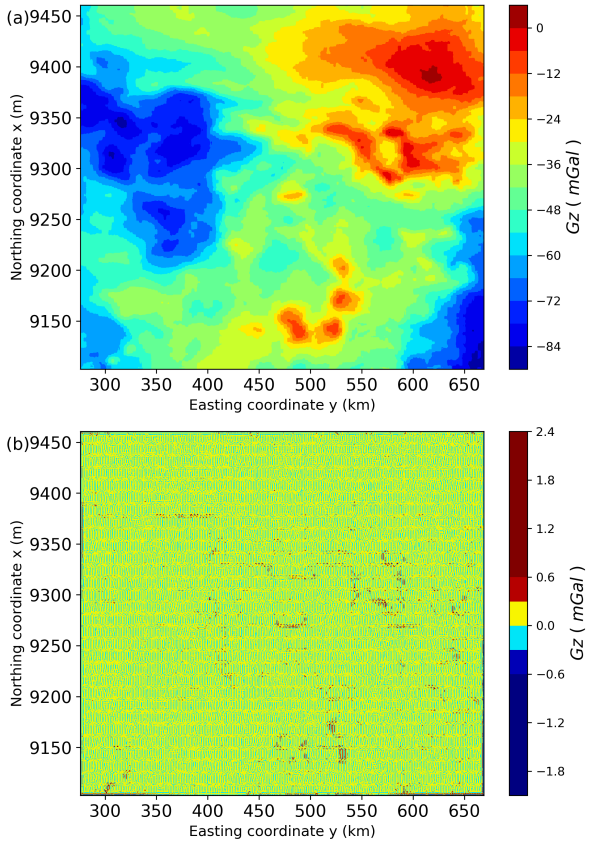

Figure 10: Application to field data over the Carajás Province, Brazil. (a) Predicted data produced by our method. (b) Residuals between the observed (Figure 9) and the predicted data (panel a), with mean 0.000292 mGal and standard deviation of 0.105 mGal.

– **GEO-XXXX**

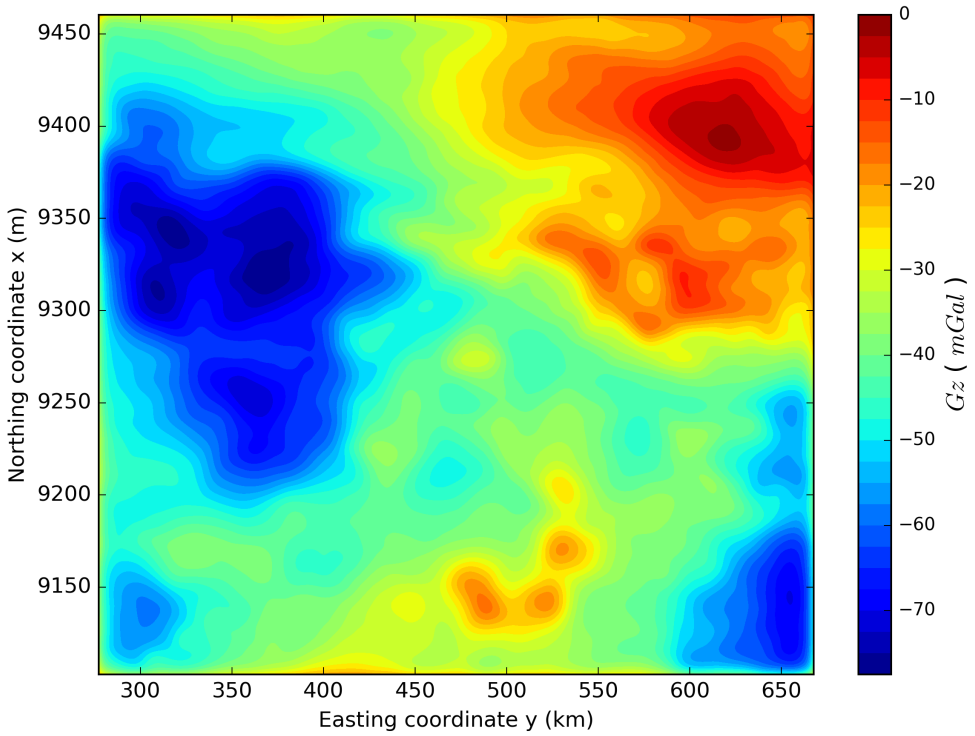

Figure 11: Application to field data over the Carajás Province, Brazil. The upward-continued gravity data obtained with our method $5,000$ m above the observed data (Figure 9). The total computation time for processing of the $250,000$ observations was 0.216 seconds.

– **GEO-XXXX**