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 of 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 two-dimensional 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. Synthetic tests demonstrate the ability of our method to satisfactorily upward- and downward-continuing the gravity data. For example, while our method requires 26:8 seconds to run one million of observations, the fast equivalent-layer technique required 48:3 seconds to run 22,500 observations. Test with real data from Carajás, Brazil, shows its applicability to process very large dataset at low computational cost.

INTRODUCTION

The equivalent layer is a well-known technique for processing potential-field data in applied geophysics since 1960. 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 yield an acceptable potential-field data fit. These fictitious sources are called equivalent sources.

Many previous works have used the equivalent layer as a processing technique of potential-field data. Dampney (1969) used the equivalent-layer technique for gridding and for computing the upward continuation of the potential-field data. Cordell (1992) and Mendonça and Silva (1994) used it for interpolating and gridding potential-field data. Emilia (1973), Hansen and Miyazaki (1984) and Li and Oldenburg (2010) used it for upward continuation of the potential-field data. Silva (1986), Leão and Silva (1989), Guspí and Novara (2009), and Oliveira Jr. et al. (2013) used it for reducing the magnetic data to the pole. Boggs and Dransfield (2004) used it for combining multiple data sets and Barnes and Lumley (2011) for gradient-data processing.

The classic equivalent-layer formulation consists in estimating the physical-property distribution within a layer, composed by a set of equivalent sources, by solving a linear system of equations formed by harmonic functions (e.g.,the inverse of the distance between the observation point and the equivalent source). When these observation points and equivalent sources are regularly spaced, a Toeplitz system arises. Toeplitz systems are well-known in many branches of science as in (1) mathematics, for solving partial and ordinary diferential equations (e.g., Lin et al., 2003); (2) image processing (e.g., Chan et al., 1999) and; (3) computational neuroscience (e.g., Wray and Green, 1994). Jin (2003) and Chan and Jin (2007) give many examples of applications for Toeplitz systems.

In potential-field methods, the properties of Toeplitz system were used for downward continuation (Zhang et al., 2016) and for 3D gravity-data inversion using a 2D multilayer model (Zhang and Wong, 2015). In the particular case of gravity data, the kernel generates a linear system with a matrix known as symmetric Block-Toeplitz Toeplitz-Block (BTTB).

A wide variety of applications in mathematics and engineers that fall into Toeplitz systems propelled the development of a large variety of methods for solving them. Grenander and Szegö (1958) noticed that a circulant matrix can be diagonalized by taking the Fast Fourier Trasform (FFT) of its first column, making it possible to calculate the matrix-vector product and solve the system with low computational cost (Strang and Aarikka, 1986; Olkin, 1986). Chan and Jin (2007) show some preconditioners to embedd the Toeplitz and BTTB matrices into, respectively, circulant matrices and Block-Circulant Circulant-Block (BCCB) by solving the system applying the conjugate gradient method.

Although the use of the equivalent-layer technique increased over the last decades, one of the biggest problem is still its high computational cost for processing large-data sets. Siqueira et al. (2017) developed a computationally efficient scheme for processing gravity data. This scheme does not solve a linear system, instead uses an iterative process that corrects the physical-property distribution over the equivalent layer by adding mass corrections that are proportional to the gravity residual. Although efficient, the method presented by Siqueira et al. (2017) requires, at each iteration, the full computation of the forward problem to guarantee the convergence of the algorithm. The time spent on forward modeling accounts for most of the total computation time of their method.

We propose the use of BTTB and BCCB matrices properties to efficiently solve the forward modeling in method of Siqueira et al. (2017), resulting in a very faster parameter estimation and the possibility to use very large datasets. Here, we show how the system memory (RAM) usage can be drastically decreased by calculating only the first column of the BTTB matrix and embedding into a BCCB matrix. Using the Szegö theorema combined with Strang and Aarikka (1986), the matrix-vector product can be accomplished with very low cost, reducing in some orders of magnitude the number of operations required to complete the process. We present synthetic tests to validate our proposal and real field data from Carajás, Brazil to demonstrate its applicability.

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 compose 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 coincide 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

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

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 coordinates of the observations and the jth coordinates of the equivalent sources. 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. Those authors showed that the fast equivalent-layer technique has a better computational efficiency than the classical equivalent layer approach (equation 3) if the dataset totalize 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 their method only up to 22 500 observation points. Hence, for very large data sets it is costful and can be overwhelming 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 from 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 j-th 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:

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

where $\Delta z = z_0 - z_1$, $\Delta k_{ij} = k(i) - k(j)$ (equations 12 and 14) and $\Delta l_{ij} = l(i) - l(j)$ (equations 11 and 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 $\mathbf{A}_q$, $q = 0, \dots, Q-1$, 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, $p = 0, \dots, P-1$, 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 and 21) and p (equations 20 and 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 and 21) and p (equations 20 and 22).

It is important noting that different matrix indices i or j produce the same absolute values for the grid indices k (equations 12 and 14) and l (equations 11 and 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 and 21) and p (equations 20 and 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 and 14) and l (equations 11 and 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, and 21) and p (equations 20 and and 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 at the upper right corner of $\mathbf{A}$ (equation 1), in the 2nd line and 10th column. 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 $j = 10$. 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^q_p (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

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

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

$$\mathbf{A}_q = \begin{bmatrix} a^q_0 & a^q_1 & a^q_2 & a^q_3 \\ a^q_1 & a^q_0 & a^q_1 & a^q_2 \\ a^q_2 & a^q_1 & a^q_0 & a^q_1 \\ a^q_3 & a^q_2 & a^q_1 & a^q_0 \end{bmatrix} , \tag{25}$$

with elements a^q_p (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

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

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

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

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 this 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 lessen 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 two-dimensional 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 $\mathbf{p}_q$, $q = 0, \dots, Q-1$, 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). What follows aims at showing 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 $\mathbf{C}$.

Matrix $\mathbf{C}$ (equation 30) is circulant blockwise, formed by $Q \times Q$ 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 defines the block diagonal where $\mathbf{C}_q$ lies within $\mathbf{C}$. This index varies from 0, at the main diagonal, to $Q-1$, at the corners of $\mathbf{C}$. Additionally, the blocks lying above the main diagonal are equal to those located below. It is known that a circulant matrix can be defined by properly downshifting its first column (Golub and Loan, 2013, p. 221). Hence, the BCCB matrix $\mathbf{C}$ (equation 30) can be obtained from its first column of blocks, which is given by

$$[\mathbf{C}]_{(0)} = \begin{bmatrix} \mathbf{C}_0 \\ \vdots \\ \mathbf{C}_{Q-1} \\ \mathbf{0} \\ \mathbf{C}_{Q-1} \\ \vdots \\ \mathbf{C}_1 \end{bmatrix}_{4N \times 2P} , \tag{35}$$

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

$$\mathbf{c}_0^q = \begin{bmatrix} a_0^q \\ \vdots \\ a_{P-1}^q \\ 0 \\ a_{P-1}^q \\ \vdots \\ a_1^q \end{bmatrix}_{2P \times 1} , \tag{36}$$

where a_p^q (equation 23), $p = 0, \dots, P-1$, 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

$$\mathbf{C} = \begin{bmatrix} \mathbf{C_0} & \mathbf{C_1} & \mathbf{C_2} & \mathbf{C_3} & \mathbf{0} & \mathbf{C_3} & \mathbf{C_2} & \mathbf{C_1} \\ \mathbf{C_1} & \mathbf{C_0} & \mathbf{C_1} & \mathbf{C_2} & \mathbf{C_3} & \mathbf{0} & \mathbf{C_3} & \mathbf{C_2} \\ \mathbf{C_2} & \mathbf{C_1} & \mathbf{C_0} & \mathbf{C_1} & \mathbf{C_2} & \mathbf{C_3} & \mathbf{0} & \mathbf{C_3} \\ \mathbf{C_3} & \mathbf{C_2} & \mathbf{C_1} & \mathbf{C_0} & \mathbf{C_1} & \mathbf{C_2} & \mathbf{C_3} & \mathbf{0} \\ \mathbf{0} & \mathbf{C_3} & \mathbf{C_2} & \mathbf{C_1} & \mathbf{C_0} & \mathbf{C_1} & \mathbf{C_2} & \mathbf{C_3} \\ \mathbf{C_3} & \mathbf{0} & \mathbf{C_3} & \mathbf{C_2} & \mathbf{C_1} & \mathbf{C_0} & \mathbf{C_1} & \mathbf{C_2} \\ \mathbf{C_2} & \mathbf{C_3} & \mathbf{0} & \mathbf{C_3} & \mathbf{C_2} & \mathbf{C_1} & \mathbf{C_0} & \mathbf{C_1} \\ \mathbf{C_1} & \mathbf{C_2} & \mathbf{C_3} & \mathbf{0} & \mathbf{C_3} & \mathbf{C_2} & \mathbf{C_1} & \mathbf{C_0} \end{bmatrix} , \tag{37}$$

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

$$\mathbf{C}_q = \begin{bmatrix} a_0^q & a_1^q & a_2^q & 0 & a_2^q & a_1^q \\ a_1^q & a_0^q & a_1^q & a_2^q & 0 & a_2^q \\ a_2^q & a_1^q & a_0^q & a_1^q & a_2^q & 0 \\ 0 & a_2^q & a_1^q & a_0^q & a_1^q & a_2^q \\ a_2^q & 0 & a_2^q & a_0^q & a_0^q & a_1^q \\ a_1^q & a_2^q & 0 & a_2^q & a_1^q & a_0^q \end{bmatrix} \tag{38}$$

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 two-dimensional unitary DFT (Davis, 1979, p. 185). It means that $\mathbf{C}$ (equation 30) satisfies

$$\mathbf{C} = \left(\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P}\right)^* \boldsymbol{\Lambda} \left(\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P}\right) , \tag{39}$$

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 $\mathbf{C}$. By substituting equation 39 in the auxiliary system (equation 30) and premultiplying both sides of the result by $(\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P})$, we obtain

$$\mathbf{\Lambda} \left(\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P} \right) \mathbf{v} = \left(\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P} \right) \mathbf{w} \, . \tag{40}$$

Now, by applying the *vec*-operator to both sides of equation 40 (see the details in Appendix A), we obtain:

$$\mathbf{F}_{2Q}^{*} \left[\mathbf{L} \circ \left(\mathbf{F}_{2Q} \, \mathbf{V} \, \mathbf{F}_{2P} \right) \right] \mathbf{F}_{2P}^{*} = \mathbf{W} \, , \tag{41}$$

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 41 contains the two-dimensional Inverse Discrete Fourier Transform (IDFT) of the term in brackets, which in turn represents the Hadamard product of matrix $\mathbf{L}$ (equation A-11) and the DFT of matrix $\mathbf{V}$ (equation A-9). Matrix $\mathbf{L}$ contains the eigenvalues of $\mathbf{\Lambda}$ (equation 39) and can be efficiently computed by using only the first column of the BCCB matrix $\mathbf{C}$ (equation 30) (see the details in Appendix B). Here, we evaluate equation 41 and compute matrix $\mathbf{L}$ by using the two-dimensional Fast Fourier Transform (2D FFT).

Therefore, 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) forming the BTTB matrix $\mathbf{A}$ (equation 17);

(2) Rearrange the first column of $\mathbf{A}$ equations

$\mathbf{c}_0^q$ (equation 36) of each block $\mathbf{C}_q$ to obtain the first column $\mathbf{c}_0$ of the BCCB matrix

C (equation 30);

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

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

(5) Rearrange **v** to obtain matrix **V**, use the 2D FFT to compute its DFT and evaluate the left side of equation A-12;

(6) Use the 2D FFT to compute the IDFT of the result obtained at item (5) and obtain the matrix **W** (equation 41);

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

Computational performance

The number of flops (floating-point operations) required to compute the

It is well known that FFT takes $5N\log_2(N)$ flops (Van Loan, 1973, p. 15). Computing the eigenvalues of the BCCB matrix ($4N \times 4N$) and applying 2D-FFT on the parameter row-oriented matrix **V** (equation **??**), takes $8N\log_2(4N)$ each. However, the sensitivity matrix does not change during the process thus, the eigenvalues of BCCB must be calculated only once, outside of the iteration. The multiplication of two complex numbers takes four real multiplications and two additions, which brings 6 times $4N$ flops to carry the Hadamard product. As it is necessary to compute the inverse FFT, another $8N\log_2(4N)$ must be taken in account. This lead us to a flops count in our method of

$$f_1 = 8N \log_2(4N) + N^{it}(27N + 16N \log_2(4N)). \tag{42}$$

Another major improvement of this methodology is the exoneration of calculating the full sensibility matrix $\mathbf{A}$ (equation 1). Each element needs 12 flops (equation 2), totalizing $12N^2$ flops for the full matrix. Calculating a single column of the BTTB matrix requires $12N$ flops. Thus, the full flops count of the method presented by Siqueira et al. (2017) is

$$f_s = 12N^2 + N^{it}(3N + 2N^2), \tag{43}$$

and it is decreased in our method to

$$f_f = 12N + 8N \log_2(4N) + N^{it}(27N + 16N \log(4N)). \tag{44}$$

PAREI AQUI

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

$$f_0 = N^{it}(3N + 2N^2) \ , \tag{45}$$

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.

Figure 1 shows the floating points to estimate the parameter vector using the fast equivalent layer by using the method of Siqueira et al. (2017) (equation 45) and our approach (equation 42) versus the number of observation points varyig from $N = 5000$ to $N = 1\,000\,000$ with 50 iterations. The number of operations is drastically decreased.

Table 1 shows the system RAM memory usage needed to store the full sensitivity matrix, the single column of the sensitivity matrix to form the BTTB matrix and the BCCB eigenvalues (8 times greater than that required by the BTTB first column). The quantities were computed for different numbers of data (N) with the same corresponding number of equivalent sources (N). Table 1 considers that each element of the matrix is a double-precision number, which requires 8 bytes of storage, except for the BCCB complex eigenvalues, which requires 16 bytes per element. Notice that 1 000 000 observation points requires nearly 7.6 Terabytes of RAM memory to store the whole sensibility matrix of the equivalent layer.

Using a PC with a Intel Core i7 4790@3.6GHz processor and 16 Gb of RAM memory, Figure 2 compares the running time of the Siqueira et al. (2017) method with the one of our work, considering a constant number of iterations equal to 50. Clearly, the major advantage of our approach is its computational efficiency that allows a rapid calculation of the gravity forward modeling with number of observations greater than 10 000. Because of the RAM available in this system, we could not perform this comparison with more observations. Therefore, the number of observation is limited to 22 500. Disregarding the RAM limitation, Figure 3 shows the running time of our method with 50 iterations and with the number of observations up to 25 millions. Our method requires 26:8 seconds to run one million of observations, whereas Siqueira et al. (2017) method took 48:3 seconds to run 22 500 observations

SYNTHETIC TESTS

In this section, we investigate the effectiveness of using the properties of BTTB and BCCB matrices (equation 30) to solve, at each iteration, the forward modeling (the matrix-vector product $\mathbf{A}\hat{\mathbf{p}}^k$) required in the fast equivalent-layer method proposed by Siqueira et al.

(2017). We simulated three sources whose horizontal projections are shown in Figure 4 as black lines. These sources are two vertical prisms with density contrasts of $0.35\,\mathrm{g/cm^3}$ (upper-left prism) and $0.4\,\mathrm{g/cm^3}$ (upper-right prism) and a sphere with radius of 1 000 m with density contrast of $-0.5\,\mathrm{g/cm^3}$. Figure 4 shows the vertical component of gravity field generated by these sources contaminated with additive pseudorandom Gaussian noise with zero mean and standard deviation of 0.015 mGal.

The advantage of using the structures of BTTB and BCCB matrices to compute forward modeling in the fast-equivalent layer method (Siqueira et al., 2017) is grounded on the use of regular grids of data and equivalent sources. Hence, we created 10 000 observation points regularly spaced in a grid of 100×100 at 100 m height. We also set a grid of equivalent point masses, each one directly beneath each observation points, located at 300 m deep. Figures 5a and 6a show the fitted gravity data obtained, respectively, by the fast equivalent layer method and by our modified form of this method that computes the forward modeling using equation 30. The corresponding residuals (Figures 5b and 6b), defined as the difference between the observed (Figure 4) and fitted gravity data (Figures 5a and 6a), show means close to zero and standard deviations of 0.0144 mGal. Therefore, Figures 5 and 6 show that Siqueira et al. (2017) method and our modified version of this method produced virtually the same results. This excellent agreement is confirmed in Figures 7 and 8 which shows that there are virtually no differences, respectively, in the fitted data presented in Figures 5b and 6b and in the estimated mass distributions within the equivalent layers (not shown) yielded by both Siqueira et al. (2017) method and our modification of this method. These results (Figures 7 and 8) show that computing the matrix-vector product ($\mathbf{A}\hat{\mathbf{p}}^k$), required in the forward modeling, by means of embedding the BTTB matrix into a BCCB matrix (equation 30) yields practically the same result as the one produced by computing this matrix-vector product with a full matrix $\mathbf{A}$ as used in Siqueira et al. (2017).

We perform two forms of processing the gravity data (Figure 4) through the equivalent layer technique: the upward (Figure 9) and the downward (Figure 10) continuations. The upward height is 300 m and the downward is at 50 m. Either in the upward continuation (Figure 9) or in the downward continuation (Figure 10), the continued gravity data using the fast equivalent layer proposed by Siqueira et al. (2017) (Figures 9a and 10a) are in close agreement with those produced by our modification of Siqueira et al. (2017) method (Figures 9b and 10b). The residuals (Figures 9c and 10c) quantify this agreement since their means and standard deviations are close to zero in both continued gravity data using both methods. All the continued gravity data shown here (Figures 9 and 10) agree with the true ones (not shown). The most striking feature of these upward or the downward continuations concerns the total computation time. The computation time spent by our method is approximately 1 500 times faster than Siqueira et al. (2017) method.

REAL DATA TEST

Test with real data are conducted with the gravity data from Carajás, north of Brazil, were provided by the Geological Survey of Brazil (CPRM). The real aerogravimetric data were collected in 113 flight lines along northsouth direction with flight line spacing of 3 km and tie lines along eastwest direction at 12 km.

This airborne gravity survey was divided in two different areas, collected in different times, having samples spacing of 7.65 m and 15.21 m, totalizing 4,353,428 observation points. The height of the flight was fixed at 900 m. The gravity data (Figure 11) were gridded into a regularly spaced dataset of 250 000 observation points (500×500) with a grid spacing of 716.9311 km north-south and 781.7387 km east-west.

To apply our modification of the fast equivalent-layer method (Siqueira et al., 2017) that computes the forward modeling using the properties of BTTB and BCCB matrices (equation 30), we set an equivalent layer at 300 m deep. Figure 12a shows the fitted gravity data after 50 iterations by applying our method. The residuals (Figure 12b), defined as the difference between the observed (Figure 11) and the predicted (Figure 12a) data, show an acceptable data fitting because they have a mean close to zero (0.0003 mGal) and a small standard deviation of 0.105 mGal which corresponds to approximately 0.1 % of the amplitude of the gravity data.

These small residuals indicate that our method yielded an estimated mass distribution (not shown) that can be used in the data processing. We perform upward-continuation of the real gravity data (Figure 11) at a constant height of 5 000 m over the real data. The upward-continued gravity data (Figure 13) seem a reasonable processing because of the attenuation of the short wavelenghts. By using our approach, the processing of the 250 000 observations was extremely fast and took 0.216 seconds.

CONCLUSIONS

By exploring the BTTB structure of the sensitivity matrix in the gravity data processing, we have proposed a new efficient approach for calculating the gravity-data forward modeling required in the iterative fast equivalent-layer technique grounded on excess mass constraint that does not demand the solution of linear systems. Its efficiency requires the use of regular grids of observations and equivalent sources (point masses). Our algorithm greatly reduces the number of flops necessary to estimate a 2D mass distribution within the equivalent layer that fits the observed gravity data. For example, when processing one million observations the number of flops is reduced in 104 times. Traditionally, such amount of data impractically requires 7.6 Terabytes of RAM memory to handle the full sensitivity matrix. Rather, in our method, this matrix takes 61.035 Megabytes of RAM memory only.

Our method takes advantage of the symmetric BTTB system that arises when processing a harmonic function and considering that either the observations or the sources of the interpretative model (point of masses over the equivalent layer) are distributed on regular grids. Symmetric BTTB matrices can be stored by using only a single column and can be embedded into a symmetric BCCB matrix, which in turn also only needs a single column. This means that only a single column of the sensitivity matrix needs to be calculated. Once the jth column of the sensitivity matrix represents the influence of the jth source (jth point mass) has on the predicted data, our method only requires a single point mass that set up the equivalent layer to compute the gravity-data forward modeling.

Using the 2D FFT, it is possible to calculate the eigenvalues of BCCB matrices which can be used to compute a matrix-vector product (gravity-data forward modeling) in a very low computational cost. We have successfully applied the proposed method to upward (or downward) synthetic gravity data. Testing on field data from the Carajás Province, north of Brazil, confirms the potential of our approach in upward-continuing gravity data with 250 000 observations in about 0.2 seconds. Our method allows, in future research, applying the equivalent layer-technique for processing and interpreting massive data set such as collected in continental and global scales studies.

Figures

Figure 1

Figure 2

Figure 3

Figure 4

Figure 5

Figure 6

Figure 7

Figure 8

Figure 9

Figure 10

Figure 11

Figure 12

Figure 13

Tables

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

Table 1: Comparison between the system RAM memory usage needed to store the full matrix, the BTTB single column of the sensitivity matrix and the BCCB eigenvalues (eight times greater than the BTTB singel column). The quantities were computed for different numbers of data (N) with the same corresponding number of equivalent sources (N). This table considers that each element of the matrix is a double-precision number, which requires 8 bytes of storage, except for the BCCB complex eigenvalues, which requires 16 bytes per element.

APPENDIX A

COMPUTATIONS WITH THE 2D DFT

In the present Appendix, we deduce equation 41 by using the row-ordered *vec*-operator (here designated simply as *vec*-operator). This equation can be efficiently computed by using the two-dimensional 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. It is important to stress that this operator is different from that defined by Neudecker (1969), which is most commonly found in the literature and transforms a matrix into a column vector by stacking its columns.

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

$$\mathbf{M} = \begin{bmatrix} \mathbf{m}_1^\top \\ \vdots \\ \mathbf{m}_N^\top \end{bmatrix} , \qquad \text{(A-1)}$$

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:

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

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

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

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

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 A-3), it can be shown that

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

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):

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

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 A-5), Jain (1989, p. 31) showed that:

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

where "$\otimes$" denotes the Kronecker product (Neudecker, 1969). It is important to stress the difference between equation A-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 A-6 uses the *vec*-operator defined by equation A-2, which transforms a matrix into a column vector by stacking its rows.

Now, let us deduce equation 41 by using the above-defined properties (equation A-4 and A-6). We start calling attention to the right side of equation 40. Consider that vector $\mathbf{w}$ (equation 40) is obtained by applying the *vec*-operator (equation A-2) to a matrix $\mathbf{W}$, whose two-dimensional DFT $\tilde{\mathbf{W}}$ is represented by the following separable transformation (Jain, 1989, p. 146):

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

where $\mathbf{F}_{2Q}$ and $\mathbf{F}_{2P}$ are the $2Q \times 2Q$ and $2P \times 2P$ unitary DFT matrices. Using equation A-6 and the symmetry of unitary DFT matrices, we rewrite the right side of equation 40 as follows:

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

Similarly, consider that $\mathbf{v}$ (equation 40) is obtained by applying the *vec*-operator (equation A-2) to a matrix $\mathbf{V}$, whose two-dimensional DFT (equation A-7) is represented by $\tilde{\mathbf{V}}$, and use equation A-6 to rewrite the left side of equation 40 as follows:

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

Note that both sides of equation A-9 are defined as the product of the diagonal matrix $\boldsymbol{\Lambda}$ (equation 39) and a vector. In this case, the matrix-vector product can be conveniently replaced by

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

where $\boldsymbol{\lambda}$ is a $4QP \times 1$ vector containing the diagonal of $\boldsymbol{\Lambda}$ (equation 39). Then, consider that $\boldsymbol{\lambda}$ is obtained by applying the *vec*-operator (equation A-2) to a $2Q \times 2P$ matrix $\mathbf{L}$ and use equations A-4 and A-6 to rewrite equation A-10 as follows:

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

Equations A-7, A-8 and A-11 show that equation 40 is obtained by applying the *vec*-operator to

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

Finally, we premultiply both sides of equation A-12 by $\mathbf{F}_{2Q}^{*}$ and then postmultiply both sides of the result by $\mathbf{F}_{2P}^{*}$ to deduce equation 41.

APPENDIX B

THE EIGENVALUES OF C

In the present Appendix, we show how to efficiently compute matrix $\mathbf{L}$ (equations A-11, A-12 and 41) by using only the first column of the BCCB matrix $\mathbf{C}$ (equation 30).

We need first premultiply both sides of equation 39 by $(\mathbf{F}_{2Q} \otimes \mathbf{F}_{2P})$ to obtain

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

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

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

where $\mathbf{c}_0$ is a $4QP \times 1$ vector representing the first column of $\mathbf{C}$ (equation 30) and $\boldsymbol{\lambda}$ (equation A-10) is the $4QP \times 1$ vector that contains the diagonal of matrix $\mathbf{\Lambda}$ (equation 39) and is obtained by applying the *vec*-operator (equation A-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 A-5) and the symmetry of unitary DFT matrices, equation B-2 can be rewritten as follows

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

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 two-dimensional 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.

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., T. F. Chan, and C.-K. Wong, 1999, Cosine transform based preconditioners for total variation deblurring: IEEE transactions on Image Processing, **8**, 1472–1478.

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.

Grenander, U., and G. Szegö, 1958, Toeplitz forms and their applications: California Monographs in Mathematical Sciences. University of California Press, Berkeley, CA.

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.

Jin, X.-Q., 2003, Developments and applications of block toeplitz iterative solvers: Springer Science & Business Media, **2**.

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.

Lin, F., X. Jin, and S. Lei, 2003, Strang-type preconditioners for solving linear systems from delay differential equations: BIT Numerical Mathematics, **43**, 139–152.

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.

Olkin, J. A., 1986, Linear and nonlinear deconvolution problems: PhD thesis, Rice University.

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.

Strang, G., and K. Aarikka, 1986, Introduction to applied mathematics: Wellesley-Cambridge Press Wellesley, MA, **16**.

Van Loan, C., 1973, Computational frameworks for the fast fourier transform: SIAM.

Wray, J., and G. G. Green, 1994, Calculation of the volterra kernels of non-linear dynamic systems using an artificial neural network: Biological Cybernetics, **71**, 187–195.

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 FIGURES

dard deviation of 0.0144 mGal.

7 Difference between the fitted gravity data produced by Siqueira et al. (2017) method (Figure 5a) and by our modified form of this method (Figure 6a) that computes the forward modeling using the properties of BTTB and BCCB matrices (equation 30).

8 Difference between the estimated mass distribution within the equivalent layer produced by Siqueira et al. (2017) method (Figure 5) and by our modified form of this method (Figure 6) that computes the forward modeling using the properties of BTTB and BCCB matrices (equation 30).

9 The upward-continued gravity data using: (a) the fast equivalent layer proposed by Siqueira et al. (2017) and (b) our modified form of Siqueira et al. (2017) method by using the properties of BTTB and BCCB matrices (equation 30) to calculate the forward modeling. (c) Residuals, defined as the difference between panels a and b with their mean of $-5.938e^{-18}$ and standard deviation of $8.701e^{-18}$. The total computation times in the Siqueira et al. (2017) method and in our approach are 7.62026 and 0.00834 seconds, respectively.

10 The downward-continued gravity data using: (a) the fast equivalent layer proposed by Siqueira et al. (2017) and (b) our modified form of Siqueira et al. (2017) method by using the properties of BTTB and BCCB matrices (equation 30) to calculate the forward modeling. (c) Residuals, defined as the difference between panels a and b with their mean of $5.914e^{-18}$ and standard deviation of $9.014e^{-18}$. The total computation times in the Siqueira et al. (2017) method and in our approach are 7.59654 and 0.00547 seconds, respectively.

11 Carajás Province, Brazil. Gravity data on a regular grid of 500×500 points, totaling $250,000$ observations. The inset shows the study area (blue rectangle) which covers the southeast part of the state of Pará, north of Brazil.

12 Carajás Province, Brazil. (a) Predicted gravity data produced by our modification of the fast equivalent-layer method (Siqueira et al., 2017) that computes the forward modeling using the properties of BTTB and BCCB matrices (equation 30). (b) Gravity residuals, defined as the difference between the observed data in Figure 11 and the predicted data in panel a, with their mean of 0.000292 mGal and standard deviation of 0.105 mGal.

13 Carajás Province, Brazil. The upward-continued gravity data using our modification of the fast equivalent layer method (Siqueira et al., 2017) that computes the forward modeling using the properties of BTTB and BCCB matrices (equation 30). The total computation time is 0.216 seconds for processing of the $250,000$ observations.

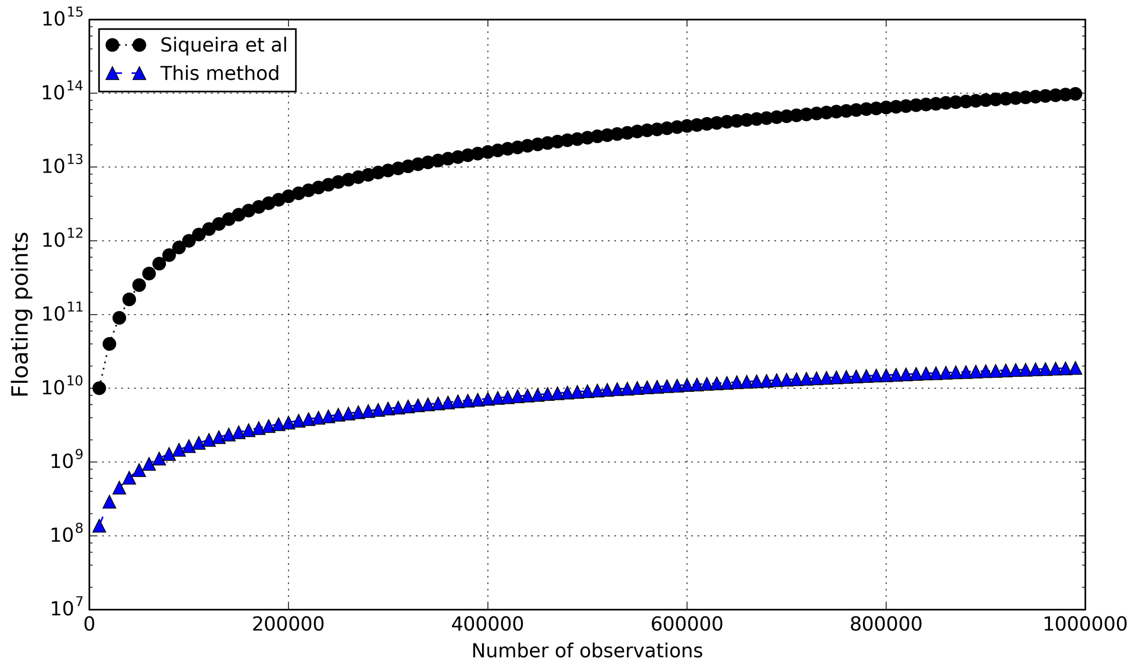

Figure 1: floating points to estimate the parameter vector using the fast equivalent layer using Siqueira et al. (2017) method (equation 45) and our approach (equation 42) versus the numbers of observation points varyig from $N = 5\,000$ to $N = 1\,000\,000$ with 50 iterations. The number of operations is drastically decreased.

– **GEO-XXXX**

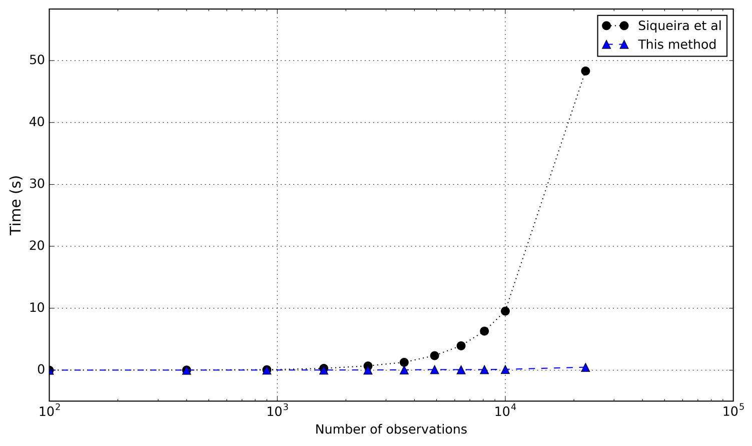

Figure 2: time necessary to run 50 iterations of the Siqueira et al. (2017) method and the one presented in this work. With the limitation of 16 Gb of RAM memory in our system, we could test only up to 22 500 obervation points.

– **GEO-XXXX**

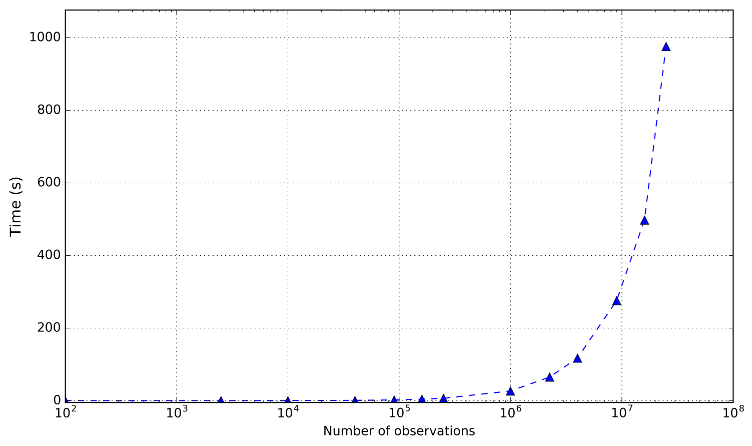

Figure 3: time necessary to run the equivalent layer technique with 50 iterations using our approach, where the RAM is not a limitation factor. We could run up to 25 million observation points. In comparison, 1 million observation points took 26.8 seconds to run, where the maximun 22 500 observation points in Figure 2, with Siqueira et al. (2017) method, took 48:3 seconds.

– **GEO-XXXX**

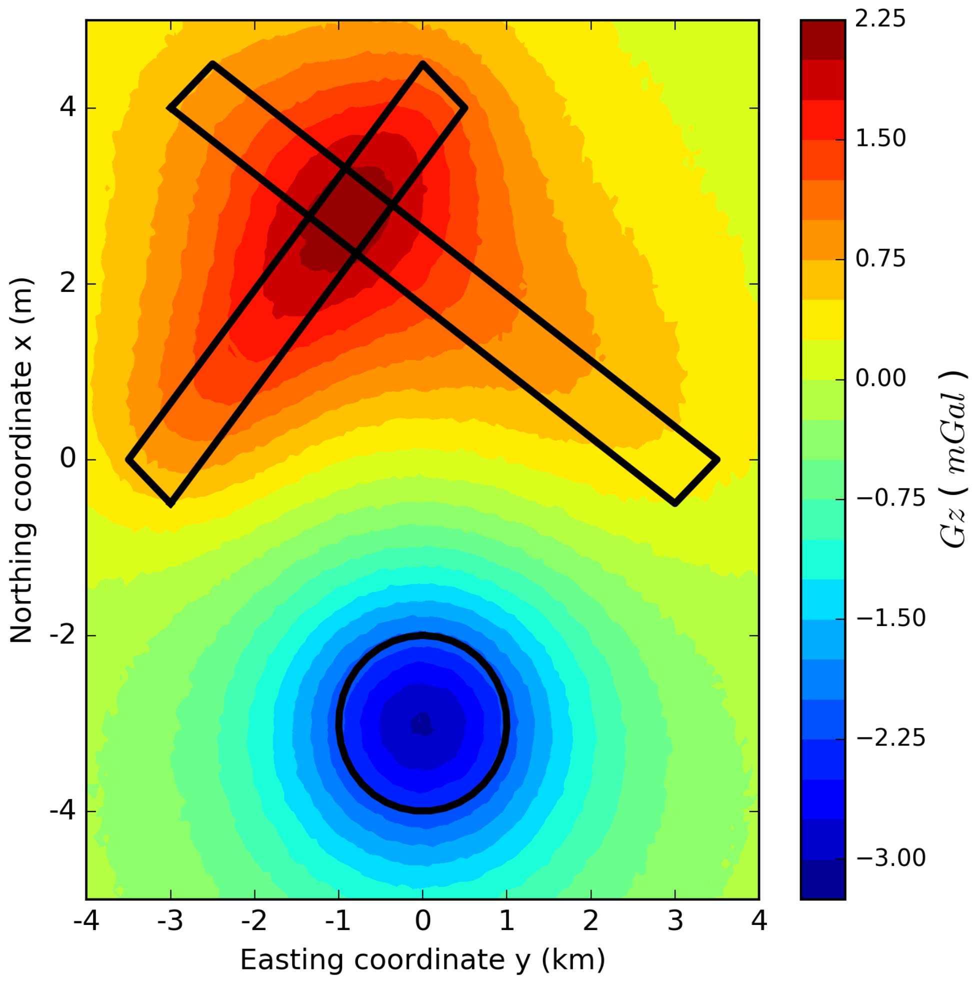

Figure 4: Noise-corrupted gravity data (in color map) produced by three simulated sources whose horizontal projections are shown in black lines. The simulated sources are: two polygonal prisms, with density contrast of $0.35\,\mathrm{g/cm^3}$ (upper-left body) and $0.4\,\mathrm{g/cm^3}$ (upper-right body), and a sphere with radius of 1 000 m with density contrast of $-0.5\,\mathrm{g/cm^3}$.

– **GEO-XXXX**

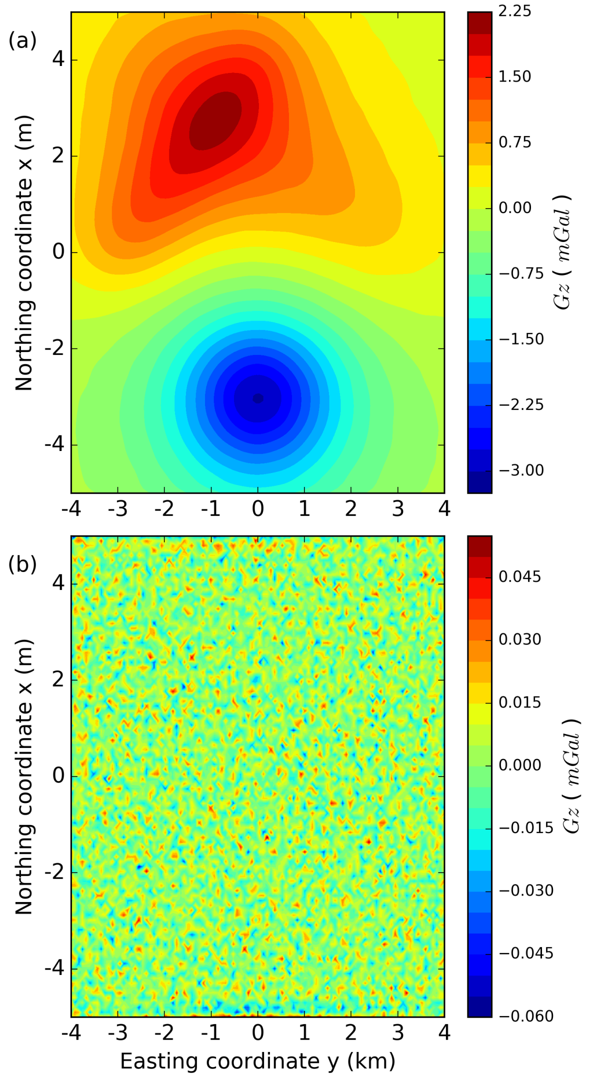

Figure 5: (a) Fitted gravity data produced by the fast equivalent-layer technique proposed by Siqueira et al. (2017). (b) Gravity residuals, defined as the difference between the observed data in Figure 4 and the predicted data in panel a, with their mean of $8.264e^{-7}$ and standard deviation of 0.0144 mGal.

– **GEO-XXXX**

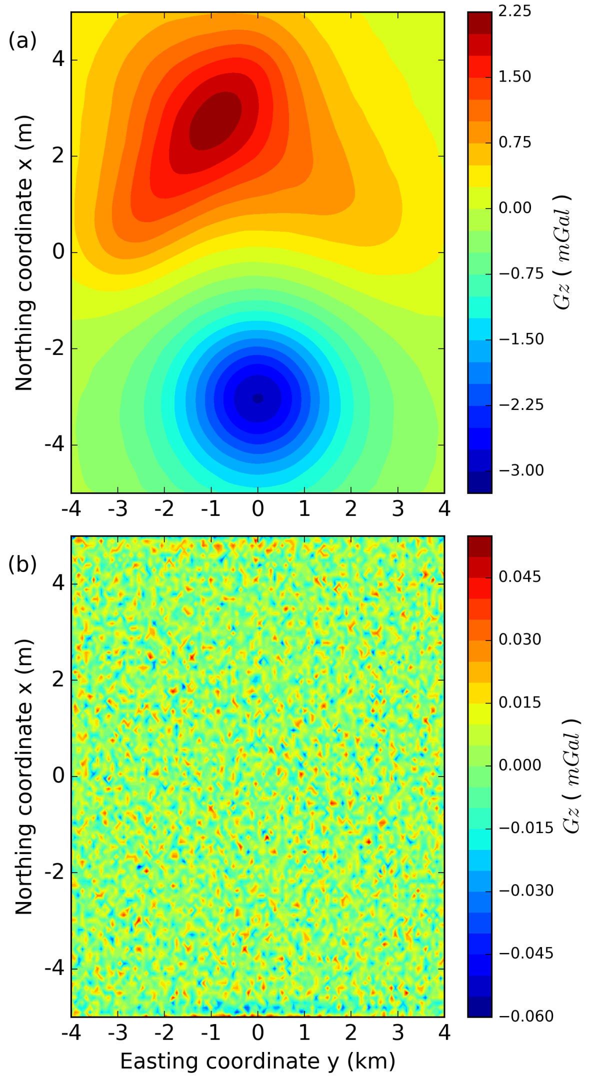

Figure 6: (a) Fitted gravity data produced by our modification of the fast equivalent layer (Siqueira et al., 2017). (b) Gravity residuals, defined as the difference between the observed data in Figure 4 and the predicted data in panel a, with their mean of $8.264e^{-7}$ and standard deviation of 0.0144 mGal.

– **GEO-XXXX**

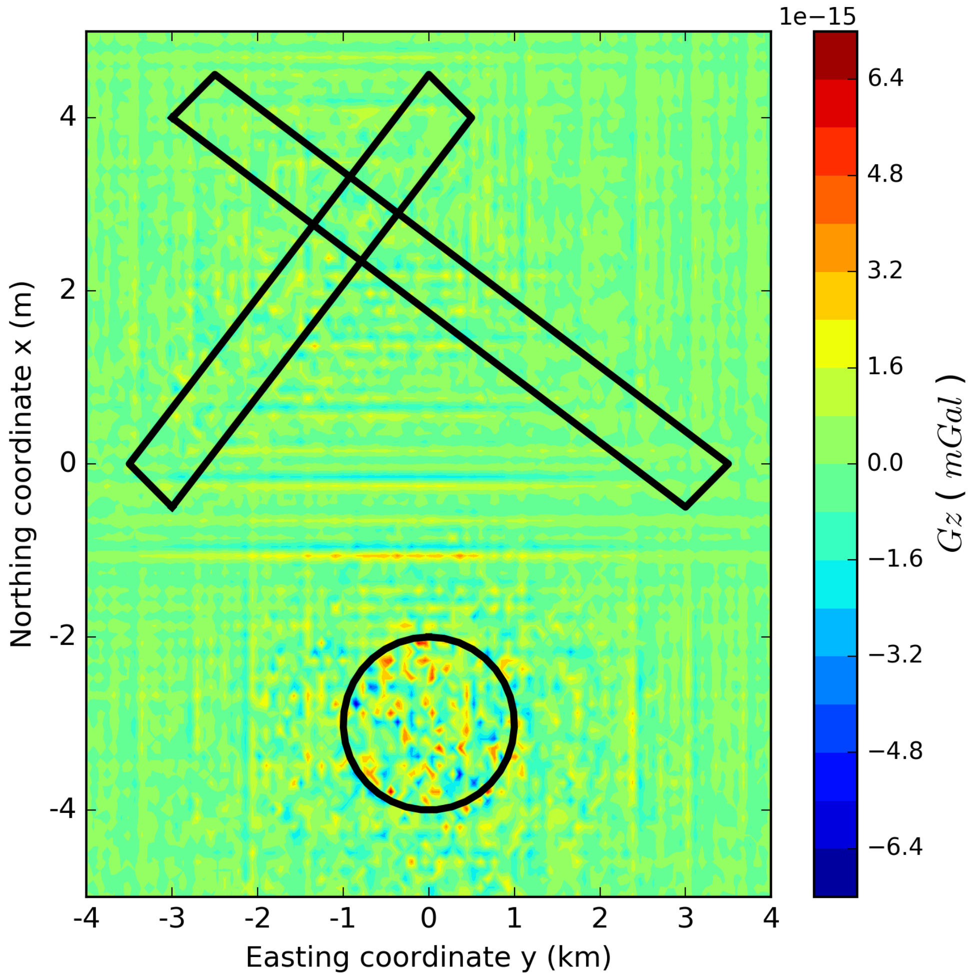

Figure 7: Difference between the fitted gravity data produced by Siqueira et al. (2017) method (Figure 5a) and by our modified form of this method (Figure 6a) that computes the forward modeling using the properties of BTTB and BCCB matrices (equation 30).

– **GEO-XXXX**

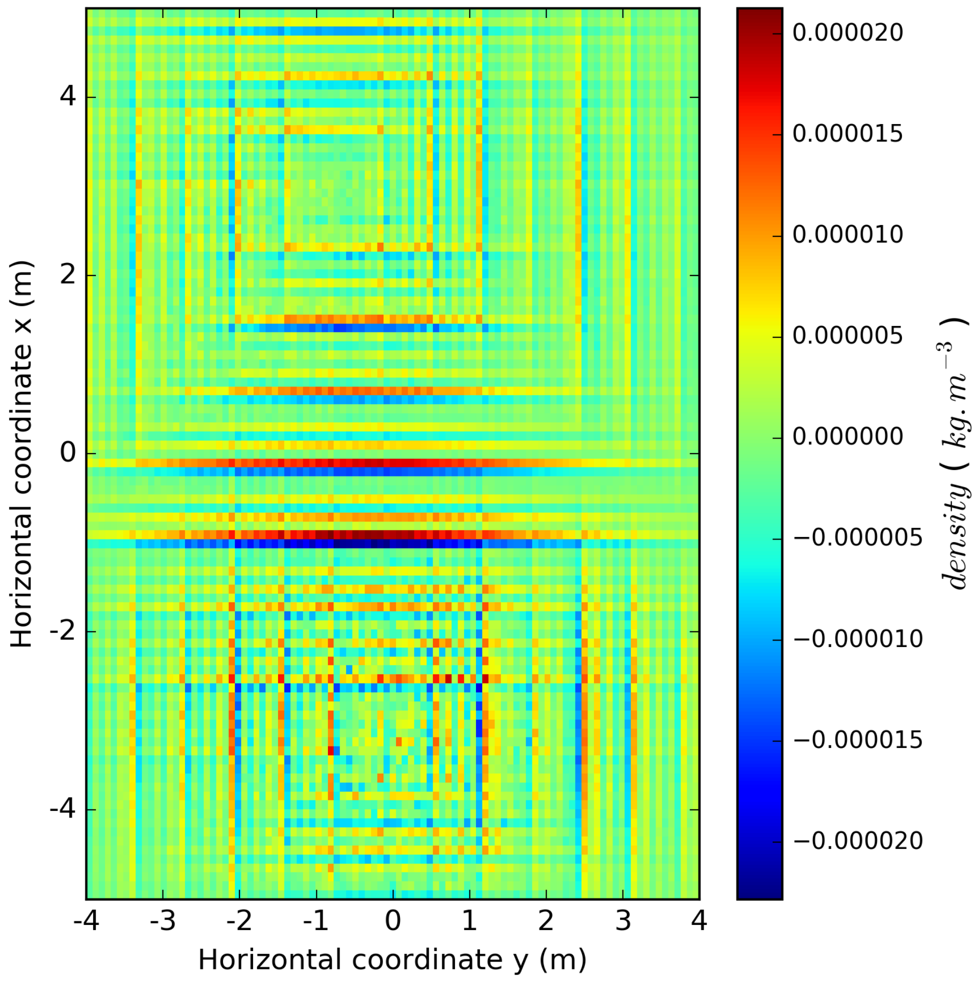

Figure 8: Difference between the estimated mass distribution within the equivalent layer produced by Siqueira et al. (2017) method (Figure 5) and by our modified form of this method (Figure 6) that computes the forward modeling using the properties of BTTB and BCCB matrices (equation 30).

– **GEO-XXXX**

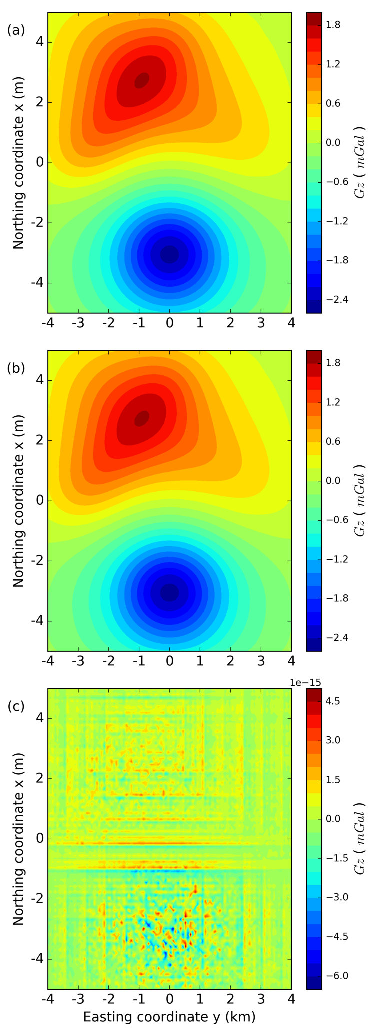

Figure 9: The upward-continued gravity data using: (a) the fast equivalent layer proposed by Siqueira et al. (2017) and (b) our modified form of Siqueira et al. (2017) method by using the properties of BTTB and BCCB matrices (equation 30) to calculate the forward modeling. (c) Residuals, defined as the difference between panels a and b with their mean of $-5.938e^{-18}$ and standard deviation of $8.701e^{-18}$. The total computation times in the Siqueira et al. (2017) method and in our approach are 7.62026 and 0.00834 seconds, respectively.

– **GEO-XXXX**

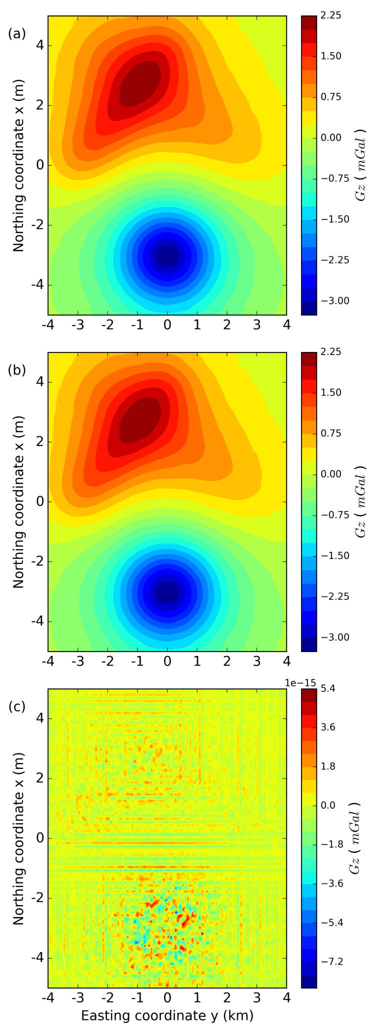

Figure 10: The downward-continued gravity data using: (a) the fast equivalent layer proposed by Siqueira et al. (2017) and (b) our modified form of Siqueira et al. (2017) method by using the properties of BTTB and BCCB matrices (equation 30) to calculate the forward modeling. (c) Residuals, defined as the difference between panels a and b with their mean of $5.914e^{-18}$ and standard deviation of $9.014e^{-18}$. The total computation times in the Siqueira et al. (2017) method and in our approach are 7.59654 and 0.00547 seconds, respectively.

– **GEO-XXXX**

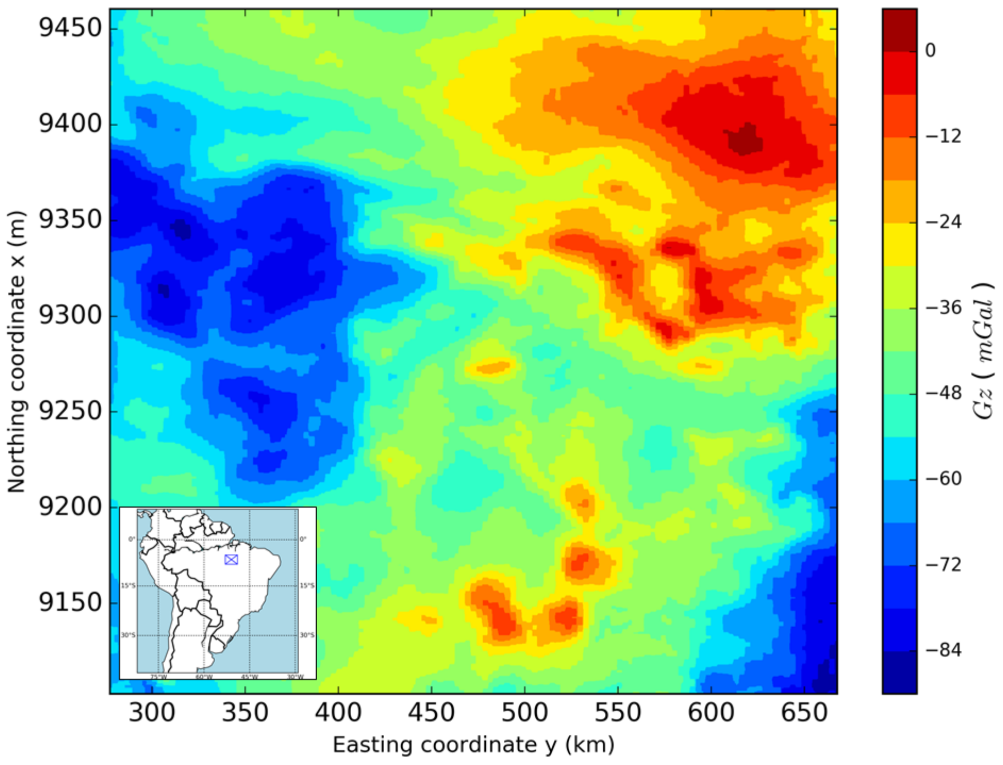

Figure 11: Carajás Province, Brazil. Gravity data on a regular grid of 500×500 points, totaling $250,000$ observations. The inset shows the study area (blue rectangle) which covers the southeast part of the state of Pará, north of Brazil.

– **GEO-XXXX**

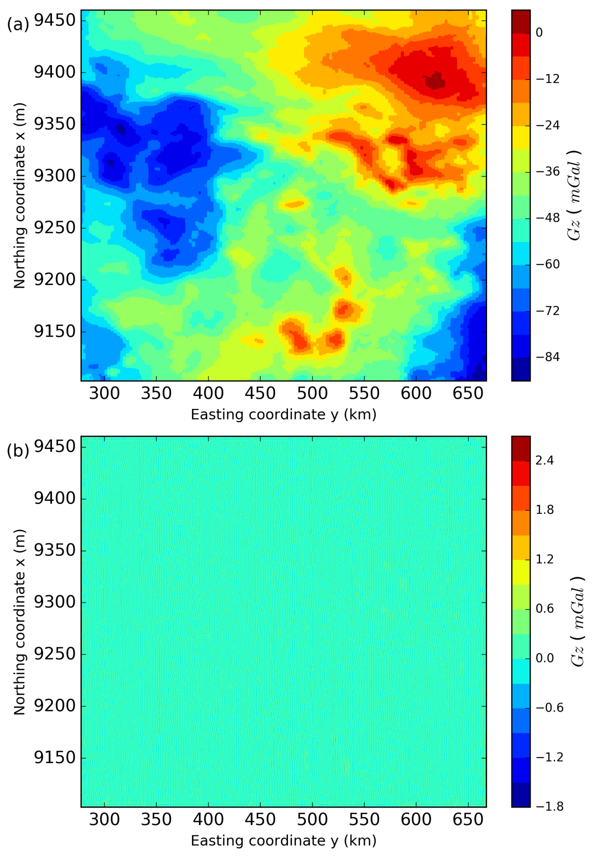

Figure 12: Carajás Province, Brazil. (a) Predicted gravity data produced by our modification of the fast equivalent-layer method (Siqueira et al., 2017) that computes the forward modeling using the properties of BTTB and BCCB matrices (equation 30). (b) Gravity residuals, defined as the difference between the observed data in Figure 11 and the predicted data in panel a, with their mean of 0.000292 mGal and standard deviation of 0.105 mGal.

– **GEO-XXXX**

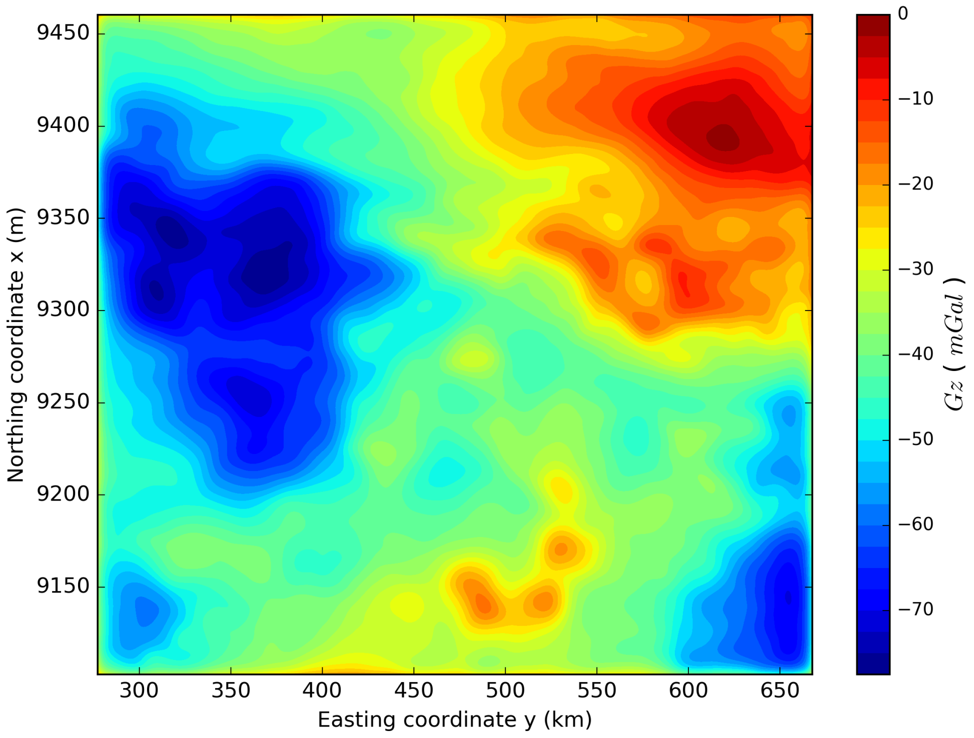

Figure 13: Carajás Province, Brazil. The upward-continued gravity data using our modification of the fast equivalent layer method (Siqueira et al., 2017) that computes the forward modeling using the properties of BTTB and BCCB matrices (equation 30). The total computation time is 0.216 seconds for processing of the $250,000$ observations.

– **GEO-XXXX**