Tesseroids: forward modeling gravitational fields in spherical coordinates

Leonardo Uieda[1,2,*], Valéria C. F. Barbosa[2], and Carla Braitenberg[3]

[1] Universidade do Estado do Rio de Janeiro, Rio de Janeiro, Brazil.
[2] Observatório Nacional, Rio de Janeiro, Brazil.
[3] Department of Mathematics and Geosciences, University of Trieste, Trieste, Italy
[*] e-mail: leouieda@gmail.com

Accepted 2016 July 11. Received in original form 2015 March 30.

Peer-reviewed code related to this article can be found at software.seg.org/2016/0004 or github.com/pinga-lab/paper-tesseroids.

Abstract

We present the open-source software *Tesseroids*, a set of command-line programs to perform the forward modeling of gravitational fields in spherical coordinates. The software is implemented in the C programming language and uses tesseroids (spherical prisms) for the discretization of the subsurface mass distribution. The gravitational fields of tesseroids are calculated numerically using the Gauss-Legendre Quadrature (GLQ). We have improved upon an adaptive discretization algorithm to guarantee the accuracy of the GLQ integration. Our implementation of adaptive discretization uses a "stack" based algorithm instead of recursion to achieve more control over execution errors and corner cases. The algorithm is controlled by a scalar value called the distance-size ratio (D) that determines the accuracy of the integration as well as the computation time. We determined optimal values of D for the gravitational potential, gravitational acceleration, and gravity gradient tensor by comparing the computed tesseroids effects with those of a homogeneous spherical shell. The values required for a maximum relative error of 0.1% of the shell effects are $D = 1$ for the gravitational potential, $D = 1.5$ for the gravitational acceleration, and $D = 8$ for the gravity gradients. Contrary to previous assumptions, our results show that the potential and its first and second derivatives require different values of D to achieve the same accuracy. These values were incorporated as defaults in the software.

1 Introduction

Satellite missions dedicated to measuring the Earth's gravity field (like CHAMP, GRACE, and GOCE) have provided geophysicists with almost uniform and global data coverage. These new data have enabled interpretations on regional and global scales (e.g. Reguzzoni et al., 2013; Braitenberg, 2015). Modeling at such scales requires taking into account the curvature of the Earth and calculating gravity gradients as well as the traditional gravitational acceleration. A common approach to achieve this is to discretize the Earth into tesseroids (Figure 1) instead of rectangular prisms. An analytical solution exists when the computation point is along the polar axis and the tesseroid is extended into a spherical cap (LaFehr, 1991; Mikuška et al., 2006; Grombein et al., 2013). For more general cases, the integral formula for the gravitational effects of a tesseroid must be solved numerically. Approaches to this numerical integration include Taylor series expansion (Heck and Seitz, 2007; Grombein et al., 2013) and the Gauss-Legendre Quadrature (Asgharzadeh et al., 2007). Taylor series expansion produces accurate results at low latitudes but presents a decrease in accuracy towards the polar regions. This is attributed to tesseroids degenerating into an approximately triangular shape at the poles. The Gauss-Legendre Quadrature (GLQ) integration consists in approximating the volume integral by a weighted sum of the effect of point masses. An advantage of the GLQ approach is that it can be controlled by the number of point masses used. The larger the number of point masses, the better the accuracy of GLQ integration. A disadvantage is the increased computation time as the number of point masses increases. Thus, there is a trade-off between accuracy and computation time. This is a common theme in numerical methods. Wild-Pfeiffer (2008) investigated the use of different mass elements, including tesseroids, to compute the gravitational effects of topographic masses. The author concludes that using tesseroids with GLQ integration gives the best results for near-zone computations. However, the question of how to determine the optimal parameters for GLQ integration remained open.

Previous work by Ku (1977) investigated the use of the GLQ in gravity forward modeling. Ku (1977) numerically integrated the vertical component of the gravitational acceleration of right rectangular prisms. The author suggested that the accuracy of the GLQ integration depends on the ratio between distance to the computation point and the distance between adjacent point masses. Based on this, Ku (1977) proposed an empirical criterion that the distance between point masses should not be greater than the distance to the computation point. Asgharzadeh et al. (2007) used this criterion for the GLQ integration of the gravity gradient tensor of tesseroids. To our knowledge, an analysis of how well this ad hoc criteria of Ku (1977) works for gravity gradient components or for tesseroids has never been done before. There has also been no attempt to quantify the error committed in the GLQ integration when applying the criteria of Ku (1977).

Li et al. (2011) devised an algorithm to automatically enforce the criteria of Ku (1977). Their algorithm divides the tesseroid into smaller ones instead of increasing the number of point masses per tesseroid. A tesseroid is divided if the minimum distance to the computation point is smaller than the largest dimension of the tesseroid. This division is repeated recursively until all tesseroids obey the criterion. Then, GLQ integration is performed for each of the smaller tesseroids using the specified number of point masses. The advantage of this adaptive discretization over increasing the number of points masses is that the total distribution of point masses will be greater only close to the computation point. This makes the adaptive discretization more computationally efficient.

Grombein et al. (2013) developed optimized formula for the gravitational fields of tesseroids using Cartesian integral kernels. These formulas are faster to compute and do not have singularities at the poles like their spherical counterparts. The Cartesian formulae are numerically integrated using a Taylor series expansion as per Heck and Seitz (2007). Grombein et al. (2013) use a near-zone separation to mitigate the increased error at high latitudes. In the so called "near-zone" of the computation point they use a finer discretization composed by smaller tesseroids. This is accomplished by dividing the tesseroids along their horizontal dimensions. However, the determination of an optimal size of the near-zone remains an open question (Grombein et al., 2013).

We have implemented a modified version of the adaptive discretion of Li et al. (2011) into the open-source software package *Tesseroids*. The software uses the Cartesian formula of Grombein et al. (2013) for improved performance and robustness. Previous versions of the software have been used by, e.g., Álvarez et al. (2012); Bouman et al. (2013a,b); Mariani et al. (2013); Braitenberg et al. (2013, 2011); Fullea et al. (2014).

This article describes the software design and the implementation of our modified adaptive discretization algorithm. We also present a numerical investigation of the error committed in the computations. These results allow us to calibrate the adaptive discretization algorithm separately for the gravitational potential, gravitational acceleration, as well as the gravity gradient tensor components.

2 Theory

A tesseroid is a mass element defined in geocentric spherical coordinates (Figure 1). It is bounded by two meridians, two parallels, and two concentric circles. The gravitational fields of a tesseroid at a point $P = (r, \phi, \lambda)$ are determined with respect to the local North-oriented coordinate system at P (x, y, z in Figure 1). Grombein et al. (2013) formulated Cartesian kernels for the volume integrals that define the

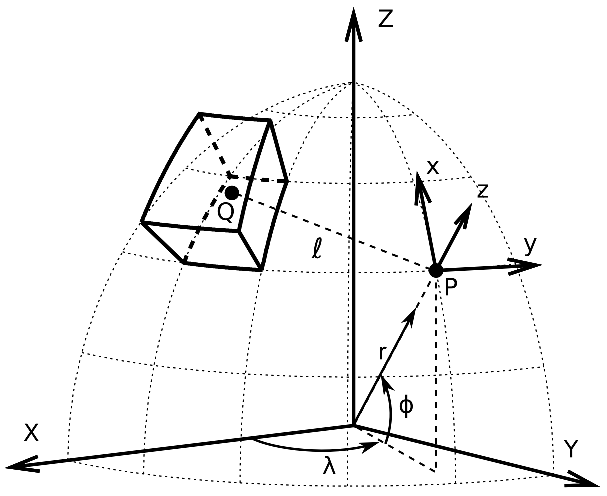

Figure 1: View of a tesseroid, the integration point Q inside the tesseroid, a geocentric coordinate system (X, Y, Z), the computation P and it's local coordinate system (x, y, z). r, ϕ, λ are the radius, latitude, and longitude, respectively, of point P, and ℓ is the Cartesian distance between P and Q. After Uieda (2015).

tesseroid gravitational potential, gravitational acceleration, and Marussi tensor, respectively,

$$V(r,\phi,\lambda) = G\rho \int_{\lambda_1}^{\lambda_2}\int_{\phi_1}^{\phi_2}\int_{r_1}^{r_2} \frac{1}{\ell}\kappa \; dr' d\phi' d\lambda', \tag{1}$$

$$g_\alpha(r,\phi,\lambda) = G\rho \int_{\lambda_1}^{\lambda_2}\int_{\phi_1}^{\phi_2}\int_{r_1}^{r_2} \frac{\Delta_\alpha}{\ell^3}\kappa \; dr' d\phi' d\lambda', \tag{2}$$

and

$$g_{\alpha\beta}(r,\phi,\lambda) = G\rho \int_{\lambda_1}^{\lambda_2}\int_{\phi_1}^{\phi_2}\int_{r_1}^{r_2} I_{\alpha\beta} \; \kappa \; dr' d\phi' d\lambda', \tag{3}$$

$$I_{\alpha\beta} = \left(\frac{3\Delta_\alpha\Delta_\beta}{\ell^5} - \frac{\delta_{\alpha\beta}}{\ell^3} \right), \tag{4}$$

where $\alpha, \beta \in \{x, y, z\}$, ρ is the density, $G = 6.674 \times 10^{-11}\ m^3 kg^{-1} s^{-1}$ is the gravitational constant, $\delta_{\alpha\beta}$ is Kronecker's delta ($\delta_{\alpha\beta} = 1$ if $\alpha = \beta$ and $\delta_{\alpha\beta} = 0$ if $\alpha \neq \beta$), and

$$\Delta_x = r'(\cos\phi \sin\phi' - \sin\phi\cos\phi'\cos(\lambda' - \lambda)), \tag{5}$$

$$\Delta_y = r'\cos\phi'\sin(\lambda' - \lambda), \tag{6}$$

$$\Delta_z = r'\cos\psi - r, \tag{7}$$

$$\kappa = {r'}^2 \cos\phi', \tag{8}$$

$$\ell = \sqrt{{r'}^2 + r^2 - 2r'r\cos\psi}, \tag{9}$$

$$\cos\psi = \sin\phi\sin\phi' + \cos\phi\cos\phi'\cos(\lambda' - \lambda). \tag{10}$$

We will follow Asgharzadeh et al. (2007) and perform the numerical integration using the Gauss-Legendre Quadrature (GLQ). The GLQ consists in approximating the integral by a weighted sum of the integration kernel (Hildebrand, 1987),

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\sum_{i=1}^{N} W_i f(x_i), \tag{11}$$

in which N is the order of the quadrature, i.e. the number of points used in the GLQ. The points x_i are called the quadrature nodes. They are the roots of the N^{th} order Legendre polynomial $P_N(x)$. For a second order polynomial ($P_2(x)$), the roots are $x = \pm 0.577350269$. Roots for larger order polynomials can be determined by a root finder algorithm. Roots of Legendre polynomials will be within the range $[-1, 1]$. Before being used for GLQ integration, the roots must be scaled to the integration limits $[a, b]$ using

$$x_i^{scaled} = \frac{b-a}{2}x_i + \frac{b+a}{2}. \tag{12}$$

The weights of the GLQ are given by (Hildebrand, 1987),

$$W_i = \frac{2}{(1 - x_i^2)(P'_N(x_i))^2}. \tag{13}$$

The values of $P_N(x)$ and its first derivative $P'_N(x)$ can be calculated with recursive relations.

The Gauss-Legendre Quadrature for three-dimensional volume integrals, like equations 1-3, becomes (Asgharzadeh et al., 2007)

$$\iiint_\Omega f(r', \lambda', \phi')d\Omega \approx A \sum_{i=1}^{N^r}\sum_{j=1}^{N^\phi}\sum_{k=1}^{N^\lambda} W_i^r W_j^\phi W_k^\lambda f(r_i, \phi_j, \lambda_k), \tag{14}$$

where

$$A = \frac{(\lambda_2 - \lambda_1)(\phi_2 - \phi_1)(r_2 - r_1)}{8}. \tag{15}$$

Comparing equation 14 with equations 1-3, we see that $f(r_i, \phi_j, \lambda_k)$ is the effect of a point mass located on the quadrature nodes. Thus, it can be said that the GLQ integration approximates the volume integrals by a weighted sum of point mass effects.

The accuracy of the integration depends on the number of point masses used in the summation. Ku (1977) showed that it also depends on the ratio between the distance to the computation point and the distance between adjacent nodes. Figure 2 illustrates this effect on the g_{xy} gravity gradient component. The g_{xy} component was produced by a $7° \times 7° \times 20\ km$ tesseroid with $2.67\ g.cm^{-3}$ density and top at $z = 0\ km$. The maps were calculated on a regular grid with 100×100 points. Figure 2a shows the g_{xy} component calculated at 400 km height using GLQ with order two ($2 \times 2 \times 2 = 8$ point masses). Figure 2b shows g_{xy} computed with order two GLQ as well but at 150 km height. Notice that the computed effect is concentrated around each point mass of the GLQ (black dots) and does not resemble the effect of a tesseroid. Ku (1977) determined an ad hoc criterion that the distance between point masses (quadrature nodes) should be smaller than the minimum distance to the computation point. Thus, if a computation point is too close to the tesseroid one would have to decrease the distance between the point masses in order to obtain an accurate result. One way to accomplish this would be increase the order of the quadrature N in all three directions. Figure 2c shows the g_{xy} component calculated at 150km height but with a GLQ order of 30 ($30 \times 30 \times 30 = 27,000$ point masses). The computed g_{xy} component more closely resembles the expect results for a single tesseroid (Asgharzadeh et al., 2007).

2.1 Adaptive discretization

Li et al. (2011) proposed an alternative method for decreasing the distance between point masses on the quadrature nodes aiming at achieving an accurate integration. Instead of increasing the GLQ order, they keep it fixed to a given number and divide the tesseroid into smaller volumes. The sum of the effects of the smaller tesseroids is equal to the gravitational effect of the larger tesseroid. This division effectively decreases the distance between nodes because of the smaller size of the tesseroids. The criterion for dividing a tesseroid is that the distance to the computation point should be smaller than a constant times the size of the tesseroid. This is analogous to the criterion proposed by Ku (1977) because the size of the tesseroid serves as a proxy for the distance between point masses. This proce-

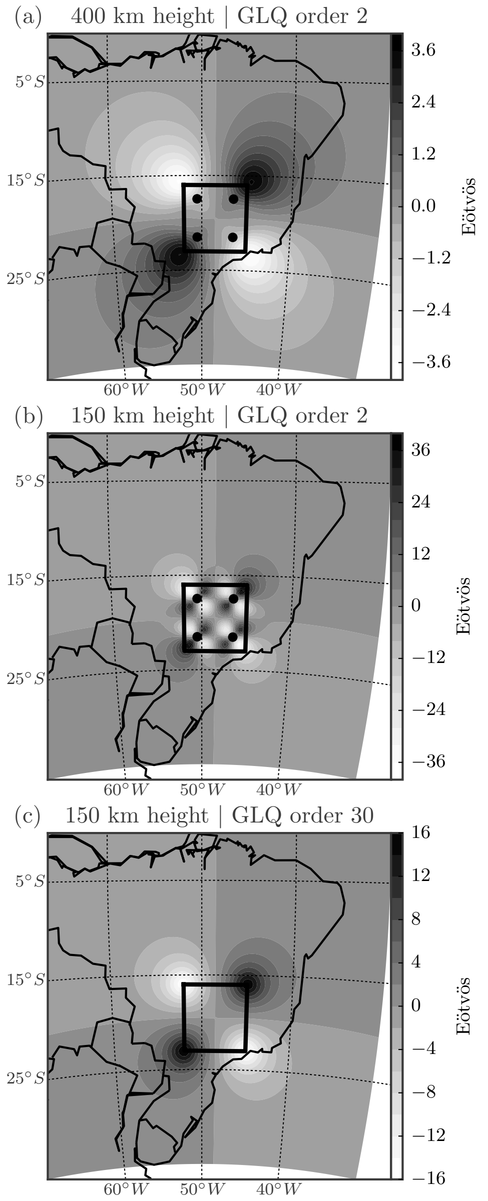

Figure 2: Example of the effect of varying the computation height and the number of point masses in the Gauss-Legendre Quadrature. Black circles represent the horizontal location of the point masses. a) g_{xy} calculated at 400 km height using GLQ order 2 ($2 \times 2 \times 2 = 8$ point masses). b) At 150 km height and GLQ order 2, the result resembles that of four point masses instead of a single tesseroid. This effect was shown by Ku (1977). c) At 150 km but with a higher GLQ order of 30. In (c) the horizontal locations of the point masses were not shown. Notice that the results shown in (c) are similar to that expected for a single mass source.

dure is repeated recursively until all tesseroids are within the acceptable ratio of distance and size or a minimum size is achieved.

The advantage of this adaptive discretization is that the number of point masses is only increased in parts of the tesseroid that are closer to the computation point. Notice that the alternative approach of simply increasing the order of the GLQ would increase the number of point masses evenly throughout the whole tesseroid.

3 Implementation

We have implemented the calculation of the tesseroid gravitational fields with adaptive discretization in version 1.2 of the open-source package *Tesseroids*. It is freely available online[1][2] under the BSD 3-clause open-source license. An archived version of the source code is also available as part of this article.

Tesseroids consists of command-line programs written in the C programming language. The package includes programs to calculate the gravitational fields of tesseroids and rectangular prisms (in both Cartesian and spherical coordinates). All programs receive input through command-line arguments and the standard input channel ("STDIN") and output the results through the standard output channel ("STDOUT"). For example, the command to generate a regular grid with $NLON \times NLAT$ points, calculate g_z and g_{zz} caused by the tesseroids in a file "MODELFILE", and save the results to a file called "OUTPUT" is:

```
tessgrd -rW/E/S/N -bNLON/NLAT -zHEIGHT | \
    tessgz MODELFILE | \
    tessgzz MODELFILE > OUTPUT
```

The *src* folder of the source code archive contains the C files that build the command-line programs (e.g., *tessgz.c*). The *src/lib* folder contains the source files that implement the numerical computations. We will not describe here the implementation of the input/output parsing and other miscellanea. Instead, we will focus on the details of the Gauss-Legendre Quadrature integration of equations 1-3 and the adaptive discretization of tesseroids.

[1]http://tesseroids.leouieda.com
[2]http://dx.doi.org/10.5281/zenodo.16033

3.1 Numerical integration

The source file *src/lib/glq.c* contains the code necessary to perform a Gauss-Legendre Quadrature integration. The first step in the GLQ is to compute the locations of the discretization points (i.e., the point masses). These points are roots of Legendre polynomials. Precomputed values are available for low order polynomials, typically up to order five. For flexibility and to compute higher order roots, we use the multiple root-finder algorithm of Barrera-Figueroa et al. (2006). The additional computational load is minimal because the root-finder algorithm must be run only once per program execution. The root-finder is implemented in functions *glq_nodes* and *glq_next_root*. The computed roots will be in the range $[-1, 1]$ and must be scaled to the integration limits (the physical boundaries of the tesseroid) using function *glq_set_limits* (see equation 12).

The GLQ weights (equation 13) are computed by function *glq_weights*. Both the computed roots and weights are stored in a data structure (a C *struct*) called *GLQ*. Function *glq_new* handles memory allocation, calculates the roots and weights, and returns the complete *GLQ* structure.

The numerical integration of the tesseroid gravitational fields is performed by the functions in module *src/lib/grav_tess.c*. Functions *tess_pot*, *tess_gx*, *tess_gy*, and so on, compute the gravitational fields of a single tesseroid on a single computation point. These functions require three *GLQ* structures, each containing the roots and weights for GLQ integration in the three dimensions. The roots must be scaled to the integration limits $[\lambda_1, \lambda_2], [\phi_1, \phi_2], [r_1, r_2]$ (see equations 1-3). The integration consists of three loops that sum the weighted kernel functions evaluated at each GLQ point mass (the scaled roots).

The biggest bottlenecks for the numerical integration are the number of point masses used and the evaluation of the trigonometric functions in equations 1-3 inside the inner loops. Better performance is achieved by pre-computing the sine and cosine of latitudes and moving some trigonometric function evaluations to the outer loops.

3.2 Implementation of adaptive discretization

Our implementation of the adaptive discretization algorithm differs in a few ways from the one proposed by Li et al. (2011). In Li et al. (2011), a tesseroid will be divided when the smallest distance between it and the computation point is smaller than a constant times the largest dimension of the tesseroid. Instead of the smallest distance, we use the easier to calculate distance between the computation point (r, λ, ϕ) and the geometric center of the tesseroid (r_t, λ_t, ϕ_t)

$$d = \left[r^2 + r_t^2 - 2rr_t \cos\psi_t\right]^{\frac{1}{2}}, \quad (16)$$

$$\cos\psi_t = \sin\phi \sin\phi_t + \cos\phi \cos\phi_t \cos(\lambda - \lambda_t). \quad (17)$$

Our definition of the dimensions of the tesseroid (the "side lengths" of Li et al. (2011)) along longitude, latitude, and radius, respectively, are (Figure 3a)

$$L_\lambda = r_2 \arccos(\sin^2\phi_t + \cos^2\phi_t \cos(\lambda_2 - \lambda_1)), \quad (18)$$

$$L_\phi = r_2 \arccos(\sin\phi_2 \sin\phi_1 + \cos\phi_2 \cos\phi_1), \quad (19)$$

$$L_r = r_2 - r_1. \quad (20)$$

L_λ and L_ϕ are arc-distances measured along the top surface of the tesseroid (Figure 3a). Specifically, L_λ is measured long the middle latitude of the tesseroid (ϕ_t).

To determine if a tesseroid must be divided, we check if

$$\frac{d}{L_i} \geq D, \quad (21)$$

for each $i \in (\lambda, \phi, r)$. D is a positive scalar hereafter referred to as the "distance-size ratio". If the inequality holds for all three dimensions, the tesseroid is not divided. Thus, the distance-size ratio determines how close the computation point can be before we must divide the tesseroid. The value of D is indirectly responsible for the accuracy of the solution and the computation time. We will explore the relationship with the accuracy in the following section.

Figure 3 shows examples of the resulting tesseroid models after adaptive discretization. Figure 3a shows the initial tesseroid and computation point P. Figures 3b-d are the result of adaptive discretization using different values of the distance-size ratio D, respectively, $D = 1$, $D = 2$, and $D = 6$. The number of tesseroids in the resulting discretization is, respectively, 4, 38, and 936.

Instead of using recursive function calls, as originally proposed by Li et al. (2011), we use a stack-based implementation of the algorithm. Stacks are array-like data structures with a particular way of inserting and removing elements from it. In a stack,

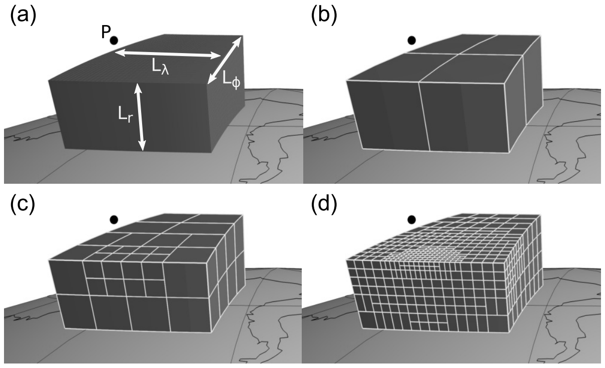

Figure 3: Adaptive discretization of the tesseroid shown in (a) for a computation point P using the distance-size ratio D equal to (b) 1, (c) 2, and (d) 6. L_r, L_ϕ, and L_λ are the dimensions of the tesseroid. Note that increasing D results in a fine division of the tesseroid close the computation point and a coarser division further away.

one can only insert elements to the top of the stack (the last empty position). Likewise, one can only remove the last element of the stack (commonly referred to as "popping" the stack). Because of these restrictions, stacks are also known as "Last-In-First-Out" (LIFO) data structures.

The discretization algorithm is implemented in function *calc_tess_model_adapt* of the file *src/lib/grav_tess.c*. This function calculates the effect of a single tesseroid on a single computation point. The stack of tesseroids is represented by the *stack* variable, an array of *TESSEROID* structures. We must define a maximum size for the stack to allocate memory for it. Defining a maximum size allows us to avoid an infinite loop in case the computation point is on (or sufficiently close to) the surface of the tesseroid. We use the integer *stktop* to keep track of the index of the last element in the stack (the top of the stack).

Below, we describe the algorithm to calculate the effect of a single tesseroid from the input model on a single computation point. The algorithm starts by creating an empty stack of tesseroids. Then, the stack is initialized with the single input tesseroid. The initialization is done by copying the tesseroid into the stack and setting *stktop* to zero (the first element). It is important to note that the stack is not the input tesseroid model. Instead, it is a buffer used to temporarily store each stage of the discretization algorithm.

Once the stack is initialized, the steps of the algorithm are:

1. "Pop" the stack (i.e., take the last tesseroid from it). This will cause *stktop* to be reduced by one. This tesseroid is the one that will be evaluated in the following steps.

2. Compute the distance d (equation 16) between the geometric center of the tesseroid and the computation point.

3. Compute the dimensions of the tesseroid L_λ, L_ϕ, and L_r using equations 18-20.

4. Check the condition in equation 21 for each dimension of the tesseroid.

5. If all dimensions hold the inequality 21, the tesseroid is not divided and its gravitational effect is computed using the Gauss-Legendre Quadrature (equations 1-3 and 14). We use a GLQ order of two for all three dimensions ($2 \times 2 \times 2 = 8$ point masses) by default. This value can be changed using a command-line argument of the modeling programs.

6. If any of the dimensions fail the condition:

   (a) Divide the tesseroid in half along each dimension that failed the condition.

   (b) Check if there is room in the stack for the new tesseroids (i.e.,the number of new elements plus *stktop* is smaller than the maximum stack size). If there isn't, warn the user of a "stack overflow" and compute the effect of the tesseroid, as in step 5. If there is room in the stack, place the smaller tesseroids into the stack.

7. Repeat the above steps until the stack is empty (*stktop* is equal to -1).

The algorithm above is repeated for every tesseroid of the input model and the results are summed. This will yield the gravitational effect of the input tesseroid model on a single point. Thus, the computations must be repeated for every computation point. The whole algorithm can be summarized in the following pseudo-code.

```
Initialize the output array with zeros.
for tesseroid in model:
  for point in grid:
    Initialize the stack with tesseroid.
    stktop = 0
    while stktop >= 0:
      Perform steps 1-6 of the algorithm.
    Sum the calculated value to the output.
```

This stack-based implementation has some advantages over the original recursive implementation, namely: (1) It gives the developer more control over the recursion step. (2) In general, it is faster because it bypasses the overhead of function calls. In recursive implementations, the developer has no control over the maximum number of consecutive recursive calls (i.e., the "recursion depth"). This limit may vary with programming language, compiler, and operating system. Overflowing the maximum recursion depth may result in program crashes, typically with cryptic or inexistent error messages. In the stack-based implementation, the developer has complete control. Overflowing of the stack can be handled gracefully with an error message or even performing a suitable approximation of the result.

3.3 Code for figures and error analysis

The error analysis and all figures in this article were produced in IPython notebooks (Pérez and Granger, 2007). The notebook files combine source code in various programming languages, program execution, text, equations, and the figures generated by the code into a single document. We used the following Python language libraries to perform the error analysis and generate figures: *pandas* by McKinney (2010), *matplotlib* by Hunter (2007) for 2D figures and maps, and *Mayavi* by Ramachandran and Varoquaux (2011) for 3D figures.

The IPython notebooks and the data generated for the error analysis, as well as instructions for installing the software and running the programs, are also included in the source code archive that accompanies this article. Alternatively, all accompanying material is available in an online repository[3].

4 Evaluation of the accuracy

The key controlling point of the adaptive discretization algorithm is the distance-size ratio D (equation 21). The specific value chosen for D determines how many divisions will be made (Figure 3). Thus, D indirectly controls both the accuracy of the integration and the computation time. In this section,

[3]https://github.com/pinga-lab/paper-tesseroids

	Grid location	Grid height	Tesseroid size
Experiment 1 (pole)	89N–90N/0E–1E	2 km	$1^\circ \times 1^\circ$
Experiment 2 (equator)	0N–1N/0E–1E	2 km	$1^\circ \times 1^\circ$
Experiment 3 (260 km)	89N–90N/0E–1E	260 km	$1^\circ \times 1^\circ$
Experiment 4 (30° size)	60N–90N/0E–30E	2 km	$30^\circ \times 30^\circ$

Table 1: Parameters of the numerical experiments to quantify the accuracy of the numerical integration. All grids had 10×10 regularly spaced computation points at a constant height. Tesseroids used to discretize the spherical shell had 1 km thickness and the horizontal dimensions shown in the table.

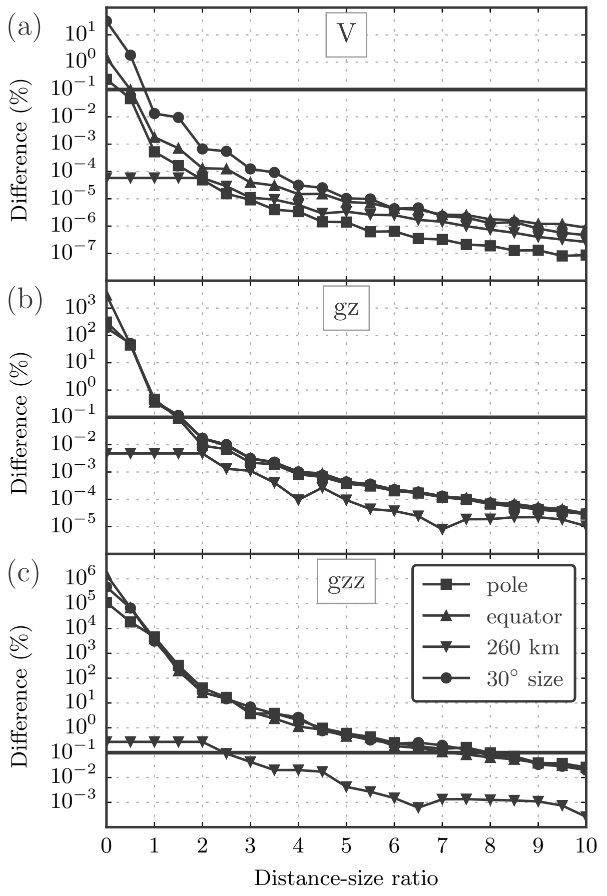

Figure 4: The maximum difference between the computed tesseroid and shell effects as a function of the distance-size ratio D for (a) the gravitational potential, (b) g_z, and (c) g_{zz}. The difference is given as a percentage of the shell effect. Curves correspond to the different tesseroid models and computation grids shown in Table 1. The horizontal solid black line marks the established error threshold of 0.1%. A value of $D = 0$ means that no divisions are made.

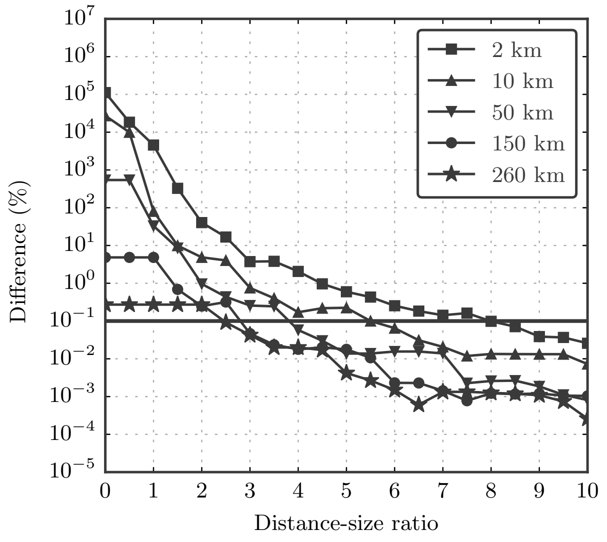

Figure 5: Difference between the computed g_{zz} for the spherical shell and the tesseroid model at different heights. Curves show the maximum difference as a percentage of the shell value. The horizontal solid black line marks the established error threshold of 0.1%. A value of $D = 0$ means that no divisions are made.

we investigate the relationship between the distance-size ratio and the integration error. We perform the analysis for the gravitational potential, acceleration, and gradient tensor components to evaluate if the same value of D yields compatible error levels for different fields.

The reference against which we compare the computed tesseroid fields is a homogeneous spherical shell. The shell has analytical solutions along the polar axis (LaFehr, 1991; Mikuška et al., 2006; Grombein et al., 2013) and can be perfectly discretized into tesseroids. We chose a spherical shell with a thickness of 1 km, density of 2670 $kg.m^{-3}$, bottom at height 0 km above the reference sphere, and top at 1 km height. We produced tesseroid models of the shell by discretizing it along the horizontal dimensions into a regular mesh.

Figure 2 shows that the largest errors are spread over on top of the tesseroid. Thus, calculating the tesseroid fields at a single point might not capture the point of largest error. Instead, we calculate the effect of the tesseroid model on a regular grid of 10×10 points at different geographic locations (see Table 1). Fortunately, the symmetry of the shell allows us to consider the computation point at any geocentric coordinate. Therefore, the effect of the shell will be same along the entire grid. We compute the differences between the effects of the shell and the tesseroid model on the grid. However, we will consider only the largest error in our analysis.

We placed the grid on top of a particular tesseroid to increase the chances of capturing the true largest integration error. We calculate the errors for values of the distance-size ratio D varying from 0 (i.e., no divisions) to 10 in 0.5 intervals. Furthermore, we repeated the error analysis in four different numerical experiments, each with computation grids at different locations and different tesseroid model sizes. Table 1 describes the different numerical experiments and the corresponding parameters of the computation grid and tesseroid model.

Figure 4 shows the maximum difference between the shell and tesseroid fields as a function of D for the four experiments. The differences are given as a percentage of the shell value. We established a maximum tolerated error of 0.1%, represented by the horizontal solid lines in Figure 4. Only results for the gravitational potential, g_z, and g_{zz} are shown. The results for the other diagonal components of the gravity gradient tensor are similar to g_{zz}. Figures for these components can be found in the supplementary material (see section "Code for figures and error analysis").

For the potential V, a distance-size ratio $D = 1$ guarantees that the curves for all experiments are below the 0.1% error threshold. For g_z, the same is achieved with $D = 1.5$. Conversely, g_{zz} requires a value of $D = 8$ to achieve an error level of 0.1%. For a computation height of 260 km, the error curve for g_{zz} intercepts the error threshold line at $D = 2.5$. This behavior suggests that the error curves for g_{zz} might depend on the computation height. To test this hypothesis, we computed the error curves for g_{zz} at heights 2, 10, 50, 150, and 260 km. Figure 5 shows the results for g_{zz} at varying computation heights. Notice that the distance-size ratio required to achieve 0.1% accuracy decreases as the computation height increases. For example, computation at 260 km height requires $D = 2.5$ whereas at 10 km height a value of $D = 5.5$ is required to achieve the same accuracy. One can take advantage of this behavior to reduce the distance-size ratio for computations of the gravity gradient tensor at high altitudes, saving computation time.

We have implemented the values of the distance-size ratio producing 0.1% accuracy determined above as defaults for the software *Tesseroids*. We chose the conservative value of $D = 8$ for the gravity gradient components as a fail-safe alternative. Users can control the value of D used in the computations through command-line arguments to achieve greater performance at the cost of accuracy.

5 Conclusions

We have presented the open-source software *Tesseroids*. It consists of command-line programs, written in the C programming language, to perform the forward modeling of gravitational fields in spherical coordinates. The fields are calculated from a mass model composed of spherical prisms, the so-called tesseroids. The volume integrals of the gravitational fields of a tesseroid are solved numerically using the Gauss-Legendre Quadrature (GLQ). The GLQ approximates the volume integrals by weighted sums of point mass effects. The error of the GLQ integration increases as the computation point gets closer to the tesseroid. To counter this effect, the accuracy of the GLQ integration can be increased by using more point masses or by dividing each tesseroid into smaller ones.

We have implemented and improved upon an adaptive discretization algorithm to achieve an optimal division of tesseroids. Tesseroids are divided into more parts closer to the computation point, where more point masses are needed. Our implementation of the adaptive discretization uses a "stack" data structure in place of the originally proposed recursive implementation. As a rule of thumb in procedural languages (like C), stack-base implementations are computationally faster than the equivalent code using function recursion. Furthermore, the stack-based algorithm allows more control over errors when too many divisions are necessary. The adaptive discretization is controlled by a scalar called the distance-size ratio (D). The algorithm ensures that all tesseroids will have dimensions smaller than D times the distance to the computation point. The value of D indirectly controls the accuracy of the integration as well as the computation time.

We performed an error analysis to determine the optimal value of D required to achieve a target accuracy. We used a spherical shell as a reference to calculate the computation error of our algorithm for different values of D. Our results show that the values of D required to achieve a maximum error of 0.1% of the shell values are 1 for the gravitational potential, 1.5 for the gravitational acceleration, and 8 for the gravity gradients. Previous assumptions in the literature were that accurate results are guaranteed if the distance to the tesseroid is larger than the distance between point masses. This condition was previously applied indiscriminately to both the gravitational acceleration and the gravity gradients. That assumption is equivalent to using $D = 1.5$ for all fields. Our results show that this is valid for the gravitational acceleration and results in a 0.1% computation error. This is expected because the original study that determined the above condition was performed on the vertical component of gravitational acceleration. However, applying the same condition to the gravity gradients produces an error of the order of 10^2%.

For the gravity gradients in particular, the distance-size ratio required for 0.1% error decreases with height. We believe this is because the decay factor for the gravity gradient components is d^{-3}, whereas the discretization algorithm uses d/L_i. As the computation point becomes closer to the tesseroid, the field increases more rapidly than the algorithm increases the amount of discretization. Hence, a higher value of D (i.e., more discretization) is required.

The values of the distance-size ratio determined above were incorporated as defaults in the software *Tesseroids*. We chose the value $D = 8$ for the gravity gradients as a conservative default. If the user desires, the value of D used can be controlled by a command-line argument.

In situations that require many tesseroid divisions, the stack used in the algorithm will overflow and further divisions become impossible. The current implementation warns the user that the overflow occurred and proceeds with the GLQ integration without division. Future improvements to the algorithm include a better way to handle such situations as they arise. An alternative would be to replace the tesseroid by an equivalent right rectangular prism and compute its effects instead. This would allow accurate computations at smaller distances. Furthermore, the computation time increases drastically as the computation point gets closer to the tesseroid. This effect can be prohibitive for computing the gravity gradients at relatively low heights (e.g., for terrain corrections of ground or airborne surveys). Further investigation of different criteria for dividing the tesseroids could yield better performance through a reduced number of divisions.

6 Acknowledgments

We are indebted to the developers and maintainers of the open-source software without which this work would not have been possible. The authors thank associate editor Joe Dellinger, reviewer Roman Pasteka, and four anonymous reviewers for their hard work and helpful comments. The authors were supported in this research by a fellowship (VCFB) from Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) and a scholarship (LU) from Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Brazil. Additional support for the authors was provided by the Brazilian agency FAPERJ (grant E-26/103.175/2011) and by the GOCE-Italy project (ASI).

References

Álvarez, O., M. Gimenez, C. Braitenberg, and A. Folguera, 2012, GOCE satellite derived gravity and gravity gradient corrected for topographic effect in the south central andes region: GOCE derivatives in the south central andes: Geophysical Journal International, **190**, 941–959.

Asgharzadeh, M. F., R. R. B. von Frese, H. R. Kim, T. E. Leftwich, and J. W. Kim, 2007, Spherical prism gravity effects by Gauss-legendre quadrature integration: Geophysical Journal International, **169**, 1–11.

Barrera-Figueroa, V., J. Sosa-Pedroza, and J. López-Bonilla, 2006, Multiple root finder algorithm for legendre and Chebyshev polynomials via Newton's method: Annales Mathematicae et Informaticae, 3–13.

Bouman, J., J. Ebbing, and M. Fuchs, 2013a, Reference frame transformation of satellite gravity gradients and topographic mass reduction: Journal of Geophysical Research: Solid Earth, **118**, 759–774.

Bouman, J., J. Ebbing, S. Meekes, R. Abdul Fattah, M. Fuchs, S. Gradmann, R. Haagmans, V. Lieb, M. Schmidt, D. Dettmering, and W. Bosch, 2013b, GOCE gravity gradient data for lithospheric modeling: International Journal of Applied Earth Observation and Geoinformation, **35**, 16–30.

Braitenberg, C., 2015, Exploration of tectonic structures with GOCE in Africa and across-continents: International Journal of Applied Earth Observation and Geoinformation, **35**, 88–95.

Braitenberg, C., P. Mariani, and A. De Min, 2013, The European alps and nearby orogenic belts sensed by GOCE: Bollettino di Geofisica Teorica e Applicata, **54**, 321–334.

Braitenberg, C., P. Mariani, J. Ebbing, and M. Sprlak, 2011, The enigmatic Chad lineament revisited with global gravity and gravity-gradient fields: Geological Society, London, Special Publications, **357**, 329–341.

Fullea, J., J. Rodríguez-González, M. Charco, Z. Martinec, A. Negredo, and A. Villaseñor, 2014, Perturbing effects of sub-lithospheric mass anomalies in GOCE gravity gradient and other gravity data modelling: Application to the atlantic-mediterranean transition zone: International Journal of Applied Earth Observation and Geoinformation, **35**, 54–69.

Grombein, T., K. Seitz, and B. Heck, 2013, Optimized formulas for the gravitational field of a tesseroid: Journal of Geodesy, **87**, 645–660.

Heck, B., and K. Seitz, 2007, A comparison of the tesseroid, prism and point-mass approaches for mass reductions in gravity field modelling: Journal of Geodesy, **81**, 121–136.

Hildebrand, F. B., 1987, Introduction to numerical analysis: Dover Publications.

Hunter, J. D., 2007, Matplotlib: A 2d graphics environment: Computing in Science & Engineering, **9**, 90–95.

Ku, C. C., 1977, A direct computation of gravity and magnetic anomalies caused by 2-and 3-dimensional bodies of arbitrary shape and arbitrary magnetic polarization by equivalent-point method and a simplified cubic spline: Geophysics, **42**, 610–622.

LaFehr, T., 1991, An exact solution for the gravity curvature (Bullard B) correction: GEOPHYSICS, **56**, 1179–1184.

Li, Z., T. Hao, Y. Xu, and Y. Xu, 2011, An efficient and adaptive approach for modeling gravity effects in spherical coordinates: Journal of Applied Geophysics, **73**, 221–231.

Mariani, P., C. Braitenberg, and N. Ussami, 2013, Explaining the thick crust in paraná basin, Brazil, with satellite GOCE gravity observations: Journal of South American Earth Sciences, **45**, 209–223.

McKinney, W., 2010, Data structures for statistical computing in python: Proceedings of the 9th Python in Science Conference, 51 – 56.

Mikuška, J., R. Pašteka, and I. Marušiak, 2006, Estimation of distant relief effect in gravimetry: GEOPHYSICS, **71**, J59–J69.

Pérez, F., and B. E. Granger, 2007, IPython: a system for interactive scientific computing: Computing in Science and Engineering, **9**, 21–29.

Ramachandran, P., and G. Varoquaux, 2011, Mayavi: 3d visualization of scientific data: Computing in Science & Engineering, **13**, 40–51.

Reguzzoni, M., D. Sampietro, and F. Sanso, 2013, Global Moho from the combination of the CRUST2.0 model and GOCE data: Geophysical Journal International, **195**, 222–237.

Uieda, L., 2015, A tesserioid (spherical prism) in a geocentric coordinate system with a local-North-oriented coordinate system: figshare, available from: http://dx.doi.org/10.6084/m9.figshare.1495525. (Accessed May 2016).

Wild-Pfeiffer, F., 2008, A comparison of different mass elements for use in gravity gradiometry: Journal of Geodesy, **82**, 637–653.