Geophys. J. Int. (2011) **187,** 161–177 doi: 10.1111/j.1365-246X.2011.05167.x

Optimal, scalable forward models for computing gravity anomalies

Dave A. May[1] and Matthew G. Knepley[2]

[1]*Institute of Geophysics, Department of Earth Sciences, ETH Zürich, Zürich, Switzerland. E-mail: dave.mayhem23@gmail.com*
[2]*Computation Institute, University of Chicago, Illinois, USA*

Accepted 2011 July 25. Received 2011 July 25; in original form 2011 May 21

SUMMARY

We describe three approaches for computing a gravity signal from a density anomaly. The first approach consists of the classical 'summation' technique, while the remaining two methods solve the Poisson problem for the gravitational potential using either a finite-element (FE) discretization employing a multilevel pre-conditioner, or a Green's function evaluated with the fast multipole method (FMM). The methods using the Poisson formulation described here differ from previously published approaches used in gravity modelling in that they are optimal, implying that both the memory and computational time required scale linearly with respect to the number of unknowns in the potential field. Additionally, all of the implementations presented here are developed such that the computations can be performed in a massively parallel, distributed memory-computing environment. Through numerical experiments, we compare the methods on the basis of their discretization error, CPU time and parallel scalability. We demonstrate the parallel scalability of all these techniques by running forward models with up to 10^8 voxels on 1000s of cores.

Key words: Numerical solutions; Numerical approximations and analysis; Gravity anomalies and Earth structure; Geopotential theory.

1 INTRODUCTION

1.1 Background

The use of forward models to compute synthetic gravity signals is necessary to conduct inversions of the subsurface density structure. Given a volume Ω_M over which we have a density field $\rho(\mathbf{x})$, the gravity attraction at a point $\mathbf{r} = (r, s, t)$ due to this body can be computed via

$$\mathbf{g}(\mathbf{r}) = G \int_{\Omega_M} \rho(\mathbf{x}) \frac{\mathbf{r} - \mathbf{x}}{[(r-x)^2 + (s-y)^2 + (t-z)^2]^{3/2}} \, \mathrm{d}V. \tag{1}$$

An alternative way to compute the gravity field is to solve the gravitational potential equation

$$\nabla^2 \phi = -4\pi G \rho(\mathbf{x}) \qquad \text{in } \Omega_\infty, \tag{2}$$

where ϕ is the potential, G is the gravitational constant, Ω_∞ denotes the entire free space and we assume that $\rho(\mathbf{x}) = 0, \ \forall \ \mathbf{x} \notin \Omega_M$. The potential is subject to the following Dirichlet boundary condition:

$$\phi = 0, \qquad \text{at } \mathbf{x} = \infty. \tag{3}$$

The gravity field sought is given by the gradient of the potential ϕ;

$$\mathbf{g}(\mathbf{x}) = -\nabla \phi. \tag{4}$$

The physical model is depicted in Fig. 1. Forward gravity models typically fall into one of two categories: summation-based techniques which evaluate eq. (1), or partial differential equation (PDE)-based techniques which solve the gravitational potential formulation in eqs (2)–(4).

The summation methods require the subsurface density structure to be discretized into a set of volumes. At each location $\mathbf{r}$, in the model domain where a gravity signal is sought, the gravitational contribution from each density element in the domain is evaluated using eq. (1) and summed. The summation methods differ in the manner in which the integral expression in eq. (1) is evaluated. Several analytic approaches exist in which a closed-form expression for eq. (1) is used in either Cartesian (see Li & Chouteau 1998, for an overview) or spherical coordinates (Johnson & Lithehiser 1972; Smith *et al.* 2001). The limitation of analytic expression is that one is forced to choose a spatial discretization for the density structure which is orthogonal to the coordinate system, and the density is usually required to be constant over each element. The complexities and discretization restrictions of the analytic method can be overcome by using a sufficiently accurate

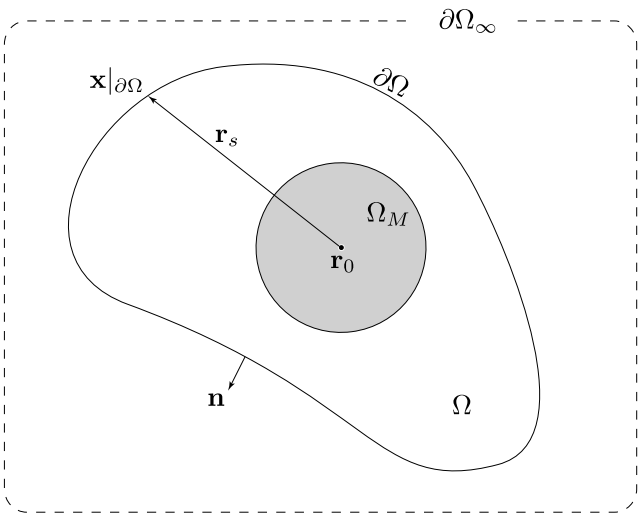


Figure 1. Problem domain for computing gravity. Here, we denote the infinite domain boundary by $\partial\Omega_\infty$, the model domain by Ω and density anomaly domain by Ω_M. The centre of mass of Ω_M is denoted by $\mathbf{r}_0$ and $\mathbf{n}$ is the outward pointing normal to the boundary of Ω.

quadrature scheme to approximate eq. (1). This approach permits any spatial discretization to be used provided a high-accuracy quadrature rule can be defined over the geometry of each cell used in the discretization (Asgharzadeh *et al.* 2007).

Recently, there has been some interest in using PDE-based approaches to compute gravity anomalies, as these methods have been demonstrated to be both faster and produce more accurate forward models than the summation techniques. In Cai & Wang (2005), a finite-element (FE) method was used to obtain the solution to the Poisson equation. They favoured the FE method over the finite-difference method as the former allowed more geometric freedom in meshing the density anomalies and the formulation easily permitted a variable density field within each voxel. Their formulation used a Robin-type boundary condition to approximate the boundary condition in eq. (3). The method was regarded as being 'fast' since within a finite-size domain, the Robin condition yielded a smaller error than setting $\phi = 0$ on the boundary of a finite domain. That is, the convergence of the error using this method was faster than simply setting $\phi = 0$ on the boundary of the finite domain. In contrast, Farquharson & Mosher (2009) employed a finite-difference discretization to solve eq. (2), where the boundary condition $\phi = 0$ at $\mathbf{x} = \infty$ is approximated by ensuring that the model boundaries are 'far' from the density anomaly, which in their work constituted using a model domain with side lengths six times larger than the side length of the anomaly.

The development of fast and efficient forward models is crucial to enable high-resolution inversion to be performed. In considering the computation complexity of the summation algorithm, we see that if we discretize the domain with N density elements and we have M measurements, that is, locations where we will evaluate the gravity, the calculation will require $\mathcal{O}(MN)$ time. Given the ease with which gravity measurements can be made on a regional scale using either a land-based relative gravimeter or via airborne measurements, or on a global scale using satellite-based gravimetry, applied geophysics studies may typically have values of M on the order of 10 000. The number of measurements M is continually increasing as new techniques are developed, or existing techniques become affordable or automated. We note that the computational cost of evaluating the gravity contribution from one element via eq. (1) is not insignificant. Even the simplest one-point quadrature rule requires: five additions, seven multiplications and one square root, which is equivalent to the cost of $\sim$20 multiplications (Fog 2011).

Using the PDE approach, one obtains the value of the potential over the entire domain, from which the gravity can be computed as a post-processing task. Consequently, the PDE approaches have a computational complexity which is not a strong function of the number of evaluation points, but instead is dominated by the complexity of the linear solver (X) used to obtain the potential, that is, the overall method scales according to $\mathcal{O}(N + X)$. If sparse direct factorizations (such as Cholesky or LU) are used, the solve time will scale like $X = \mathcal{O}(n^{3/2})$ in 2-D and $X = \mathcal{O}(n^2)$ in 3-D, where n is the number of unknowns used to represent the discrete potential field. The memory usage for these solvers is $\sim\mathcal{O}(n \log n)$ and $\sim\mathcal{O}(n^{4/3})$ for 2-D and 3-D, respectively (Li & Widlund 2007). If unpre-conditioned Krylov methods like conjugate gradient are used, the solve time will scale according to $X = \mathcal{O}(n^{3/2})$ and $\mathcal{O}(n^{4/3})$ in 2-D and 3-D, respectively. Numerous optimal multilevel pre-conditioners exist for the Poisson equation in which both the solve time and memory usage will scale like $\mathcal{O}(n)$ (Trottenbert *et al.* 2001).

1.2 This work

Here, we examine several variants of the summation method, an FE method with two types of boundary conditions and a fast multipole method (FMM) to compute synthetic gravity fields. Our examination of the different methods focuses on the accuracy and the algorithmic complexity (optimality) of the techniques. All of the methods used in this study are developed to be executed on massively parallel, distributed memory computer architectures. We also examine the parallel performance (scalability) of the three classes of the methods under consideration.

2 NUMERICAL TECHNIQUES

2.1 Summation

We considered three variants of the summation technique in this study. Each of the summation techniques is defined in a Cartesian coordinate system and used a structured mesh of hexahedral cells to discretize the density field. The model domain considered was always 'brick' shaped and thus was easily decomposed into a set of $M_x \times M_y \times M_z$ cells . Within each cell, the density is assumed to be constant. The first summation approach (which we identify as SUM-AN) uses the analytic expression from Li & Chouteau (1998) to evaluate the vertical component of the gravitational contribution $g_z(\mathbf{x})$, given by eq. (1). The other two methods we consider use either a one point Gauss (SUM-G1), or a two point Gauss (SUM-G2) quadrature scheme to evaluate the gravity integral.

Parallelism is achieved in the summation methods via a spatial decomposition of the mesh used to discretize the density field. The locations where the gravity field is required to be evaluated are duplicated on each processor. Every processor calculates a local gravitational contribution at each evaluation point from a subset of cells within the entire domain. This operation can be completed without any communication. The only communication required is a global reduction of the local gravity contributions from each processors local subdomain.

2.2 FE method

The Poisson equation in eq. (2) is solved using a standard Galerkin FE formulation (Hughes 1987). The variational form is given by

$$\int_{\Omega_\infty} v\nabla^2\phi \,\mathrm{d}V = 4\pi G \int_{\Omega_\infty} v\rho(\mathbf{x}) \,\mathrm{d}V, \tag{5}$$

where v is a test function which vanishes on all Dirichlet boundaries. Applying integration by parts to the second-order derivative in eq. (5), we obtain

$$-\int_{\Omega_\infty} \nabla v \,.\, \nabla\phi \,\mathrm{d}V + \int_{\partial\Omega_\infty} v\nabla\phi \,.\, \mathbf{n} \,\mathrm{d}S = 4\pi G \int_{\Omega_\infty} v\rho(\mathbf{x}) \,\mathrm{d}V. \tag{6}$$

Here, we consider using two different approaches to approximate the 'Dirichlet at infinity' boundary condition in eq. (3). Both methods first approximate the entire free space domain Ω_∞, by a finite-sized domain Ω, satisfying $\Omega_M \subseteq \Omega$. The first approximation of eq. (3) we consider simply requires that

$$\phi\,|_{\partial\Omega} = 0, \tag{7}$$

where $\partial\Omega$ denotes the boundary of Ω. Clearly, the larger the domain Ω is compared to the domain of the density anomaly Ω_M, the better the approximation. We will denote this particular boundary condition approximation as FEM-D.

The second approximate boundary condition we considered was introduced by Cai & Wang (2005) and consists of approximating the far-field gravitational attraction on a finite-sized domain Ω. The far-field gravity is approximated according to

$$\mathbf{g}\,|_{\partial\Omega} = \left.\frac{\phi}{\mathbf{r}_s}\right|_{\partial\Omega}, \tag{8}$$

where $\mathbf{r}_s = \mathbf{x}|_{\partial\Omega} - \mathbf{r}_0$ and $\mathbf{r}_0$ is the centroid of the density anomaly domain Ω_M. These quantities are indicated on Fig. 1. Using the definition of the potential from eq. (4), we can introduce eq. (8) naturally into the variational problem in eq. (6) as a Robin boundary condition. We denote this boundary condition approximation as FEM-GT. For a thorough description of the FE formulation and the implementation of the Robin boundary conditions, we refer readers to Cai & Wang (2005).

As in the summation method, the domain consisted of a brick-like geometry and was discretized with $M_x \times M_y \times M_z$ hexahedral elements. The discrete solution for ϕ was represented with piecewise trilinear (Q_1 basis) functions over each hexahedral element. The same mesh was used to define the density structure. In the FE implementation used here, the density was assumed to be constant over each element. The resulting discrete problem from the FE discretization yields the sparse matrix problem

$$[\mathbf{L} + \mathbf{F}]\,\mathbf{x} = \mathbf{b}, \tag{9}$$

where $\mathbf{x}$, $\mathbf{b}$ represent the discrete potential and force term, $\mathbf{L}$ is the discrete Laplacian and $\mathbf{F}$ is the term associated with the far-field boundary condition appearing in the surface integral in eq. (6). We note that $\mathbf{F} = \mathbf{0}$ when the FEM-D approach is used.

Following the solution of eq. (9), we compute the gravity within each element by interpolating the gradient of the trilinear basis functions used to approximate ϕ. This approach has the disadvantage that the gravity field computed is discontinuous across element boundaries. The reconstruction of a continuous C^0 nodal field from the gradient of an FE solution is a thoroughly studied problem. The Super Convergent Patch Recovery (SPR; Zienkiewicz & Zhu 1992) and the Recovery by Equilibrium of Patches (REP; Boroomand & Zienkiewicz 1997) are both appropriate techniques to recover an accurate nodal gravity field. In Cai & Wang (2005), a nodal gravity field was computed using a global L_2 projection. A local L_2 projection can also be used (Hughes 1987), which has the advantage of not requiring the solution of a global matrix problem. In practice, to enable the gravity field to be evaluated everywhere, a continuous gravity field defined on the nodes of the FE mesh is the most useful representation. In this work however, we only use the results of the gravity field to compute error norms, for which the element wise, discontinuous representation of the gravity field is sufficient.

The matrix problem in eq. (9) was solved using the Flexible Generalized Minimal Residual (FGMRES) Krylov method (Saad 2003), pre-conditioned with one V-cycle of geometric multigrid (GMG). The GMG pre-conditioner we used is fairly standard and we refer to Briggs *et al.* (2000); Wesseling (1992) and Trottenbert *et al.* (2001) for an introduction to these methods. Here, we briefly summarize the components used in our multigrid pre-conditioner.

The multigrid method uses a mesh hierarchy consisting of n_l levels. Each level in the hierarchy defines a mesh of different spatial resolution. In the results presented here, a grid refinement factor of 2 was used between each grid level. The mesh at level n_l has the finest resolution and represents the mesh used to discretize the potential field problem. The operator $\mathbf{A} = \mathbf{L} + \mathbf{F}$ was defined on each mesh within the hierarchy by re-discretizing the PDE. Trilinear interpolation was used to define the restriction operator $\mathbf{R}$, which is required to project nodal fields from a fine grid, to the next coarsest grid. Interpolation of fields from a coarse to fine grid was given by $\mathbf{R}^T$. On every grid level except the coarsest, we employed N_k Richardson's iterations, combined with a Jacobi pre-conditioner as our smoother. Given a vector $\mathbf{y}_k$ at iteration k, the application of the smoother is given by the following sequence:

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \mathrm{diag}(\mathbf{A})^{-1} (\mathbf{b} - \mathbf{A}\mathbf{y}_k) . \tag{10}$$

Unless otherwise stated, $N_k = 2$ was used in all experiments. On the coarsest grid level, the smoother was defined via an LU factorization.

In our Poisson solver, the action of $\mathbf{A}\mathbf{y}_k$, required by the smoother in eq. (10) (on all grid levels expect the coarsest) and during each FGMRES iteration (finest grid only), was defined in a matrix-free manner. Similarly, diag($\mathbf{A}$) was computed element-by-element, without explicitly assembling the full stiffness matrix $\mathbf{A}$. On the coarsest grid, $\mathbf{A}$ was explicitly assembled to allow an LU factorization to be performed.

At each iteration i of the Krylov method, we monitor the 2-norm of the residual $\mathbf{r}_i = \mathbf{b} - \mathbf{A}\mathbf{x}_i$. The current estimated solution $\mathbf{x}_i$ obtained from the iterative method was deemed to be converged if $\|\mathbf{r}_i\|_2 < 10^{-10}\|\mathbf{r}_0\|_2$, where $\mathbf{r}_0$ is the initial residual.

Support for parallel linear algebra, Krylov methods and the structured mesh representation were provided by the Portable Extensible Toolkit for Scientific (c)omputation (PETSc; Balay *et al.* 2010).

2.3 FMM

The FMM is an algorithm that accelerates the solution of an N-body problem

$$\mathbf{g}(\mathbf{x}'_j) = \sum_{i=1}^{N} \rho_i \, \mathbb{K}(\mathbf{x}'_j, \mathbf{x}_i), \tag{11}$$

which is simply a discrete form of eq. (1). Here, $\mathbf{g}(\mathbf{x}'_j)$ represents the gravitational field evaluated at a point $\mathbf{x}'_j$, where the field is generated by the influence of sources located at the set of points $\{\mathbf{x}_i\}$. The sources are often associated with particle-type objects, such as charged particles, or in this case rock masses. In summary: $\{\mathbf{x}'_j\}$ is a set of evaluation points; $\{\mathbf{x}_i\}$ is a set of source points with densities given by ρ_i and $\mathbb{K}(\mathbf{x}', \mathbf{x})$ is the kernel that governs the interactions between evaluation and source particles. The kernel for the gravitational interaction in three dimensions is given by

$$\mathbb{K}(\mathbf{x}'_j, \mathbf{x}_i) = \frac{\mathbf{x}' - \mathbf{x}}{|\mathbf{x}' - \mathbf{x}|^3}. \tag{12}$$

Obtaining the field $\mathbf{g}$ at all the evaluation points requires in principle $\mathcal{O}(MN)$ operations, for N source points and M evaluation points. The FMM obtains $\mathbf{g}$ approximately with a reduced operation count, $\mathcal{O}(M + N)$.

In the FMM algorithm, the influence of a cluster of particles is approximately represented by a series expansion, which is then used to evaluate far-away interactions with controllable accuracy. To accomplish this, the computational domain is hierarchically decomposed, allowing pairs of subdomains to be grouped into 'near' and 'far', with far interactions treated approximately. Fig. 2 illustrates such a hierarchical space decomposition for a 2-D domain, associated to a quadtree structure.

Using this decomposition of the computational domain, the sum in eq. (11) can be decomposed as

$$\mathbf{g}(\mathbf{x}'_j) = \sum_{k=1}^{N_{\mathrm{near}}} \rho_k \mathbb{K}(\mathbf{x}'_j, \mathbf{x}_k) + \sum_{k=1}^{N_{\mathrm{far}}} \rho_k \mathbb{K}(\mathbf{x}'_j, \mathbf{x}_k). \tag{13}$$

The first term, corresponding to the near field of an evaluation point, will have a small fixed size independent of N. The second sum of eq. (13), representing the far field, will be evaluated efficiently using a series approximation so that the total complexity for the evaluation is $\mathcal{O}(N)$. We will use the following terminology for our field approximations:

Multipole Expension (ME) is a p term series expansion that represents the influence of a cluster of particles at distances large with respect to the cluster radius.

Local Expension (LE) is a p term series expansion, valid only inside a subdomain, used to efficiently evaluate a group of MEs locally in a cluster of evaluation points.

The centre of the series for an ME is the centre of the cluster of source particles, and it converges only outside a given radius centred at the cluster of particles. In the case of an LE, the series is centred near an evaluation point and converges only inside a given radius.

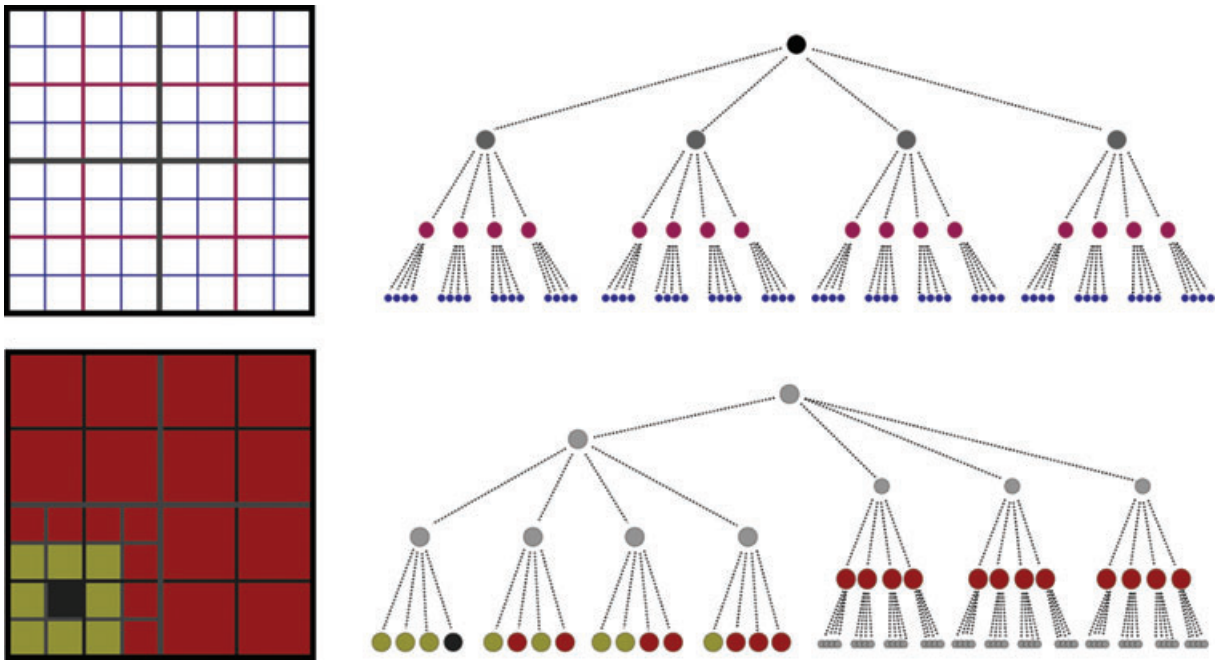


Figure 2. Quadtree decomposition of a 2-D domain: (a) presents a hierarchical tree related to the full spatial decomposition of the domain; (b) presents a coloured 2-D spatial decomposition for interacting with particles in the black box, and its equivalence on the tree. The near field is composed by the dark yellow boxes and the black box itself, while the far field is composed by the dark red coloured boxes. Note that the far field is composed of boxes of different levels of the tree structure. The relationships between the nodes of the tree simplify the process of composing the near and far domains.

The introduction of a single representation for a cluster of particles, via the multipole expansion, effectively permits a decoupling of the influence of the source particles from the evaluation points. This is a key idea, resulting in the factorization of the computations of MEs that are centred at the same point, so that the kernel can be written

$$\mathbb{K}(\mathbf{x}'_j, \mathbf{x}_i) = \sum_{m=0}^{p} C_m(\mathbf{x}_i) f_m(\mathbf{x}'_j). \tag{14}$$

This factorization allows pre-computation of terms that can be reused many times, reducing the complexity of evaluation from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$. Similarly, the local expansion is used to decouple the influence of an ME from the evaluation points. A group of MEs can be factorized into a single LE, which allows the $\mathcal{O}(N \log N)$ complexity to be further reduced to $\mathcal{O}(N)$. By representing MEs as LEs one can efficiently evaluate the effect of a group of clusters on a group of evaluation points.

2.4 Hierarchical space decomposition

To make use of the ME and LE, the domain must be decomposed into near and far subdomain pairs. A hierarchical decomposition provides an efficient implementation for this operation. The hierarchical subdivision of space is associated to a tree structure ('quadtree' structure in two dimensions, or an 'octree' structure in three dimensions) to represent each subdivision. The nodes of the tree structure are used to define the spatial decomposition, and different scales are obtained by looking at different levels. Consider Fig. 2(a), where a quadtree decomposition of the space is illustrated. The nodes of the tree at each level cover the entire domain. The domain covered by a parent box is further decomposed into smaller subdomains by its child nodes. As an example of its use in FMM, consider Fig. 2(b) where the near field for the 'black' coloured box is represented by the dark yellow coloured boxes, and the far field is composed by the dark red coloured boxes.

2.5 Overview of the algorithm

We use a diagram of the tree structure to illustrate the whole algorithm in one picture (Fig. 3). The importance of this presentation is that it relates the control flow and computation to the data structure used by FMM.

After the spatial decomposition stage, the FMM can be summarized in three stages: the upward sweep, the downward sweep and field evaluation. In the 'upward sweep', MEs are constructed for each node of the tree. For each leaf node, MEs are derived for each particle. On succeeding levels, these expansions are translated to the centre of the parent node and combined. This is shown in Fig. 3 by the black arrows going up from the nodes on the left side of the tree. In the 'downward sweep' phase, MEs are first transformed into LEs for all the cells in the 'interaction list' of a given box. This process is represented by the dashed red-coloured arrows in Fig. 3. For a given cell, the interaction list corresponds to the cells of the same level that are not nearest neighbours, but are children of the nearest neighbours of its parent cell.

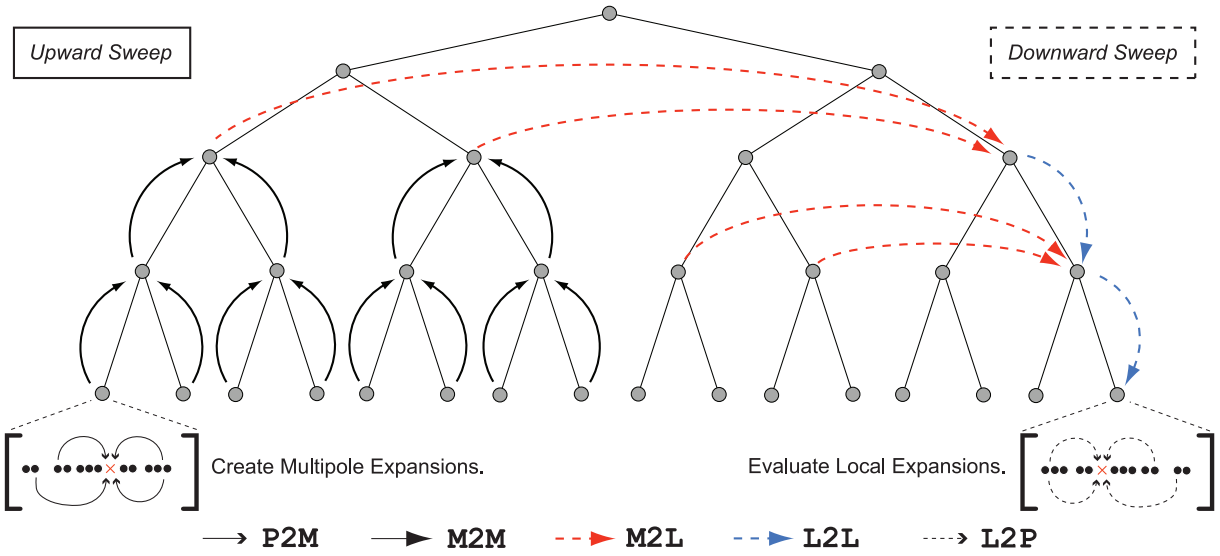


Figure 3. Overview of the FMM algorithm. The diagram illustrates the 'upward sweep' and the 'downward sweep' stages on the tree. The following operations are illustrated: P2P, transformation of particles into MEs (particle-to-multipole); M2M, translation of MEs (multipole-to-multipole); M2L, transformation of an ME into an LE (multipole-to-local); L2L, translation of an LE (local-to-local); L2P, evaluation of LEs at particle locations (local-to-particle).

After this series transformation, the LEs of upper levels are translated to the centres of child cells, and their influence is summed to obtain the complete far field for each leaf cell. This process is represented by the dashed blue-coloured arrows going down the right side of the tree in Fig. 3. At the end of the downward sweep, each box will have an LE that represents the complete far field for the box. Finally, during the 'field evaluation' phase, the total field is evaluated for every particle by adding the near-field and far-field contributions. The near field is obtained by directly computing the interactions between all the particles in the near domain of the box, consisting of nearest neighbour cells in the tree.

In this work, we used the open source PetFMM package (Cruz *et al.* 2010) to calculate the fast multipole operation in parallel. The PetFMM library was designed to offer both high serial performance and scalability, but also to be easily integrated into existing codes. The serial code is completely reused in the parallel setting so that we are never required to maintain two versions of the same algorithm. PetFMM leverages existing packages to keep its own code base small and clean. Parallel data movement is handled by the *Sieve* package (Knepley & Karpeev 2009) from PETSc (Balay *et al.* 2010, 2011), whereas load and communication are balanced using a range of different partitioners. In this work, we employed either a simple geometric-based partitioner which subdivides the space into $N_x \times N_y \times N_z$ cubes, or the graph partitioner ParMETIS (Karypis & Kumar 1998; Karypis 2011).

3 NUMERICAL EXPERIMENTS

To understand the discretization error and CPU time required by each of the different classes of forward models, we considered a synthetic gravity model for which we have an analytic solution for the vertical gravity component g_z. The model domain Ω consisted of a cube with side lengths $L = 600$ m, orientated such that $\Omega \equiv [0, 600] \times [0, 600] \times [-450, 150]$ m. Located at the centre of the domain was a cube with side lengths $H = 100$ m, to which we assigned the density, $\rho = 2000\ \mathrm{kg\,m^{-3}}$. The surrounding material in the remainder of the domain was regarded as void and assigned a density, $\rho = 0\ \mathrm{kg\,m^{-3}}$. The model setup is identical to that used in Farquharson & Mosher (2009). By regarding the dense cube as a simple prism, the analytic gravity field can be computed using the closed-form expression of Li & Chouteau (1998). The model setup and the analytic gravity field component g_z is shown in Fig. 4.

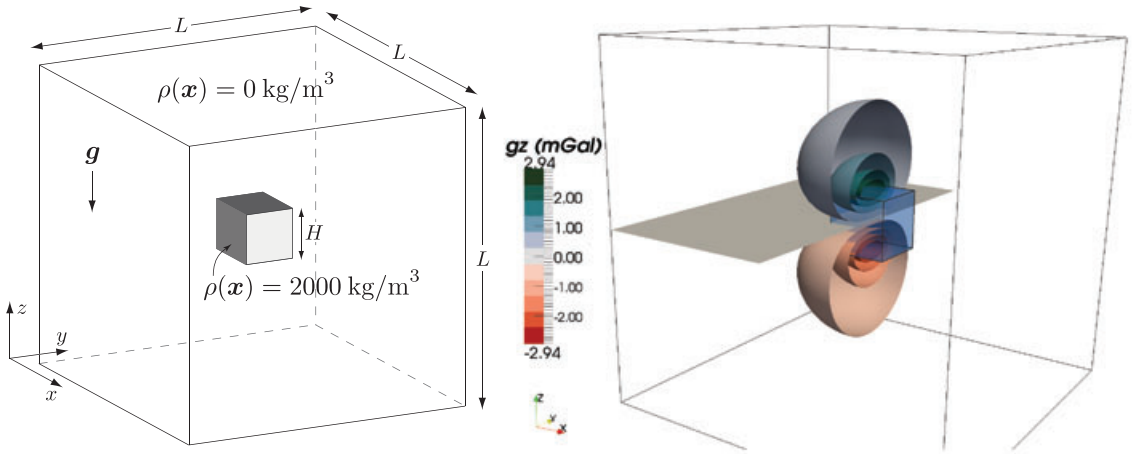
3.1 Discretization error (convergence)

The calculations for each numerical method used a mesh comprised of hexahedral elements. The number of elements in each direction was chosen such that the density anomaly was exactly resolved by the hexahedral elements. Hence, the error we measure from each method does not include any error due to the discretization of the density field. We quantify the error in the vertical component of the gravity field g_z, using the L_1 norm

$$E_1 = \int_\Omega |g_z(\mathbf{x}) - g_z^h(\mathbf{x})|\, \mathrm{d}V, \tag{15}$$

the L_2 norm

$$E_2 = \left[\int_\Omega |g_z(\mathbf{x}) - g_z^h(\mathbf{x})|^2\, \mathrm{d}V \right]^{1/2} \tag{16}$$



(a) Geometry (b) Gravity field

Figure 4. Synthetic model used thorough out the numerical experiments. (a) Domain and density anomaly and (b) the corresponding analytic gravity field g_z (mGal). The inclusion is indicated by the transparent blue cube. See text for dimensions of the domain and density anomaly.

and the L_∞ norm

$$E_\infty = \max_{\mathbf{x}\in\Omega} \left| g_z(\mathbf{x}) - g_z^h(\mathbf{x}) \right| . \tag{17}$$

Here, g_z is the exact gravity computed via the analytic solution from Li & Chouteau (1998), g_z^h is the approximate gravity field computed using one of three numerical methods (summation, FE and FMM) and Ω is the model domain.

3.1.1 Summation

We computed the gravity component g_z with SUM-G1 and SUM-G2 using a number of meshes composed of $\bar{M}$ elements in each x, y, z direction. The following grid sequence was used to measure the convergence rate, $\bar{M} = \{12, 24, 48, 96\}$. The side length of each element is given by $h = 600/\bar{M}$, hence for the mesh sequence used we have $h = \{50, 25, 12.5, 6.25\}$ m. Given that SUM-AN employs an analytic solution for the gravity at a point due to hexahedral-shaped density anomaly, the error expected is of machine precision. Hence, we omit this method from the discussion of errors. The error in eqs (15), (16) and (17) was approximated via a one-point quadrature rule over each hexahedral element in the mesh. The error E_1 as a function of grid resolution is shown in Fig. 5. The convergence rate of gravity field in the discrete error measures E_1, E_2, E_∞ is shown in Table 1.

3.1.2 FE method

The convergence behaviour of the FE methods FEM-D and FEM-GT was computed using the same grid sequence as in the summation test. Again, the mesh consisted of undeformed elements with $\Delta x = \Delta y = \Delta z = h$. A high-order Gauss quadrature scheme was used to evaluate the error measures E_1, E_2 and E_∞. Details of how the error for the FE approaches was computed is provided in Appendix. The L_2 discretization

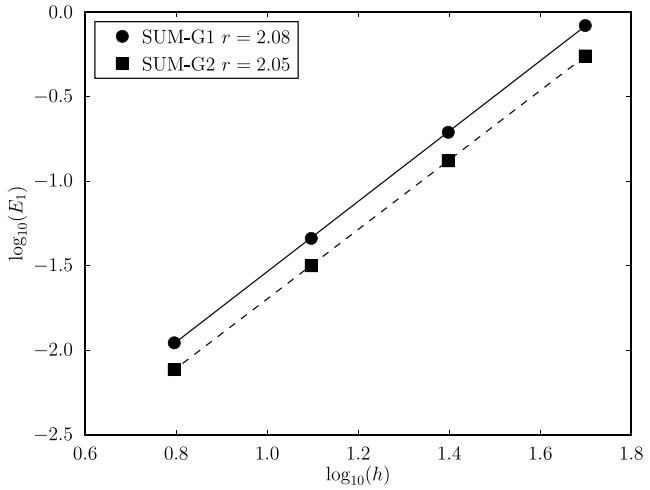


Figure 5. Convergence rate of the L_1 norm for the gravity field computed using SUM-G1 and SUM-G2.

Table 1. Convergence rates obtained with the summation methods.

error	SUM-G1	SUM-G2
E_1	2.08	2.05
E_2	1.53	1.52
E_∞	0.99	0.99

error as a function of grid resolution h is shown in Fig. 6. The convergence rate of the gravity field in the discrete E_1, E_2 and E_∞ norms is shown in Table 2. From these results, it immediately obvious that using the Robin boundary condition not only produces smaller errors, but the FEM-GT method yields much higher convergence rates.

To investigate sensitivity of the two boundary conditions used in the FE approaches to the size of the model domain, we performed another convergence test and varied the aspect ratio L/H, where model domain and anomaly length are denoted by L and H, respectively. The anomaly size H was kept fixed at 600 m, while L was increased such that we had the following aspect ratios $L/H = \{3, 12, 18\}$. As in the other convergence tests, four meshes of increasing resolution were used. To keep the discretization errors comparable between the different models, we ensured that element size on each of the four meshes, for each L/H yielded element sizes of $h = \{50, 25, 12.5, 6.25\}$ m. The L_2 convergence rates are shown in Fig. 7. Here, we see that the convergence rate of FEM-GT is independent of the domain size, while the convergence rate of the gravity field computed using FEM-D increases as the model domain increases. We expect that the rate from FEM-D approaches 1.0 as $L/H \to \infty$.

3.1.3 FMM

The convergence rate of PetFMM was performed using the same mesh sequence as in the summation experiments. As for the summation methods, the error measures were approximated via a one-point quadrature rule over each hexahedral element. The accuracy of the solution obtained via PetFMM is strongly related to the number of terms p used in the expansion of eq. (14). The measured convergence rate in the different norms are presented for $p = \{1, 4, 8, 20\}$ in Table 3. For the error measure E_1, we show the variation with grid resolution h in Fig. 8. Comparing with the rates from the summation methods from Table 1, we note that as p increases, the convergence rates of PetFMM approach those obtained using SUM-G1 and SUM-G2.

3.2 Optimality (CPU time)

Here, we report the CPU time of the different numerical methods applied to the synthetic model described in Section 3. All timings reported were obtained with code compiled using GCC 4.4.3 with level three optimization and with an optimized build of the PETSc library.

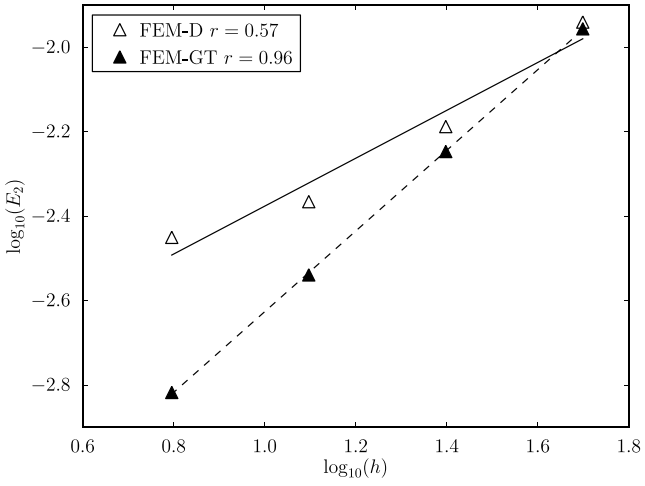


Figure 6. L_2 error of the gravity field computed via FEM-D and FEM-GT.

Table 2. Convergence rates of the finite-element methods for $L/H = 6$.

error	FEM-D	FEM-GT
E_1	0.23	0.68
E_2	0.57	0.96
E_∞	0.97	0.97

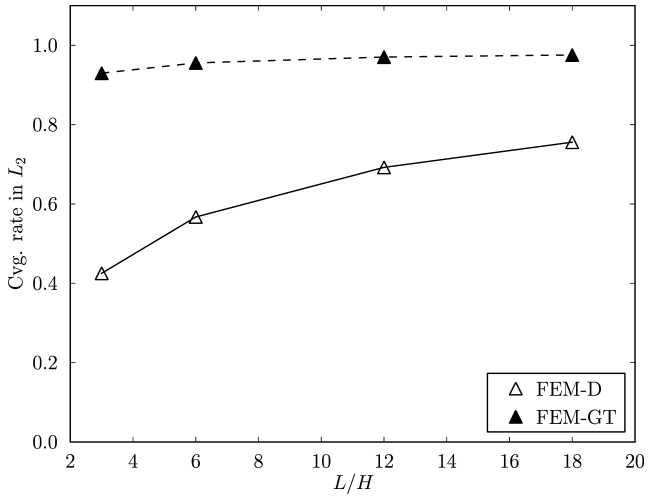


Figure 7. Convergence rate in L_2 as a function of the domain size.

Table 3. Convergence rates of PetFMM using different values of p.

error	p			
	1	4	8	20
E_1	−2.58	−0.66	1.45	2.08
E_2	−1.81	0.08	1.51	1.53
E_∞	−1.76	−0.04	0.99	0.99

The timing runs were performed on Octopus, which is an eight-core Intel Xeon 2.67 GHz (Nehalem) machine possessing 64 GBytes of RAM.

3.2.1 Summation

On a given mesh, the time required for the summation methods is proportional to the number of locations where the gravity is evaluated. For this series of tests, we evaluated the gravity on a regularly spaced array of 150 × 150 points, located at the upper surface of the model domain. In Table 4, we report the total CPU time (seconds) per gravity station on the following sequence of meshes, $\bar{M} = \{6, 12, 24, 48, 96\}$. Methods SUM-G1 and SUM-G2 compute the three components of the gravity vector, while SUM-AN and SUM-G1(z) only compute the gravity field in the z direction. All methods possess an approximately linear relationship between the CPU time/station and the number of cells used to discretize the domain. Considering the one-point quadrature rule methods, SUM-G1 is only a factor of 1.5 slower than SUM-G1(z). The slight increase in time required for SUM-G1 is a consequence of a more general quadrature. In this implementation, arbitrarily deformed hexahedral elements are permitted, while element edges were required to be perpendicular to the coordinate system in SUM-G1(z). Allowing the elements to be deformed requires that the integration be performed in a reference coordinate system, which thus requires the inverse Jacobian (coordinate transformation) to be evaluated. SUM-G2 was observed to be approximately seven times slower than SUM-G1, even though it employs eight times as many quadrature points. The closed-form method, SUM-AN is ∼150 times slower than SUM-G1(z) and ∼13 times slower than SUM-G2.

3.2.2 FE method

The FE calculations were performed using meshes consisting of $\bar{M} = \{12, 24, 48, 96, 192, 384\}$ elements in each direction. In all the calculations performed, the multigrid pre-conditioner used a coarse grid consisting of 6 × 6 × 6 elements. The number of grid levels n_l, was chosen to give the desired value of $\bar{M}$ on the finest grid level.

In Table 5, we report the time required to perform the linear solve of the system in eq. (9) using both FEM-D and FEM-GT. The time required for the solve represented more than 99 per cent of the total execution time, thus only the solve time is reported. We observe that the number of iterations required by both methods is independent of the grid resolution. Furthermore, both the CPU time and memory usage scale approximately linearly with respect to the number of unknowns in the potential field, $n = (\bar{M} + 1)^3$. The solve time for the FEM-D method is slightly higher than that required by FEM-GT. The difference in CPU time is attributed to the manner in which the Dirichlet boundary conditions were imposed during each application of the matrix-free product, **Ay**. This particular operation could easily be further optimized in the future.

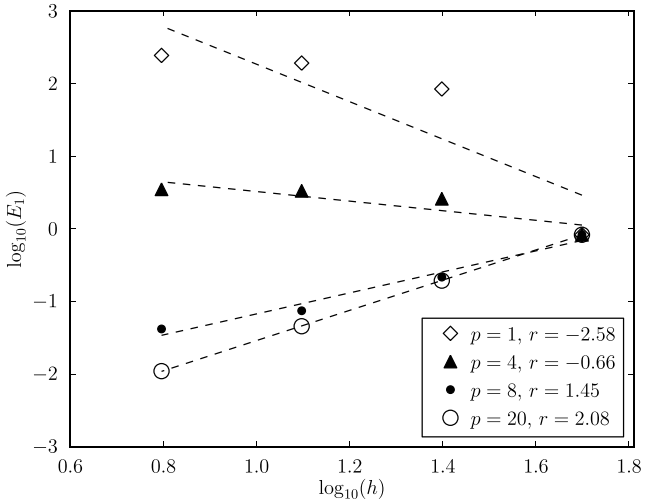


Figure 8. Convergence rate of the L_1 norm of the gravity field computed using PetFMM using different values of p.

Table 4. CPU time (sec) for the summation methods. The times reported are normalized by the number of locations where the gravity field was evaluated. Here, $\bar{M}$ is the number of cells used to discretize the subsurface in each direction and h is side length (m) of each cell.

		CPU time (sec)/station			
h (m)	$\bar{M}$	SUM-G1(z)	SUM-G1	SUM-G2	SUM-AN
100	6	4.78e−07	7.36e−07	5.34e−06	7.39e−05
50	12	3.71e−06	5.72e−06	4.29e−05	5.50e−04
25	24	2.98e−05	4.56e−05	3.40e−04	4.39e−03
12.5	48	2.37e−04	3.65e−04	2.68e−03	3.76e−02
6.25	96	1.90e−03	2.92e−03	2.13e−02	2.83e−01

3.2.3 FMM

The performance of the PetFMM algorithm was measured using the same sequence of meshes as used in the FE approaches, that is, the mesh contained $\bar{M} = \{12, 24, 48, 96, 192, 384\}$ elements in each direction. The octree used to define the FMM data structure used $k = 2^L$ cells along each axis, where L denotes the number of levels within the tree. For the mesh sequence used, we employed $L = \{2, 3, 4, 5, 6, 7\}$. In these calculations presented, the gravity vector was computed at the centroid of each cell used to discretize the density field. In Table 6, we report the CPU time (seconds) required to execute the PetFMM algorithm. The time required to evaluate the gravity field is negligible compared to time spend in the PetFMM algorithm and is thus not reported here. For these experiments, the graph partitioner ParMETIS was used.

For the sequence of meshes used in our test, an optimal FMM algorithm may be expected to yield execution times and memory usage requirements which increased by a factor of 8, for each increase in grid resolution. From Table 6, the memory usage is observed to follow this scaling. However, we note that the CPU time for PetFMM is observed to only approach the anticipated result as $\bar{M}$ increases. In Fig. 9, the solution time (solid thin line, left y-axis) and the solution time ratio for t_k/t_{k-1} (solid thick line, right y-axis), is plotted as a function of the number of elements in each direction $\bar{M}$. The anticipated optimal value of $t_k/t_{k-1} = 8$ is denoted via the thin grey line.

We can explain the deviation of this ratio observed with small numbers of voxels to a surface to volume effect. For a cube, divided into k pieces along each axis, we obviously have k^3 small constituent cubes. Of these, eight are corner cubes which have seven neighbours. There are 12 edges of the large cube, each of which has $k - 2$ small cubes with 11 neighbours. Similarly, there are six faces of the large cube, each of which has $(k - 2)^2$ small cubes with 17 neighbours. The remaining $(k - 2)^3$ interior cubes have 26 neighbours. We can check that the number of small cubes is correct.

$$(k-2)^3 + 6(k-2)^2 + 12(k-2) + 8 \tag{18}$$

$$= (k^3 - 6k^2 + 12k - 8) + 6(k^2 - 4k + 4) + 12(k-2) + 8 \tag{19}$$

$$= k^3. \tag{20}$$

If we assume that B particles are in every cube, then the direct work done per cube is given by

$$W_c = \frac{B(B-1)}{2} + N_B B^2 \approx \left(N_B + \frac{1}{2}\right) B^2, \tag{21}$$

Table 5. Performance of the FE methods. The CPU time (sec) and the number of iterations required by the Poisson solver are reported. The memory usage (MB) for FEM-D and FEM-GT are the same and are reported in the final column.

		FEM-D		FEM-GT		
h (m)	$\bar{M}$	CPU time (sec)	Iter.	CPU time (sec)	Iter.	Mem. (MB)
50	12	1.17e−02	8	1.07e−01	8	<10
25	24	1.10e+00	9	1.01e+00	9	<10
12.5	48	8.87e+00	9	8.08e+00	9	4.00e+01
6.25	96	7.11e+01	9	6.45e+01	9	2.85e+02
3.13	192	5.67e+02	9	4.68e+02	8	2.20e+03
1.56	384	4.15e+03	8	3.74e+03	8	1.70e+04

Table 6. CPU time (sec) and memory usage (MB) for PetFMM with increasing grid resolution. In these calculations, we used an expansion order of $p = 8$. We note that the memory counter used in the implementation of PetFMM was not able to represent the number of bytes required for the case $\bar{M} = 384$.

h (m)	$\bar{M}$	L	CPU time (sec)	Mem. (MB)
50	12	2	8.02e−02	< 1.00e+00
25	24	3	1.19e+00	1.67e+00
12.5	48	4	1.34e+01	1.34e+01
6.25	96	5	1.27e+02	1.07e+02
3.13	192	6	1.11e+03	8.56e+02
1.56	384	7	9.33e+03	(counter overflow)

where N_B is the number of cube neighbours. The ratio of work R, between a $2k$ division compared to a k division along each axis is given by

$$R\left(\frac{2k}{k}\right) = \frac{(2k-2)^3\frac{53}{2} + 6(2k-2)^2\frac{35}{2} + 12(2k-2)\frac{23}{2} + 8\frac{15}{2}}{(k-2)^3\frac{53}{2} + 6(k-2)^2\frac{35}{2} + 12(k-2)\frac{23}{2} + 8\frac{15}{2}} \tag{22}$$

$$= \frac{53(2k-2)^3 + 210(2k-2)^2 + 276(2k-2) + 120}{53(k-2)^3 + 210(k-2)^2 + 276(k-2) + 120} \tag{23}$$

$$= \frac{(8k^3 - 24k^2 + 24k - 8) + 3.96(4k^2 - 8k + 4) + 5.21(2k-2) + 2.26}{(k^3 - 6k^2 + 12k - 8) + 3.96(k^2 - 4k + 4) + 5.21(k-2) + 2.26} \tag{24}$$

$$= \frac{8k^3 - 8.16k^2 + 2.74k - 0.32}{k^3 - 2.04k^2 + 1.37k - 0.32}. \tag{25}$$

In the first two tests considered in Table 6, we have $k = 2$ and 4, thus

$$R\left(\frac{4}{2}\right) = \frac{968.00}{60.00} = 16.13. \tag{26}$$

Even at $k = 8$, we have

$$R\left(\frac{16}{8}\right) = \frac{95288.00}{10392.00} = 9.17 \tag{27}$$

and we can see that not inconsiderable surface-to-volume effects persist for larger octrees. The optimal ratio defined by eq. (25) is denoted in Fig. 9 via the dashed line. The agreement between the optimal and measure work scaling illustrated in Fig. 9 verify the optimality of the M2L–transformation.

3.3 Parallel scalability

To measure the parallel performance of an algorithm, two types of studies are typically employed. The first measure considers 'weak scaling', in which a fixed number of unknowns per processor (i.e. the work per processor) is kept constant and more processors are introduced. Thus, the overall problem size increases with the number of processors, but the work per process remains constant. Ideal weak scaling would yield a solution time which was independent of the number of processors which were employed. Alternatively, 'strong scaling' considers a problem with a fixed number of unknowns that is solved using an increasing number of processors. Thus, the unknowns per processor decrease as the number of processors increases. Ideal strong scaling would yield a solution time, which linearly decreases in proportion to the number of processors used to solve the problem.

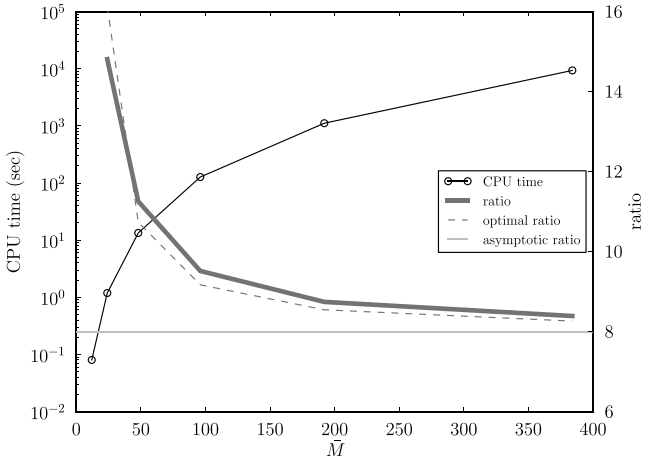


Figure 9. Computation time as a function of the number of density blobs $\bar{M}^3$ for PetFMM. The left y-axis denotes CPU time (sec) and right y-axis denotes the ratio of solution times between the current and previous grid resolution. For the grid sequence used, the asymptotic (linear) scaling would yield a ratio of 8, here denoted via the dashed line.

In the interest of developing fast algorithms for performing gravity inversions in a reduced amount of time, here we only consider the strong scalability of the three algorithms presented. If a simulation required t_0 s on p_1 processors, the optimal time t_{opt}, on $p_2 > p_1$ processors is $t_{\text{opt}} = t_0(p_1/p_2)$. The parallel efficiency E of the strong scaling is measured according to

$$E = 100\left(\frac{t_{\text{opt}}}{t_{\text{measured}}}\right), \tag{28}$$

where t_{measured} is the measured time taken for the computation on p_2 processors. All parallel results presented here were performed on the CADMOS IBM Blue Gene/P (http://bluegene.epfl.ch).

3.3.1 Summation

All of the summation algorithms considered here exploit parallelism by subdividing the set of voxels used to represent the density structure amongst n_p processors. The spatial decomposition of the mesh was defined by slicing the domain into N_x, N_y, N_z subdomains such that $n_p = N_x \times N_y \times N_z$. The only communication required in our implementation is the global reduction (sum) of a vector of length equal to the number of evaluation points. Thus, if the number of voxels in each processors subdomain is equal, the only departure from perfect strong scaling can be attributed to the single call to MPI_Allreduce. In Table 7, we report the CPU times obtained from using SUM-G1(z) with a model domain of 128^3 voxels and 100^2 evaluation points which were regularly spaced in a horizontal plane located at the upper surface of the model. Both the CPU time for the total computation and the time for the global reduction are reported. We note the time for the global reduction does not exhibit perfect strong scaling for this set of experiments. Accordingly, when the time required to perform the evaluation and local sum of the gravity contributions is much larger than the time required for the reduction, excellent scalability is observed ($n_p \leq 256$).

Table 7. Strong scaling for SUM-G1 on CADMOS BG/P, using a mesh with 128^3 cells and 100^2 evaluation points. Here, n_p indicates the number of processors used. (D) indicates the job was executed in DUAL mode, implying two processors per node were used. (V) indicates the job was launched in VN mode, in which all four processors per node were used.

	CPU time (s)	
n_p	Total	Reduction
1	1.0632e+03	1.6999e-04
8	1.3922e+02	1.2458e+01
64	1.8222e+01	3.1192e+00
128	9.3883e+00	2.0806e+00
256 (D)	4.8376e+00	1.3013e+00
512 (V)	2.4939e+00	7.8258e−01
512	2.4934e+00	7.8200e−01
2048 (V)	7.0128e-01	3.2787e−01

When this time is comparable with the cost of the reduction, the suboptimal scaling of the reduction will become significant and deteriorate the scaling of the total execution time. Comparing the total CPU times for $n_p = 8$ and 2048, we observe a parallel efficiency of $E \approx 78$ per cent.

3.3.2 FE method

The success of a parallel multigrid is largely dependent on the type of coarse grid solver used. We consider a direct extension of sequential multigrid algorithms, which employ a direct solver on the coarsest grid level. The direct solve on the coarse grid was performed in parallel using either the multifrontal method MUMPS (Amestoy *et al.* 2001), or by TFS (Tufo & Fischer 2001). MUMPS is a general-purpose parallel direct solver, while TFS is specifically designed for matrix problems in which a processors subdomain contains very few degrees of freedom (as is the case on our distributed coarse grid). TFS has the limitation that the number of processors must be a power of 2.

To examine the strong scalability, we considered two experiments in which the fine grid contained either 256^3 elements of 512^3 elements. The coarse grid was defined via $\bar{M}_c$ elements in each direction. Both experiments used six grid levels, with $\bar{M}_c$ being 8 and 16, respectively. In our GMG implementation, we require for a given grid, that each processor's local subdomain must contain at least one element. Accordingly, the number of elements in the coarse grid thus places an upper limit on the maximum number of CPU's we can use. The results of the strong scalability are shown in Fig. 10. The scalability of FEM-D and FEM-GT are expected to be identical so only the results of FEM-D are presented. The measured parallel efficiency on 512 CPUs was $E \approx 90$ per cent for the problem using MUMPS and $E \approx 68$ per cent on 2048 CPUs for the problem employing TFS.

3.3.3 FMM

To examine the strong scalability of PetFMM, we considered three different meshes with $\bar{M}^3$ elements where $\bar{M} = \{96, 192, 384\}$. For a given number of input density values, there is a number of levels L_v which minimizes the total computation time. As in the multigrid implementations, they are certain restrictions upon the number of CPUs (n_p) which can be used with PetFMM. The primary constraint is on the number of local trees in the spatial decomposition. The number of local trees N_t is given by $2^{d \times r_l}$, where $d = 3$ is the spatial dimension and r_l is the root level of the tree. For efficiency, it is required that $N_t > n_p$, so that at least one tree is distributed to every process.

For the parallel runs presented here, the simple geometric-based partitioning algorithm was used to balance load and communication. The total execution times are reported in Table 8.

The strong scaling efficiency is observed to decrease as the number of processors used increases and also as the root level increases. To better understand the reason for this scaling behaviour, we examined the scalability of individual components within the PetFMM implementation. The breakdown of CPU times for the $n_p = \{512 - 4096\}$ series of jobs is shown in Fig. 11. The downward sweep event involves both a parallel operation (indicated by 'DownSweep' in Fig. 11) and a sequential operation at the root level of the tree (indicated by 'Root Tree DownSweep' in Fig. 11). Thus, if the time required for the sequential operation is large compared to the time spent in evaluating contributions from the local parts of the tree, strong scalability will obviously suffer. The local calculations are all observed to strong scale well, however as the subdomains become smaller, the cost of the root tree will eventually dominate the overall execution time and reduce the parallel efficiency. In our experiments, the cost of the root tree evaluation grows by a factor of 8 each time r_l is increased by one. To offset the increasing cost of root level calculation, that is, to observe better strong scalability, one can easily introduce work on each subdomain by increasing $\bar{M}$.

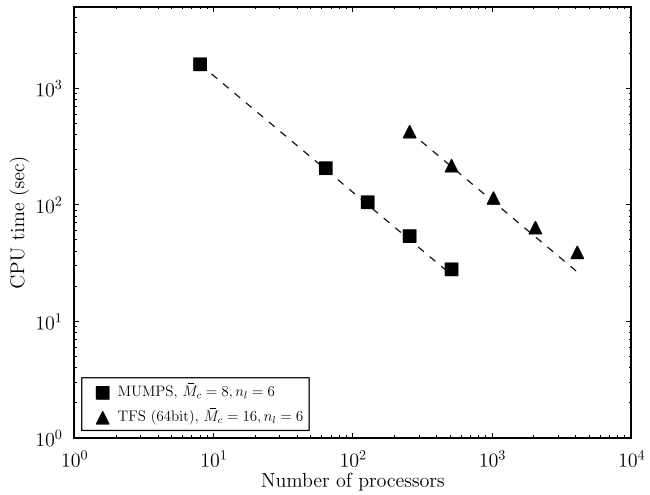


Figure 10. Strong scaling on the CADMOS BG/P for two different resolution FEM-D simulations. The optimal time is indicated by the dashed line. The coarse grid consisted of $\bar{M}_c$ elements in each direction and each model used $n_l = 6$ levels. Two different coarse grid solvers, MUMPS and TFS were employed (see text for further details).

Table 8. Strong scaling of PetFMM on CADMOS BG/P. The times reported here represent the total time taken to perform the multipole summation (ParaFMMEvaluate). (S) denotes mode SMP, (D) denotes mode DUAL and (V) denotes mode VN. $*$ indicates efficiency was computed w.r.t the 64 CPU execution time ($p_1 = 64$).

n_p	r_l	L_v	$\bar{M}$	CPU time (sec)	Efficiency
8	2	4	96	3.9740e+02 (S)	-
16				2.0950e+02 (S)	95 per cent
32				1.1086e+02 (S)	90 per cent
64				5.9088e+01 (D)	84 per cent
32	3	5	192	9.2118e+02 (S)	-,-
64				4.8627e+02 (D)	95 per cent, -
128				2.5809e+02 (S)	89 per cent, 94 per cent$*$
256				1.4380e+02 (D)	80 per cent, 85 per cent$*$
512				8.6693e+01 (V)	66 per cent, 70 per cent$*$
512	4	6	384	7.8231e+02 (V)	-
1024				5.5052e+02 (D)	71 per cent
2048				4.3421e+02 (D)	45 per cent
4096				3.7705e+02 (V)	26 per cent

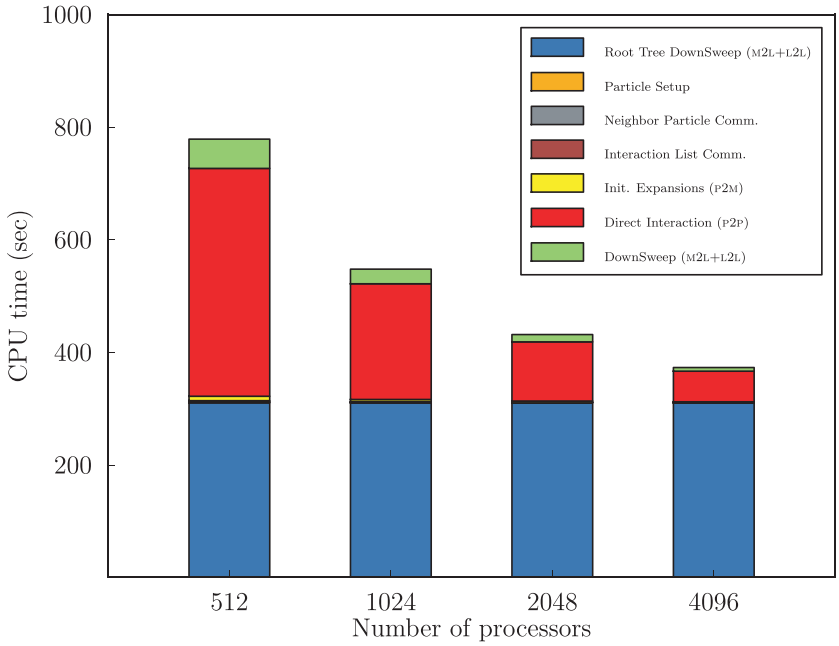


Figure 11. Breakdown of the strong scalability of the individual PetFMM components. Note that not all components listed in the legend are visible in the bar chart as they represent a very small fraction of the total execution time.

4 DISCUSSION

In the experiments described in Section 3.1, the discretization error of the three methods was examined. In the norms measured, the convergence rates obtained using SUM-G1 and SUM-G2 were nearly identical. A measurable difference in the absolute error between the different quadrature rules was observed, with SUM-G1 yielding errors approximately 2.3 times larger than SUM-G2. The rates measured between the two summation methods using Gauss quadrature and the rates obtained using PetFMM, were extremely similar, provided the expansion order p was high enough. In the cases where $p \leq 4$, suboptimal convergence (E_2), or divergence was observed (E_1, E_∞).

Both PetFMM and the summation methods incorporate the analytic solution of the potential (or gravity) within the discretization, thus these methods naturally satisfy the boundary condition, $\phi = 0$, as $\mathbf{x} \to \infty$. Within the FE methods considered here, this boundary condition was approximated. The convergence behaviour of the gravity field obtained using FE methods is thus likely to be dependent on the choice of approximation made. In the absence of any boundary condition approximation and any approximations in defining the density structure, we anticipate the gravity error computed with Q_1 elements to behave like

$$\|\mathbf{g} - \mathbf{g}^h\|_2 \leq c_1 h, \tag{29}$$

where c_1 is a constant independent of the grid resolution h. In the case of FEM-D, the boundary condition approximation is seen to limit how close the discrete solution will approximate the exact solution. Since the approximate boundary condition doesn't approach the true boundary

condition in the limit of $h \to 0$, the convergence of rate of the potential and gravity field will ultimately deteriorate with increasing grid resolution. That is we have

$$\|\mathbf{g} - \mathbf{g}^h\|_2 \leq c_1 h + c_2 \left(\frac{H}{L}\right). \tag{30}$$

This type of relationship is evident in Fig. 6 where we observe a low correlation between the straight line with slope 0.57 and the measured error. In practice, this effect can be reduced if we ensure that the model domain is significantly larger than the domain defining the density anomaly, thereby making the coefficient c_2 smaller. However, adopting this approach introduces significantly higher computational requirements.

However, the alternative boundary condition approximation used in FEM-GT does not appear to place a bound on the minimum discretization error possible on a finite-sized domain. This is apparent from Fig. 6 where a high correlation between the grid size and discretization error is observed. This suggests that the Robin boundary approximation converges like $\mathcal{O}(h)$ as the mesh is refined, since we observe the first-order convergence predicted from eq. (29) in the gravity field and this convergence rate appears to be independent of the domain aspect ratio L/H (see Fig. 7). Nevertheless, despite the improved convergence rate of FEM-GT, the rates observed are lower than those obtained using either the summation methods or PetFMM.

To assess the speed of the three methods examined, we consider defining the crossover point where the summation methods cease to be less efficient than either FEM-GT and PetFMM. The crossover point occurs when the number of evaluation points exceeds $t_{\text{FEM-GT,PetFMM}}/t_{\text{sum}}$, where t_{sum} is the time per evaluation point obtained from one of the summation algorithms. The number of evaluation points required to reach the crossover point for the sequential results are presented in Table 9. We note that the times from Table 4 are repeated in the second and third columns. The summation methods were not run at a grid resolution of $\bar{M} = \{192, 384\}$, therefore the time required for the summation algorithms was estimated from the time required by the $\bar{M} = 96$ case and scaling this value by 8 and 64, respectively.

All the three methods were observed to exhibit good strong scaling up to 1024 CPUs. By far, the easiest method to obtain good parallel scalability was the summation methods. This is simply due to the lack of algorithmic complexity in the direct summation approach. Scalability here is only limited by the network of the computer cluster used. Our tests were performed on an IBM Blue Gene/P, which is known to have an excellent network with specialized hardware for performing global reductions. The techniques used by FEM-D,FEM-GT and PetFMM are more difficult to obtain high strong scaling efficiency. In the context of the multigrid pre-conditioner, this was due to the design choice that the mesh on the coarse grid had to be distributed and that we required at least one element per CPU. This particular restriction could be relaxed if a different coarse grid solver was employed. For example, we could use a large coarse grid, use less levels in the pre-conditioner and employ an exact coarse grid solve using an algebraic multigrid (AMG) pre-conditioner. The AMG algorithms are useful in this context as they do not require any geometric information to determine how the work will be distributed across the CPUs. With PetFMM, speedup was measured up to 4096 CPUs, however the measure efficiency was only 26 per cent. Strong scaling with PetFMM is hindered by the sequential calculations, which have to be performed at the root level. This is a typical bottleneck in FMM algorithms, however it could be eliminated by overlapping the root tree computation with the local direct summation work. This will be the object of future research.

Lastly, we consider the overall usability of the different methods for computing gravity anomalies from the perspective of an end user. The quadrature-based methods are by far the easiest method to use. It permits complete geometric freedom in defining the underlying grid, which is used to discretize the density field. No connectivity is required between the cells and the vertices. The only requirement is that the cells used to partition the domain defined by the density anomaly do not overlap. Consequently, topography, curvature and locally refined regions are easily introduced. In the method described here, a constant density was used within each cell. This is not strictly necessary and spatial variations of density within a cell are possible, however the order of the quadrature rule used would likely have to be increased to maintain the accuracy of the method.

To use the GMG, a mesh hierarchy is required. Here, we considered nested hierarchies of structured meshes. With such a topology, generating a mesh that has element faces which conform to all the jumps in density may be difficult to construct. This could be partially alleviated by using an unstructured mesh, but fully unstructured meshing in parallel is still a challenging task. Furthermore, an unstructured mesh hierarchy would also be required to be generated. The convergence, and hence the CPU time required by the GMG method is strongly dependent on the mesh geometry. For example, the implementation described here ceases to be robust if the mesh possesses a high aspect ratio, or the elements are highly deformed. In such circumstances, stronger smoothers are required if rapid convergence is to be maintained.

Table 9. Crossover point between the summation algorithms and the PDE-based approaches. The rows marked with the * indicate that the summation times were estimated from the summation simulation with $\bar{M} = 96$. Columns 4–7 indicate the number of evaluation points below which the summation algorithms are faster than FEM-GT and PetFMM.

$\bar{M}$	CPU time (s) / station		FEM-GT		PetFMM	
	SUM-G1(z)	SUM-AN	SUM-G1(z)	SUM-AN	SUM-G1(z)	SUM-AN
12	3.71e−06	5.50e−04	2.88e+04	1.95e+02	2.16e+04	1.46e+02
24	2.98e−05	4.39e−03	3.39e+04	2.30e+02	3.99e+04	2.71e+02
48	2.37e−04	3.76e−02	3.41e+04	2.15e+02	5.65e+04	3.56e+02
96	1.90e−03	2.83e−01	3.39e+04	2.28e+02	6.68e+04	4.49e+02
192*	1.52e−02	2.26e+00	3.08e+04	2.07e+02	7.30e+04	4.90e+02
384*	1.22e−01	1.81e+01	3.08e+04	2.06e+02	7.67e+04	5.15e+02

Stronger smoothers may, for example, include block Jacobi with ILU factorization defined on the subblocks. Such choices mandate additional storage and careful selection and tuning of smoothers to remain optimal. To some extent, many of the aforementioned disadvantages related to geometric restriction introduced by using GMG can be overcome using AMG. AMG pre-conditioners require the stiffness matrix to be assembled and furthermore, maintaining both scalable and optimal solution times in parallel is still a challenge with these approaches. The FE approaches do have the advantage that continuous density variations can be naturally introduce throughout the element.

The FMM does not possess any geometric restrictions in how the density structure may be defined. While structured grids were used here, FMM can in principal be used with a random point distribution, which defines the location and value of density in space. In the case when a random distribution of points is used, one also needs to provide the volume of the domain that is associated to each point. This can be readily computed using a Voronoi diagram, or preferentially in parallel calculations using an approximate Voronoi diagram. Thus, the method provides high geometric fidelity without having the burden of creating a mesh, conforming or otherwise. The time required to compute the gravity signal is a function of the number of points used to discretize the density, and not dependent on their spatial distribution. The convergence of FMM could be improved by introducing a basis with more smoothness than the current delta function discretization. In future work, we will introduce a Gaussian basis for rock masses so that the convergence rate can be adjusted by varying the width of the Gaussians. The initial interpolation problem for this new basis will be solved using the PetRBF code (Yokota *et al.* 2010).

Despite being more than two times slower than FEM-GT, we believe that the geometric flexibility permitted in defining the density structure, combined with fact that the solve time is independent of the geometry of the discretization used for the density structure, make PetFMM more useful in applied geophysics studies. PetFMM was shown to be comparable in accuracy to the SUM-G2 algorithm and should be used preferentially over this method if the number of evaluation points exceeds 77×10^4.

5 CONCLUSION

Fast and robust forward models for computing a gravity signal from a density distribution is essential to perform high-resolution inversions of the density subsurface. Here, we have discussed three different forward modes for computing gravity anomalies and compared them based on the convergence rates of the obtained gravity field, the execution time required to evaluate the gravity field and the parallel scalability of the algorithms. We considered classical summation techniques based on closed-form expressions or quadrature schemes, and optimal and scalable approaches suitable for solving the Poisson equation. The PDE-based approaches consisted of an FE discretization using a GMG pre-conditioner and an implementation of the FMM.

The summation methods employing quadrature approximations are found to yield results of comparable accuracy to FMM. Only the FE method that incorporated a far-field gravitational approximation in the form of a Robin boundary condition was deemed to be useful in practice. The error incurred by specifying a vanishing potential on the boundary of a finite domain resulted in large errors, and low convergence rates in the gravity field. All the forward models demonstrated good strong scaling up to 1024 CPUs. The FMM presents itself as a viable alternative to classical summation methods due to the geometric freedom in defining the density structure and insensitivity of the overall CPU time to the underlying density structure. In comparison to the summation algorithm employing analytic expression for the gravity, FMM is faster provided more than 515 evaluation points are used. If the simplest quadrature-based summation algorithm is used, FMM will provide a faster forward model if more than 77×10^4 evaluation points are used.

ACKNOWLEDGMENTS

The authors wish to thank Laetitia Le Pourhiet for computer time on *Octopus*. All parallel computations were performed on the CADMOS BG/P, for which the authors thank Yuri Podladchikov. DAM was supported by the ETH Zürich Postdoctoral Fellowship Program. Partial support was provided by the European Research Council under the European Community's Seventh Framework Program (FP7/2007-2013) / ERC Grant agreement #258830. MGK acknowledges partial support from NSF grant EAR-0949446.

REFERENCES

Amestoy, P., Duff, I., Koster, J. & L'Excellent, J.-Y., 2001. A fully asynchronous multifrontal solver using distributed dynamic scheduling, *SIAM J. Matrix Anal. Appl.,* **23**(1), 15–41.

Asgharzadeh, M.F., von Frese, R.R.B., Kim, H.R., Leftwich, T.E. & Kim, J.W., 2007. Spherical prism gravity effects by Gauss-Legendre quadrature integration, *Geophys. J. Int.,* **169,** 1–11.

Balay, S. *et al*., 2010. PETSc users manual, Tech. Rep. ANL-95/11—Revision 3.1, Argonne National Laboratory.

Balay, S. *et al*., 2011. PETSc web page, http://www.mcs.anl.gov/petsc (last accessed 2011 February 21).

Boroomand, B. & Zienkiewicz, O.C., 1997. Recovery by Equilibrium in Patches (REP), *Int. J. Numer. Methods Eng.,* **40,** 137–164.

Briggs, W., Henson, V. & McCormick, S., 2000. *A Multigrid Tutorial,* 2nd edn, Society for Industrial and Applied Mathematics, Philadelphia, PA.

Cai, Y. & Wang, C., 2005. Fast finite-element calculation of gravity anomaly in complex geological regions, *Geophys. J. Int.,* **162,** 696–708.

Cruz, F.A., Knepley, M.G. & Barba, L.A., 2010. PetFMM—a dynamically load-balancing parallel fast multipole library, *Int. J. Numer. Methods Eng.,* **85**(4), 403–428, doi:10.1002/nme.2972.

Farquharson, C. & Mosher, C., 2009. Three-dimensional modelling of gravity data using finite differences, *J. appl. Geophys.,* **68,** 417–422.

Fog, A., 2011. Software optimization resources, http://www.agner.org/optimize (last accessed 2011 March 31).

Hughes, T.J., 1987. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis,* Prentice-Hall, Englewood Cliffs, NJ.

Johnson, L. & Lithehiser, J.J., 1972. A method for computing the gravitational attraction of three-dimensional bodies in a spherical or ellipsoidal Earth, *J. geophys. Res.,* **83,** 6999–7009.

Karypis, G., 2011. ParMETIS web page, http://glaros.dtc.umn.edu/gkhome (last accessed 2011 February 21).

Karypis, G. & Kumar, V., 1998. A parallel algorithm for multilevel graph partitioning and sparse matrix ordering, *J. Parallel Distrib. Comput.,* **48,** 71–85.

Knepley, M.G. & Karpeev, D.A., 2009. Mesh algorithms for PDE with Sieve I: mesh distribution, *Sci. Program.,* **17**(3), 215–230.

Li, J. & Widlund, O., 2007. On the use of inexact subdomain solves for BDDC algorithms, *Comput. Methods appl. Mech. Eng.,* **196,** 1415–1428.

Li, X. & Chouteau, M., 1998. Three-dimensional gravity modeling in all space, *Surv. Geophys.,* **19,** 339–368.

Saad, Y., 2003. *Iterative Methods for Sparse Linear Systems,* 2nd edn, Society for Industrial and Applied Mathematics, Philadelphia, PA.

Smith, D., Robertson, D. & Milbert, D., 2001. Gravitational attraction of local crustal masses in spherical coordinates, *J. Geod.,* **74,** 783–795.

Trottenbert, U., Oosterlee, C.W. & Schuller, A., 2001. *Multigrid,* Academic Press, London.

Tufo, H. & Fischer, P., 2001. Fast parallel direct solvers for coarse grid problems, *J. Parallel Distrib. Comput.,* **61**(2), 151–177.

Wesseling, P., 1992. *An Introduction to Multigrid Methods,* Wiley, New York, NY.

Yokota, R., Barba, L.A. & Knepley, M.G., 2010. PetRBF—a parallel $\mathcal{O}(N)$ algorithm for radial basis function interpolation, *Comput. Methods Appl. Mech. Eng.,* **199**(25–28), 1793–1804, doi:10.1016/j.cma.2010.02.008.

Zienkiewicz, O.C. & Zhu, J.-Z., 1992. The superconvergent patch recovery (SPR) and adaptive finite element refinement, *Comput. Methods Appl. Mech. Eng.,* **101,** 207–224.

APPENDIX A: ERROR EVALUATION

Here, we discuss the method used to evaluate errors defined in eqs (15), (16) and (17). The spatial variation of the discrete solution for the gravity field g_z^h is defined by the representation natural to discretization. For the summation and FMM, this means g_z^h is represented via piecewise constant over each cell. For the FE methods, g_z^h is represented via a bilinear function $g_z^h = a_0 + a_1 x + a_2 y + a_3 xy$, since the potential ϕ was discretized via trilinear basis functions. The integrals in eqs (15) and (16) were approximate via Gauss quadrature. The order of the quadrature used was determined empirically. The complexity of the analytic solution was such that low-order rules were not appropriate to accurately estimate the norm. Over each cell in the discretization, we found that a four-point quadrature rule, applied over $m \times m \times m$ subdivision (in each x, y, z direction, respectively) of each cell was sufficiently accurate. The value for m was obtained by evaluating $\|g_z\|_1$, $\|g_z\|_2$, E_∞ and examining how the error norm varied with m. The results from the experiment used to determine the value of m for each $\bar{M}$ are presented in Table A1. The final value of m shown for each $\bar{M}$ was used to calculate the norms in our experiments.

The same quadrature rule used to evaluate E_1, E_2 was used to evaluate E_∞.

Table A1. Results of the gravity quadrature test. Estimated values of the integral of the analytic gravity field obtained using a four-point Gauss quadrature scheme, with different numbers of integration regions m^3, within each cell.

$\bar{M}$	m	$\|g_z\|_1$	$\|g_z\|_2$	E_∞
12	3	2.686359587701e+02	3.461398542186e−02	3.381867068310e−05
	4	2.686359587702e+02	3.461399254307e−02	3.403021478492e−05
	5	2.686359587700e+02	3.461399453156e−02	3.415713959000e−05
	6	2.686359587701e+02	3.461399525775e−02	3.424175549691e−05
	7	2.686359587703e+02	3.461399557324e−02	3.430219515320e−05
	8	2.686359587699e+02	3.461399572806e−02	3.434752475888e−05
24	2	2.686359587702e+02	3.461399254307e−02	3.403021478492e−05
	3	2.686359587702e+02	3.461399525775e−02	3.424175549691e−05
	4	2.686359587698e+02	3.461399572806e−02	3.434752475888e−05
48	1	2.686359587702e+02	3.461399254307e−02	3.403021478492e−05
	2	2.686359587698e+02	3.461399572806e−02	3.434752475888e−05
96	1	2.686359587698e+02	3.461399572806e−02	3.434752475888e−05