AAS 07-223

American Astronautical Society

American Institute of Aeronautics and Astronautics

EFFICIENT GRAVITY FIELD COMPUTATION FOR TRAJECTORY PROPAGATION NEAR SMALL BODIES

Andrew Colombi, Anil N. Hirani, Benjamin F. Villac

17th AAS/AIAA Space Flight Mechanics Meeting

Sedona, Arizona January 28 – February 1, 2007

AAS Publications Office, P.O. Box 28130, San Diego, CA 92198

AAS 07-223

EFFICIENT GRAVITY FIELD COMPUTATION FOR TRAJECTORY PROPAGATION NEAR SMALL BODIES

Andrew Colombi[*], Anil N. Hirani[†], Benjamin F. Villac[‡]

We present an approximation scheme for gravitational force near arbitrarily shaped small bodies. The approximation uses polynomial interpolation with an adaptive spatial data structure. This data structure allows us to drive errors in the model to within user defined thresholds. The result is a method that is expected to reduce the run time of trajectory integrations around small bodies without compromising accuracy. Our method is likely to provide a substantial improvement in the speed of Monte Carlo analysis of spacecraft motion in proximity to small bodies.

INTRODUCTION

Gravitational force computation is the bottleneck of Monte Carlo simulation used in design and analysis of small body missions. This expense has lead researchers to investigate ways to reduce cost.[1] The current approach for this force evaluation, developed by Werner and Scheeres,[2,3] requires a summation of contributions from every face and edge over a polyhedral model representing the surface of the attracting body. While tractable over a single evaluation, this process becomes prohibitive when one considers the fact that numerically integrated trajectories consist of hundreds of elementary time steps, each of which rely on multiple force evaluations. Furthermore, Monte Carlo simulations represent another summation over this already expensive process. While the improvements proposed in Ref. 1 result in significantly faster running times, ultimately their complexity remains the same $O(F + E)$, where F is the number of faces and E is the number of edges in the model of the attracting body.

We propose to alleviate this computational burden by pre-computing gravitation throughout the domain, and reducing force computations to a fast, sub-linear (i.e. $O(\log N)$), interpolation operation. Indeed, given the availability of cheap, fast memory storage, trading memory against on-line computation seems advantageous. The idea of representing the gravitational potential using interpolation schemes has already been investigated by Junkins and Engels (Ref. 4 and 5) for the case of nearly spherical bodies. The extension of such methods to the case of arbitrarily shaped bodies presents several challenges: the irregularity of the domain and the taxing error bounds required for trajectory integration. This paper presents a solution to these challenges.

In the next section we review two computational models relevant to the formulation of our method. Then, the section 'Approximating an Asteroid's Gravitational Force' presents our approach to the computation of gravitational forces. The final two sections give a discussion of our results and present some challenges that remain.

[*]Graduate Student, Department of Computer Science, University of Illinois at Urbana Champaign. 201 N. Goodwin Avenue, Urbana, IL 61801; Email: colombi@uiuc.edu ; Phone: 703-431-2717.

[†]Assistant Professor, Department of Computer Science, University of Illinois at Urbana-Champaign. 201 N. Goodwin Avenue, Urbana, IL 61801; Email: hirani@uiuc.edu ; Phone: 217-333-2727.

[‡]Assistant Professor, Department of Mechanical and Aerospace Engineering, University of California, Irvine. 4200 Engineering Gateway, Irvine, CA 92697-3975 ; Email: bvillac@uci.edu ; Phone: 949-824-8148.

PREVIOUS WORK

We start with a discussion of the polyhedral method developed by Werner and Scheeres,[2,3] which we take to be the true gravitational force for the remainder of this paper. Then we discuss the method proposed by Junkins[4] as it provides a foundation for the present work.

Polyhedral Method

Polyhedral methods calculate gravity by performing an exact integration over a polyhedral approximation to the asteroid. By clever application of Green's Theorem this method transforms a three dimensional integral over the mass into a two dimensional integral over the surface. This surface integral is then further manipulated to yield a summation over all faces and edges, where each term requires calculating a transcendental function. The time complexity of this algorithm is $O(E + F)$ where E and F are the number of edges and faces respectively. As a result, simulations with this method, especially with a large model, can be very costly.

Efforts to reduce this burden have been researched,[1] and experiments in Ref. 1 used three techniques to reduce computational cost of the polyhedral method: *(i)* shape coarsening, *(ii)* calculation caching, and *(iii)* Taylor approximations. They found that all three methods could have a significant impact on the running time of a simulation. Specifically, in ideal circumstances a calculation could be sped up 100 fold without compromising the model's accuracy; more general orbits still benefited from a 10 times speed up.[1]

Interpolation Based Method

Interpolation based methods, as developed in Ref. 4, 5, compute gravitational forces through approximating polynomials fit to the forces present in a particular, small region. Such methods proceed by first dividing the domain of interest into a regular grid. Then, empirical or simulated (e.g. polyhedral methods) force measurements are made at each vertex. With this information the force in a given grid cell can be reconstructed with polynomial interpolation. At this point it is worth noting that to reconstruct force using this method one should not interpolate potential and differentiate. While the differentiation of the potential may be appropriate, it leads to some difficulties in the general case.[6] In particular, a small error in the potential may not necessarily lead to an equally small error in its derivatives.

As noted in the introduction, several challenges face the methods of Ref. 4, 5 when applied to asteroid gravity; foremost, the shape of an asteroid does not accommodate a regular grid. Working around an irregular body with a regular grid requires a very fine division level to capture the asteroid/space interface. This has an unnecessary and impractical memory cost as such a fine grid is largely wasted far from the asteroid. Furthermore, the methods of Ref. 4, 5 cannot provide a variable level of approximating power: the grid has the same resolution everywhere. On the other hand, an asteroid's gravitation will be more difficult to represent the closer we come to the body (see the section "Remainder Formulas"). These incongruities make the methods of Ref. 4, 5 difficult to apply in the case of irregular bodies.

APPROXIMATING AN ASTEROID'S GRAVITATIONAL FORCE

The primary task of our software is to provide quick and accurate estimates of the accelerations experienced by a spacecraft orbiting a small, irregular body like an asteroid or comet. To accomplish this we separate the problem into two tasks: dividing the domain into subdomains (which we shall call *cells*), and approximating the force in a cell. As noted above, this approach is similar to the interpolation method[4,5] with the additional challenges of close-proximity representation and irregularity of the domain. Thus, we choose an adaptive solution; one able to conform to the shape of an arbitrary asteroid and offer finer grained approximation in regions closer to the mass.

Subdividing the domain into manageable cells is accomplished with an adaptive "octree" data structure. This data structure operates by splitting the domain into variable sized cuboids that each contain a local model of gravity's force. Once a local model is in place we can estimate its error relative to the polyhedral method; if the error is too large the octree divides the cuboid and the process is repeated, otherwise we let the cuboid stay. In building the octree data-structure, we conservatively choose a relative error threshold of 10^{-6}, which is an order of magnitude less than the error in accelaration that is acceptable in mission design as advised by Ref. 1. We choose the more conservative threshold during octree construction because error estimation during that stage is based on random sampling, as explained in subsection "The error in a cell" below.

To approximate the gravitational force in a cell several methods are available. For this study we have chosen variable order polynomial interpolation approximating each direction of the force separately. In this scenario the actual gravity is computed using the polyhedral method at interpolation points inside each leaf cuboid of the octree. Placement of the interpolation points follows the Gauss-Lobatto-Legendre scheme, which is known for its low Lebesgue constant.[7] Once the force has been computed at all nodes an interpolating polynomial is constructed. We use Lagrange polynomial basis for their superior conditioning and ease of computation.[7,8]

Octrees

In this section we will give practical details regarding the octree data structure. Octrees are more easily illustrated by their 2D analogue; hence, the following description is given for "quadtrees". Quadtrees are adaptive tree data structures for organizing localized data in a rectangular domain. Two operations characterize the function of a quadtree: `Subdivide`, and `Find`. `Subdivide` is a constant time operation that splits a rectangle into four quadrants by splitting each dimension in half. A quadtree is built by beginning with a single rectangle and subdividing recursively until the desired tree structure is created.[9] Figure 1 shows a quadtree as a tree and as a collection of rectangles.

`Find` recalls localized data associated with query points by recursively traversing the tree. At each level `Find` picks the child quadrant containing our query; this is a constant time operation as each cell has at most four children. After every level is traversed a leaf is reached, and the data it contains is returned. As balanced trees have at most $\log N$ levels the run time complexity of `Find` is $O(\log N)$, where N is the total number of cells.[9] Thus, subdividing our model to improve accuracy incurs only a sub-linear run time penalty. This compares favorably to the polyhedral method which requires a linear cost increase to improve accuracy.

To illustrate the practical benefit of a sub-linear run time consider the following example. For the sake of argument we shall examine moving from polyhedral models with 1,000 elements to 10,000

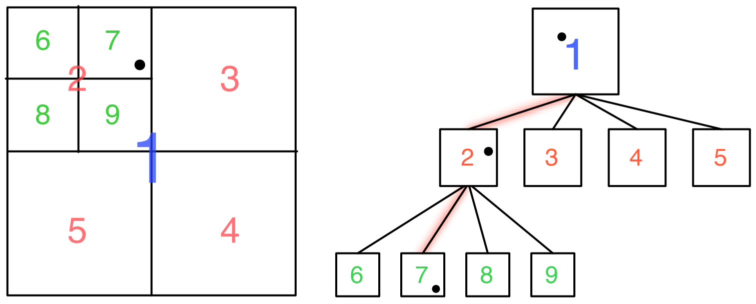

Figure 1 Left: A quadtree viewed geometrically. Right: A quadtree viewed as a tree. Labels show the mapping between geometric and tree views. The point represents a query; the tree view shows the query being resolved.

elements, and compare that to moving from 10,000 octree cells to 100,000. Following this scenario the polyhedral method would cost $10\times$ more computation when moving to the new model, whereas an octree method only costs $1.2\times$ the former computation. Taking this out another factor of 10 we find costs rising $100\times$ and $1.4\times$ respectively. In other words, methods with asymptotically better performance have dramatically superior run times as problems scale up to take advantage of newest computational power available.

It is worth mentioning that while quadtrees can be implemented in a straightforward manner, there are several clever tricks that are worth implementing. For example, a typical implementation associates a rectangle with each cell in the quadtree. Queries made to the quadtree recursively traverse each level, and perform a containment test on every rectangle the next level down. This implementation is functional, but substantial savings in time and memory can be made by taking advantage of the quadtree's special structure.[9] Following these techniques reveal a natural way to identify which leaf a query is in without traversing the tree. This is especially useful during trajectory simulations as it enables us to quickly check if the spacecraft has shifted cells.

Octrees follow the same design, but use cuboids in 3D instead of rectangles.[9]

Gauss-Lobatto-Legendre Interpolation with Lagrange Polynomials

Approximation of univariate functions with interpolating polynomials is a well studied field,[10] however, the theory of *multivariate* polynomial interpolation is still widely researched.[11,7] In this section we will provide a brief discussion of topics relevant to our model. For a deeper discussion see classic texts of the field such as Ref. 10.

Before continuing it will be helpful to define our notation. Let $N \subset \mathbb{R}^d$ be a set of points in $\mathbb{R}^d$ that accommodate interpolation, and define L_N as the operator that maps continuous functions to their polynomial interpolant; in other words, $L_N f(\mathbf{x}_i) = f(\mathbf{x}_i)$ for $\mathbf{x}_i \in N$. Define $D_\mathbf{v} f, \mathbf{v} \in \mathbb{R}^d$ as the directional derivative of f in the $\mathbf{v}$ direction, and $D^n_\mathbf{v} f$ as the directional derivative repeated n times. We will also use f^n to denote the n^{th} derivative of f when f is univariate. Define $p^N_{\mathbf{x}_i}$ to be the Lagrange polynomial centered at $\mathbf{x}_i \in N$; i.e. $p^N_{\mathbf{x}_i}(\mathbf{x}_j) = \delta_{ij}$ for $\mathbf{x}_i, \mathbf{x}_j \in N$.[11,7] Lastly, we use **boldface** to distinguish vector values from scalars.

Remainder Formulas

Remainder formulas represent the error in approximating a given continuous function f with an interpolating polynomial. The familiar remainder formula for univariate $f \in C[a,b]$ is,

$$f(x) - L_N f(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_n)}{(n+1)!} f^{n+1}(\xi), \tag{1}$$

where $x_i \in N$ are the interpolation points such that $a \leq x_0 \leq \cdots \leq x_i \leq \cdots \leq x_n \leq b$ and $\xi \in [a,b]$. Unfortunately, this result does not easily generalize to the multivariate case, for which there are several forms.

For our purposes we will discuss Ciarlet's formula,[12, 11]

$$f(\mathbf{x}) - L_N f(\mathbf{x}) = \sum_{\mathbf{x}_i \in N} p^N_{\mathbf{x}_i}(\mathbf{x}) \int_{[\mathbf{x}_i, \mathbf{x}, \ldots, \mathbf{x}]} D^{|N|}_{\mathbf{x}_i - \mathbf{x}} f, \tag{2}$$

where $|\cdot|$ denotes the cardinality of a set. (See Ref. 11 or 12 for an explanation of the $\int_{[\mathbf{x}_i, \mathbf{x}, \ldots, \mathbf{x}]}$ notation, which is beyond the scope of this paper. Suffice it to say that this notation integrates regions bounded by the subscripted points.) While such a bound is too costly to evaluate as a quantitative measure, we can still draw qualitative conclusions from it.

Let $f : \mathbb{R}^3 \to \mathbb{R}$ be the gravitational force in one direction. Taking derivatives of f always yields equations with r^{n-1} in the denominator, where n is the number of times we have differentiated. Hence, $D^{|N|}_{\mathbf{x}_i - \mathbf{x}} f$ takes on larger values the closer we are to the mass of the asteroid. By Eq. (2) we conclude that approximation by polynomial interpolation has larger error the closer we come to the asteroid.

Another lesson of Ciarlet's formula lies in the integral. By reducing the extent of N we reduce the region of the integral. This formalizes our intuition that octree subdivision reduces errors in a cell.

Lebesgue Constants and Gauss-Lobatto-Legendre Points

The Lebesgue constant is a measure of how close $L_N f$ is to the best uniform approximation of f.[7] One technique to minimize the Lebesgue constant is to find Feteke points. Feteke points reduce the Lebesgue constant by minimizing the associated Vandermonde matrix's determinant. It has been shown that Feteke points in one dimension are the Gauss-Lobatto-Legendre points, and this result scales to two and three dimensions via the tensor product, see Figure 2. Gauss-Lobatto-Legendre points for 1D, order n interpolation are the zeros of

$$\ell(x) := (x-1)(x+1)P_n'(x), \tag{3}$$

where $P_n(x)$ is the Legendre polynomial of order n. We shall refer to the zeros of Eq. (3) as $\hat{N}$.

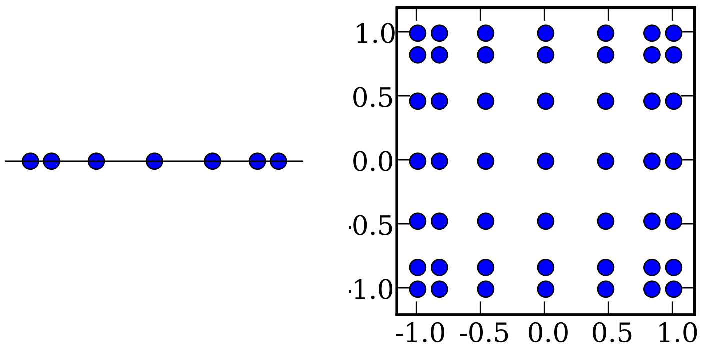

Figure 2 Left: One dimensional Gauss-Lobatto-Legendre nodes; we refer to these as $\hat{N}$. Right: Two dimensional tensor product of $\hat{N}$.

Lagrange Polynomials as a Polynomial Basis for Approximation

In the context of polynomial interpolation at points N, the $|N|$ Lagrange polynomials are defined as,

$$p^N_{\mathbf{x}_i}(\mathbf{x}_j) = \delta_{ij} \text{ for } \mathbf{x}_i, \mathbf{x}_j \in N. \tag{4}$$

In choosing Lagrange polynomials as our basis we are picking $L_N f$ to be of the form $\sum_{\mathbf{x}_i \in N} \alpha_i p^N_{\mathbf{x}_i}$. Thus, we seek α_i such that

$$f(\mathbf{x}_i) = \sum_{\mathbf{x}_j \in N} \alpha_j p^N_{\mathbf{x}_j}(\mathbf{x}_i). \tag{5}$$

Substituting Eq. (4) into Eq. (5) yields

$$\alpha_i = f(\mathbf{x}_i). \tag{6}$$

This straightforward construction of an interpolating polynomial is one of the chief advantages of a Lagrange basis.

Having determined coefficients for a local force model we need the tools to evaluate it anywhere in a cell. Lagrange bases are frequently misrepresented as an inefficient interpolation basis, but with moderate care they can be as speedy as a Newton's basis.[8] The critical step is to write the Lagrange polynomial in barycentric form. First, note that because N is formed as a tensor product of one dimensional distributions $\hat{N}$, we satisfy (4) with[7]

$$p^N_{\mathbf{x}_i}(\mathbf{x}) = p^N_{(x_{i_1}, x_{i_2}, x_{i_3})}(x_1, x_2, x_3) = p^{\hat{N}}_{x_{i_1}}(x_1) p^{\hat{N}}_{x_{i_2}}(x_2) p^{\hat{N}}_{x_{i_3}}(x_3). \tag{7}$$

Now define $\ell : \mathbb{R} \to \mathbb{R}$ as the polynomial such that $\ell(x_i) = 0$ for $x_i \in \hat{N}$. For a univariate Lagrange polynomial the barycentric form is[8]

$$p^{\hat{N}}_{x_i}(x) = \frac{\ell(x)}{\ell'(x_i)(x - x_i)}, \tag{8}$$

and substituting this into Eq. 7 yields

$$p^N_{\mathbf{x}_i}(\mathbf{x}) = \frac{\ell(x_1)}{\ell'(x_{i_1})(x - x_{i_1})} \frac{\ell(x_2)}{\ell'(x_{i_2})(x - x_{i_2})} \frac{\ell(x_3)}{\ell'(x_{i_3})(x - x_{i_3})}. \tag{9}$$

Here $\ell(x)$ is defined in Eq. (3), and $\ell'(x_{i_{\{1,2,3\}}})$ can be pre-computed and stored in table. Thus, reconstructing with a Lagrange polynomial basis requires $O(|N|^2)$ setup, but only $O(|N|)$ work per query, which is exactly what the Newton's basis achieves. The final form for the Lagrange polynomial using Gauss-Lobatto-Legendre points with $n = N^{1/3}$ is[7]

$$p^N_{\mathbf{x}_i}(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} = \mathbf{x}_i, \\ \frac{(x_1-1)(x_1+1)P'_n(x)}{n(n+1)P_n(x_{i_1})(x-x_{i_1})} \frac{(x_2-1)(x_2+1)P'_n(x)}{n(n+1)P_n(x_{i_2})(x-x_{i_2})} \frac{(x_3-1)(x_3+1)P'_n(x)}{n(n+1)P_n(x_{i_3})(x-x_{i_3})}, & \text{otherwise.} \end{cases} \tag{10}$$

The Error in a Cell

As previously stated, we choose to subdivide octree cells until the relative error in it is less than 10^{-6} or a maximum level of subdivisions has been achieved. We now describe how the error is computed. In univariate approximation it is popular to use the remainder formula (Eq. (1)) to derive an error bound. To generalize this to Ciarlet's formula let $\hat{D}_i f$ be the maximum $D^{|N|}_{\mathbf{x}_i-\mathbf{x}} f$ for all $\mathbf{x}$ in the region of the integral; then,

$$\begin{aligned} f(\mathbf{x}) - L_N f(\mathbf{x}) &= \sum_{\mathbf{x}_i \in N} p^N_{\mathbf{x}_i}(\mathbf{x}) \int_{[\mathbf{x}_i, \mathbf{x}, \ldots, \mathbf{x}]} D^{|N|}_{\mathbf{x}_i - \mathbf{x}} f & (11) \\ &\leq \sum_{\mathbf{x}_i \in N} p^N_{\mathbf{x}_i}(\mathbf{x}) \int_{[\mathbf{x}_i, \mathbf{x}, \ldots, \mathbf{x}]} \hat{D}_i f. & (12) \end{aligned}$$

Examining this formula we come to two conclusions: *(i)* Whatever bound we get will not be tight as $\hat{D}_i f$ represents a maximum in both space and direction; this is worse than the univariate counterpart, which only concerns itself with a maximum over space. *(ii)* Finding a bound with this method will be a challenging and computationally slow task. In light of this we eschewed a technically elegant solution for one that is simpler and less error prone. To find the error in a cell we took random samples of the approximate force $\mathbf{F}_a$ and exact force $\mathbf{F}_e$, and computed the relative error as $\|\mathbf{F}_a - \mathbf{F}_e\|/\|\mathbf{F}_e\|$. The maximum such error was taken as the error bound of a cell.

RESULTS AND DISCUSSION

Distance From an Asteroid

Our first experiment was designed to test the hypothesis that gravity in cells closer to the asteroid would be more difficult to approximate than in cells farther away. We setup an interpolation cell with 400 m sides and order 6 polynomials at progressively closer locations. To keep our test simple we used a cuboid with the approximate dimensions of Castalia as our asteroid. The cells used were in two types of locations. One set was at varying distances from a flat face of the asteroid and another set was at varying distances from an edge of the asteroid. Figures 3 and 4 show some test cases, and Table 1 summarizes the results.

In both cases we can see that bringing the cell closer to the asteroid increases the error of approximation. For the edge on case we see a dramatic difference, though this is probably due to the presence of the edge, which is a high curvature feature.

Table 1 EFFECT OF DISTANCE ON ERROR

Distance	Face approach max error	Edge approach max error
far	2.29×10^{-8}	3.89×10^{-9}
medium	2.95×10^{-7}	7.76×10^{-8}
adjacent	9.77×10^{-7}	5.37×10^{-3}

High Curvature Features

To further investigate the effects of high curvature features we setup another test. In this experiment we used the same Castalia-like cube, but added a tetrahedron to one face to act like a small bump on the surface. We started with a bump of 200 m edge lengths and scaled down from there. For each configuration we used a cell with 25 m sides and order 6 interpolation, placed such that the center of the closest face was aligned with the tip of the tetrahedron. Figure 5 shows some test cases, and Table 2 summarizes the results.

Table 2 EFFECT OF SMALL FEATURES ON ERROR

Feature size (m)	Max error
200	6.84×10^{-4}
175	6.62×10^{-4}
150	6.41×10^{-4}
125	6.19×10^{-4}
100	6.00×10^{-4}
75	5.79×10^{-4}
50	5.62×10^{-4}
25	5.37×10^{-4}
0	3.98×10^{-13}

As we want errors beneath 10^{-5}, it seems that even a slight bump can give polynomial interpolation significant problems. Given that the 1998 ML14 model has edge lengths as small as to 9 m, this is especially troubling. The solution, as we shall see, is to use smaller cells that wrap around the feature instead of one large cell.

Varying the Size of a Cell

Given the poor performance of polynomial interpolation for cubes at the 25 m scale, we wanted to know how small our cubes would need to be in order to achieve the desired error. In this experiment we studied the effect of varying the size of a cell on the accuracy of our interpolation. To test this we placed a cube cell with order 6 interpolation centered near a tip of a 1998 ML14 model. Starting with an edge length of 250 m, we halved each dimension of the cube for every subsequent experiment. Figure 6 shows some test cases, and Table 3 summarizes the results.

We can use these results to motivate the case for octrees. First, assume the domain of interest is a cube surrounding 1998 ML14 with edge length 2500 m; this is a fair assumption because anywhere outside that range can use low order spherical harmonics as a fast approximation. From these results

Table 3   EFFECT OF CELL SIZE ON ERROR

Size (m)	Max error
250	5.75×10^{-2}
125	2.85×10^{-2}
62.5	1.46×10^{-2}
31.3	7.24×10^{-3}
15.6	3.63×10^{-3}
7.81	1.86×10^{-3}
3.91	9.03×10^{-4}
1.95	4.63×10^{-4}
0.977	2.28×10^{-4}
0.488	1.14×10^{-4}
0.244	5.66×10^{-5}
0.122	2.82×10^{-5}

we conclude that we will need resolution down to 12 cm to capture the fine detail near the surface. With a regular grid this would require dividing the domain into $(2500/0.12)^3 \approx 9 \times 10^{12}$ cells. If each cell contains order 6 interpolation, we need $7^3 \times 3 = 1029$ double precision coefficients, or 8232 bytes per cell (the power of 3 in this is for the 3 dimensions of the cell and the factor of 3 is for the 3 components of force). The total memory cost for such a model is about 74 Petabytes. Even at order 2, we would need around 6 Petabytes to store a regular grid.

Varying the Polynomial Order

Our next experiment focused on varying the order of approximation; specifically, we explored the interaction between distance from the asteroid and order of approximation. Two cube cells centered at (441 m, 231 m, 0 m) (1-radius) and (750 m, 231 m, 0 m) (2-radii) with 250 m edge length and polynomial interpolants between orders 1 and 7 were tested. Figures 7 and 8 show some test cases, and Table 4 summarizes our results.

Table 4   EFFECT OF INTERPOLATION ORDER ON ERROR

Order	Distant 250 m domain max error	Close 250 m domain max error
1	7.94×10^{-2}	3.48×10^{-1}
2	5.37×10^{-3}	1.40×10^{-1}
3	8.91×10^{-4}	1.13×10^{-1}
4	1.20×10^{-4}	7.59×10^{-2}
5	2.34×10^{-5}	7.16×10^{-2}
6	3.23×10^{-6}	5.75×10^{-2}
7	6.03×10^{-7}	5.37×10^{-2}

Far away there is a clear benefit to using high order polynomials for approximating the gravitation. Closer in, however, these results show diminishing returns for higher order polynomials. We conclude that the appropriate strategy for our octree uses few high order cubes far away, and many low order cubes closer to the asteroid. This is some what counterintuitive as one may expect high order approximations to yield the most significant benefits where the field changes most rapidly.

Octree Models

We now present preliminary experiments in generating octrees to approximate gravitational force near the asteroid 1998 ML14. In this exploratory test we only modeled a domain 1/125 of the volume of our final intended dimensions; the domain origin is (400 m, -250 m, -250 m) and the size is (500 m, 500 m, 500 m). The error tolerance was set to 10^{-6}, and the number of subdivision levels capped at 8 (smallest cells are approximately 4 m). Following the conclusions of the previous section, interpolation order was controlled as follows: cells within the first two subdivisions were order 6, cells in the last two subdivisions were order 2, and the rest were order 4.

Figure 9 shows cross sections of the error in the xy-, xz-, and yz- planes. As we can see, errors are very well behaved at 1.5 and 2 radii from the center of the asteroid. In fact, only when we get very close to the surface are we in danger of violating our goal of 10^{-5}.

This model costs about 80 MB of storage; thus, we predict the full volume would cost on the order of 80 MB $\times 125 \approx 10$ GB, which is close to what modern personal workstations can handle. This estimate is actually rather high as the domain we have explored is more costly than most. For example, regions inside the asteroid have zero cost as they are not approximated, and regions farther away can easily use large high order cells. As such, we would not expect a full domain to actually cost 10 GB of memory.

Speed Tests

In this experiment we measured the comparative performances of the competing models. 1998 ML14 was used as our asteroid model to enable comparison with previous work. Table 5 summarizes the relative speeds (1.0 being polyhedral method) of polynomial interpolation and other methods. We can see that the polynomial interpolation scheme compares favorably with other methods. Compared to Ref. 1 we have similar performance, but better understanding of the errors. Specifically, while Ref. 1 only reports errors for orbits at 3 radii from the body, our error estimates go all the way to the body. Furthermore, our errors at 3 radii are closer to 10^{-5}, two orders of magnitude better than the coarse model results in Ref. 1. See Figures 9 and 7 at order 6.

Table 5 RELATIVE SPEEDS OF AVAILABLE METHODS

Model	Speed Factor
polyhedral	1.0
order 6 polynomial	0.0104
degree and order 6 spherical harmonics	0.0145
coarse shape, Taylor series[1]	0.091
coarse shape, Taylor series, histories[1]	0.01

CONCLUSION

We have presented an efficient method for approximating gravitational force of small, irregular bodies that is accurate enough for planning missions near them. Our technique combines an adaptive spatial data structure with polynomial interpolation to cover the entire domain with an approximation. The flexible spatial hierarchy enables us to refine our model in regions that are difficult to approximate (e.g. near high curvature regions), and enables error guarantees on the model. This method is shown to produce satisfactory results at 2 and 3 radii from the center of the asteroid, and given sufficient subdivisions, even 1 radius from the asteroid. We believe this model is competitive with its counterparts for Monte Carlo simulations of spacecraft trajectories passing near small bodies. A summary and comparison of the computational characteristics for each available method follows.

Octree with Polynomial Interpolation

Memory: Our method has very large memory requirements. Every octree cell requires $(\text{order} + 1)^3 \times 3$ coefficients, and there may be millions of cells. Our preliminary results indicate a 10 GB upper bound for the memory requirements of this technique. **Speed:** Using interpolating polynomials permits a constant time reconstruction of the force within a cell, and finding the correct octree cell is a $O(\log N)$ operation. In practice this gives a $100\times$ speed up over the polyhedral method. **Errors:** Errors can be controlled to within user tolerances. In our experiments we reached our 10^{-5} goal quite near the asteroid (further than 4 m from the surface), and surpassed it at points 2 and 3 radii away (from the center).

Spherical Harmonics

Memory: Spherical harmonics have a very small memory footprint: only $(\text{order} + 1)^2$ coefficients to store the whole model. **Speed:** Depending on the order and degree used spherical harmonics can be and up to $100\times$ faster than the polyhedral method. At best, spherical harmonics approaches the speed of our method, and compares favorably with other methods. **Error:** No matter what order is chosen errors near non-convex regions of the asteroid render this method useless for missions close to a small body.[13] This puts spherical harmonics at a significant disadvantage to both the polyhedral method and the method presented in this paper.

Mascons[3]

Memory: Mascons use memory linear to the number of point masses used. In practice this number is in the thousands; thus, mascon memory footprint is very small. **Speed:** Mascons are faster than polyhedral methods, however, they are still subject to a linear run-time complexity. In practice their performance is irrelevant as they have non-trivial error. **Error:** Errors for this method have not been theoretically bounded, and experiments show that large errors do exist.[3]

Polyhedral Method[3]

Memory: The primary memory cost of this method is storing the polyhedral model and its associated data. As most models have only tens of thousands of elements polyhedral methods require a small amount of memory. By this metric the polyhedral method performs better than our method. **Speed:** Calculation requires iterating over every edge and face; furthermore, each edge and face calculation includes a transcendental function. Hence, polyhedral methods are slow to compute gravitational force; our technique is two orders of magnitude faster. **Error:** As long as the polyhedral model and density assumptions are not far from the truth this method produces exact results. Of course, any errors in this method would spoil other models using it as a base line; e.g., the one considered in this paper.

Modified Polyhedral Method[1]

Memory: In addition to the memory costs of the polyhedral method, modified polyhedral methods cache many prior calculations; this, however, can only take a constant factor beyond the nominal. Hence, modified polyhedral methods also have modest memory requirements. **Speed:** Depending on the orbit this varies between $10\times$ and $100\times$ faster. In ideal circumstances this reaches our performance, but in general $100\times$ speed ups are not achieved. **Error:** Errors at 3-radii are close to or below 10^{-3}; closer trajectories may experience more or less error. Our method has an advantage here because we can adaptively drive error down as needed, and thereby guarantee certain error bounds.

FUTURE WORK

There are still many experiments that need to be run. Foremost among them is the need for more complete octree representations of gravity around an asteroid. At present only exploratory models have been created, but we are hoping that recent access to parallel computers will enable us to create complete models. Upon completing these larger models we shall use our method in complete trajectory simulations, and compare these results to other methods as a final verification.

Other avenues of research from here are: *(i)* performing Monte Carlo simulations with this model, *(ii)* trying different approximation schemes within nodes, and *(iii)* updating these kinds of models with data captured during missions. The last point warrants additional explanation. One of the primary draw backs of the polyhedral model is that incorporating measured error in force estimates requires updating the physical description of the small body. This fitting step is challenging to say the least. On the other hand, interpolatory techniques should be more apt at making corrections because they are based on measurements to begin with. Aside from the speed of polynomial approximations, this is potentially their greatest strength.

ACKNOWLEDGMENTS

The authors would like to thank William N. Bell for useful discussions and for suggesting the use of octrees and cuboids rather than tetrahedra, Keenan M. Crane for rendering our asteroids and assistance with our octree visualization, and the LORENE[14] project for software that computes the zeros of Gauss-Lobatto-Legendre polynomials. The 1998 ML14 shape model was provided by the Planetary Data System.[15,16]

REFERENCES

[1] L. A. Cangahuala, "Augmentations To The Polyhedral Gravity Model To Facilitate Small Body Navigation," *AAS/AIAA Space Flight Mechanics Meeting, Copper Mountain, Colorado*, American Astronautical Society and American Institute of Aeronautics and Astronautics, 2005, pp. 685–698.

[2] R. A. Werner, “The gravitational potential of a homogeneous polyhedron or don’t cut corners,” *Celestial Mechanics and Dynamical Astronomy*, Vol. 59, 1994, pp. 253–278.

[3] R. A. Werner and D. J. Scheeres, “Exterior Gravitation of a Polyhedron Dervied and Compared with Harmonic and Mascon Gravitation Representations of Asteroid 4769 Castalia,” *Celestial Mechanics and Dynamical Astronomy*, Vol. 65, 1996, pp. 313–344.

[4] J. L. Junkins, “Investigation of finite-element representations of the geopotential,” *AIAA Journal*, Vol. 14, 1976, pp. 803–808.

[5] R. C. Engels and J. L. Junkins, “Local representation of the geopotential by weighted orthonormal polynomials,” *J. Guidance and Control*, Vol. 3, 1980, pp. 55–61.

[6] T. Sauer, “Polynomial Interpolation in Several Variables: Lattices, Difference, and Ideals,” *Topics in multivariate approximation and interpolation* (K. Jetter, M. Buhmann, W. Haussmann, R. Schaback, and J. Stoeckler, eds.), Elsevier Science, 2005.

[7] G. E. Karniadakis and S. Sherwin, *Spectral/hp Element Methods for Computational Fluid Dynamics*. Oxford Science Publications, 2005.

[8] J.-P. Berrut and L. N. Trefethen, “Barycentric Lagrange interpolation,” *SIAM Rev.*, Vol. 46, 2004, pp. 501–517.

[9] S. F. Frisken and R. N. Perry, “Simple and Efficient Traversal Methods for Quadtrees and Octrees,” *The Journal of Graphics Tools*, 2002.

[10] P. J. Davis, *Interpolation and approximation*. Blaisdell Publishing Co. Ginn and Co. New York-Toronto-London, 1963.

[11] M. Gasca and T. Sauer, “Polynomial interpolation in several variables,” *Adv. Comput. Math.*, Vol. 12, 2000, pp. 377–410.

[12] P. G. Ciarlet and C. Wagschal, “Multipoint Taylor formulas and applications to the finite element method,” *Numer. Math.*, Vol. 17, 1971, pp. 84–100.

[13] W. Heiskanen and H. Moritz, *Physical Geodesy*. Freeman and Company, 1967.

[14] E. Gourgoulhon, P. Grandclément, and J.-A. Marck, “http://www.lorene.obspm.fr/school/Monday_doc/,” Polynomial interpolation software.

[15] C. Neese, “Ed. Small Body Radar Shape Models V2.0,” EAR-A-5-DDR-RADARSHAPE-MODELS-V2.0, NASA Planetary Data System, 2004.

[16] S. J. Ostro, R. S. Hudson, L. A. M. Benner, M. C. Nolan, J. D. Giorgini, D. J. Scheeres, R. F. Jurgens, and R. Rose, “Radar observations of asteroid 1998 ML14,” *Meteoritics and Planetary Science*, Vol. 36, 2001, pp. 1225–1236.

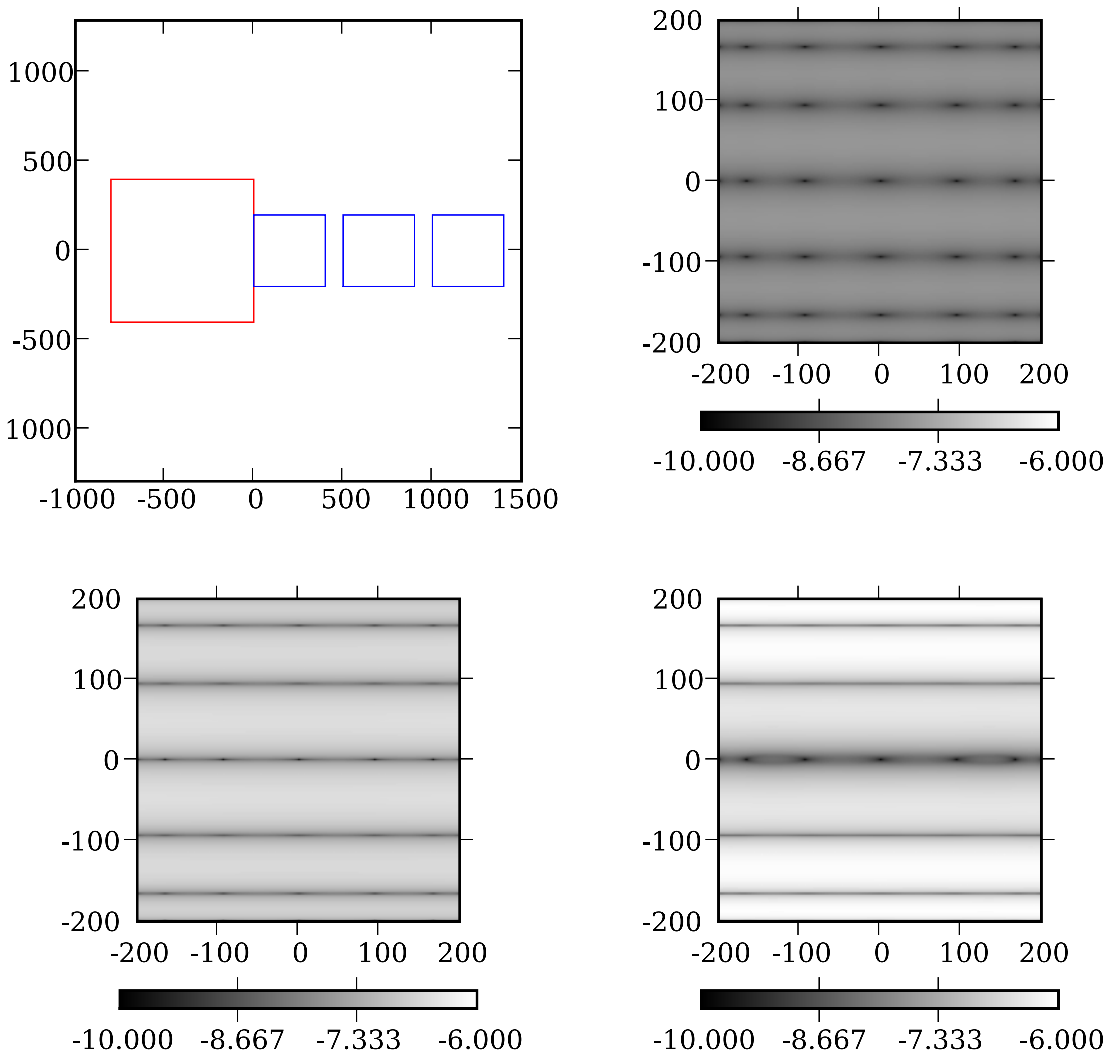

Figure 3 Error as a function of distance from a flat face of a body. The shaded plots show the logarithm of relative error of gravitational force along the face of the cell closest to the body. Upper left: Context for the test. The body is shown in red and the cells in blue. Upper right: A cell 1000 m away. Lower left: A cell 500 m away. Lower right: A cell adjacent to the body. Each cell used order 6 interpolation, and the true values were taken to be the ones computed by the polyhedral method.

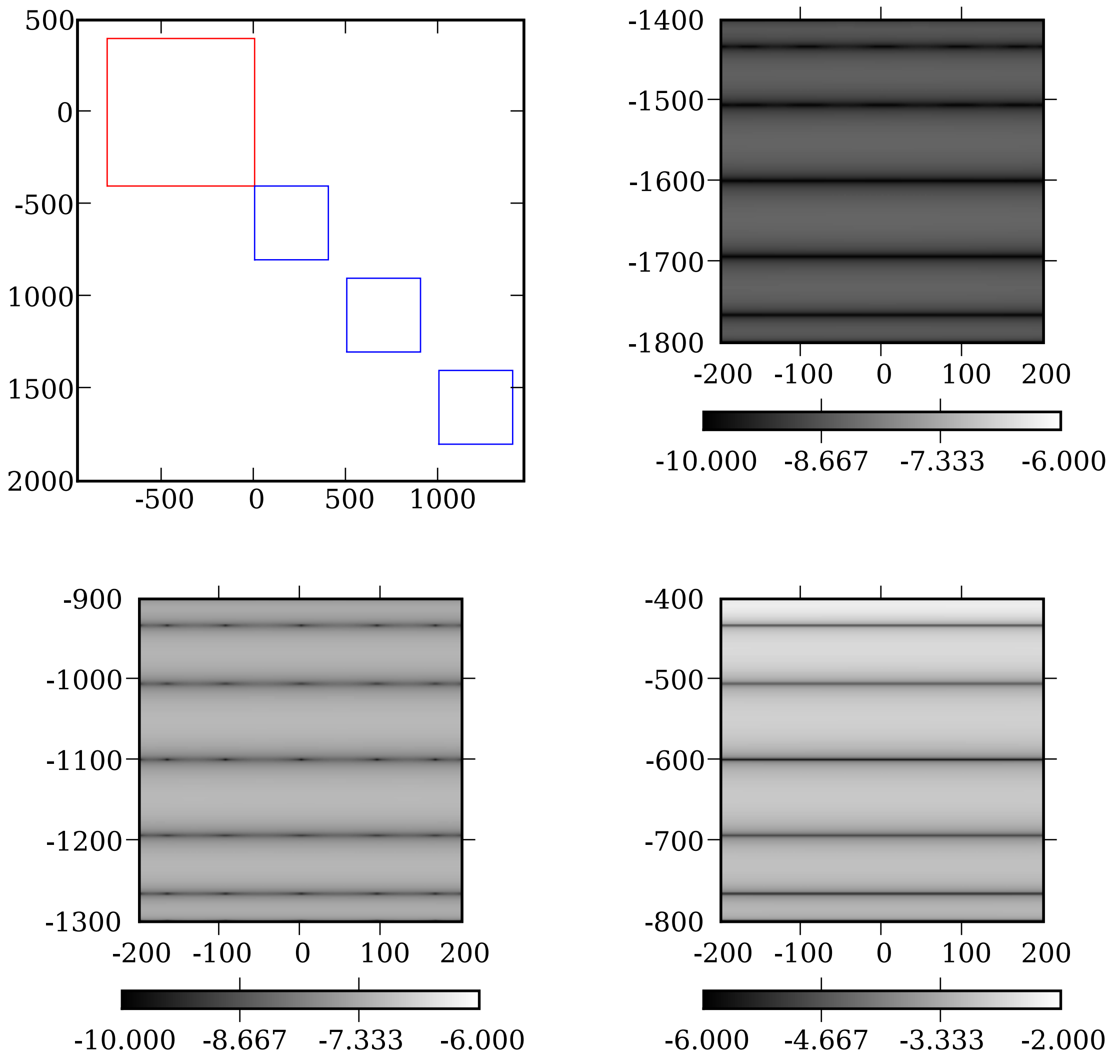

Figure 4 Error as a function of distance from an edge of a body. The shaded plots show the logarithm of relative error of gravitational force along the face of the cell closest to the body. Upper left: Context for the test. The body is shown in red and the cells in blue. Upper right: A cell 1414 m away. Lower left: A cell 707 m away. Lower right: A cell adjacent to the body. Each cell used order 6 interpolation, and the true values were taken to be the ones computed by the polyhedral method.

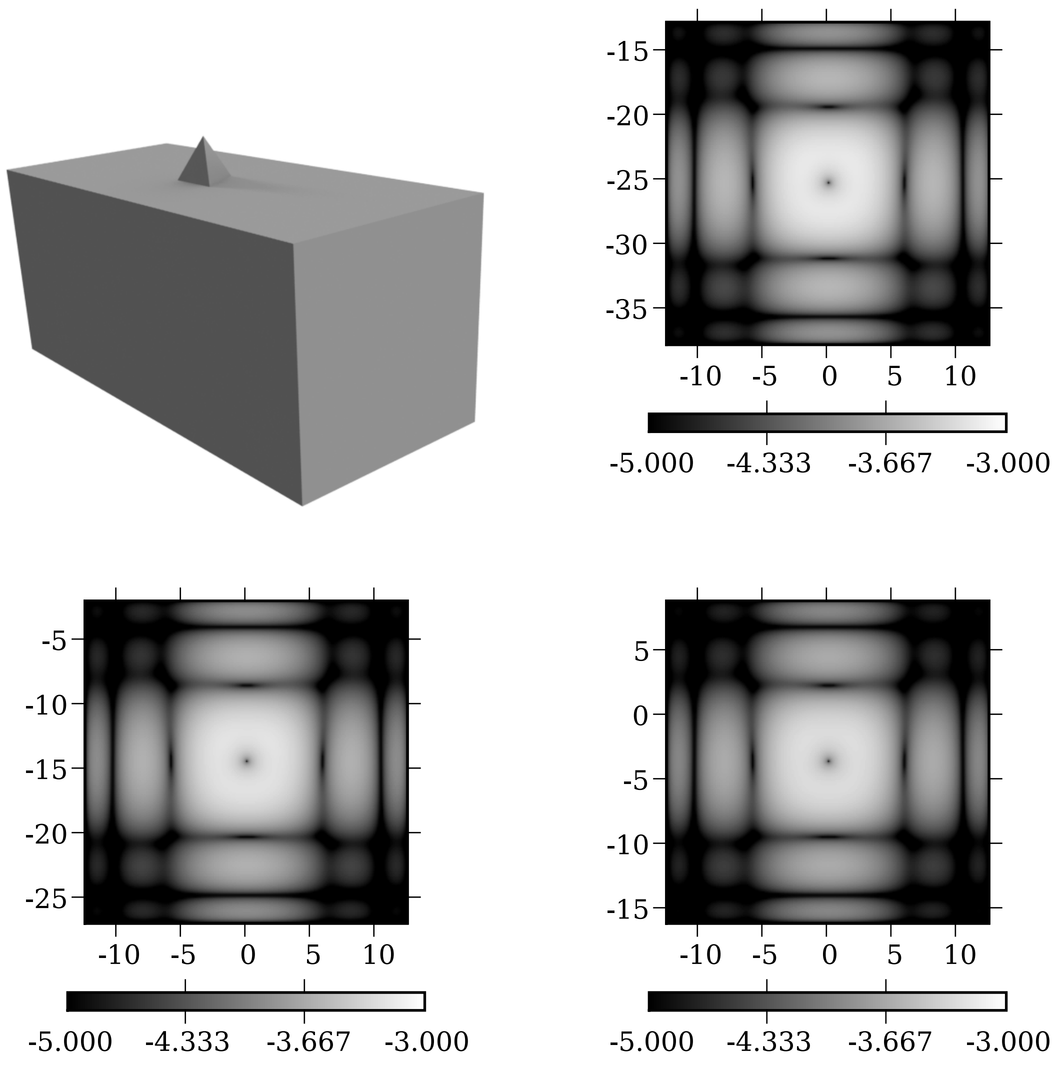

Figure 5 Effect of small features on error. The shaded plots show the logarithm of relative error of gravitational force along the face of the cell touching the tetrahedron feature. The goal is to avoid errors larger than 10^{-5}, so saturated white regions indicate a poor approximation. Upper left: An image of our test body. In this image the tetrahedron had 200 m edge lengths. Upper right: A 175 m tetrahedron. Lower left: A 100 m tetrahedron. Lower right: A 25 m tetrahedron. Each cell used order 6 interpolation, and the true values were taken to be the ones computed by the polyhedral method.

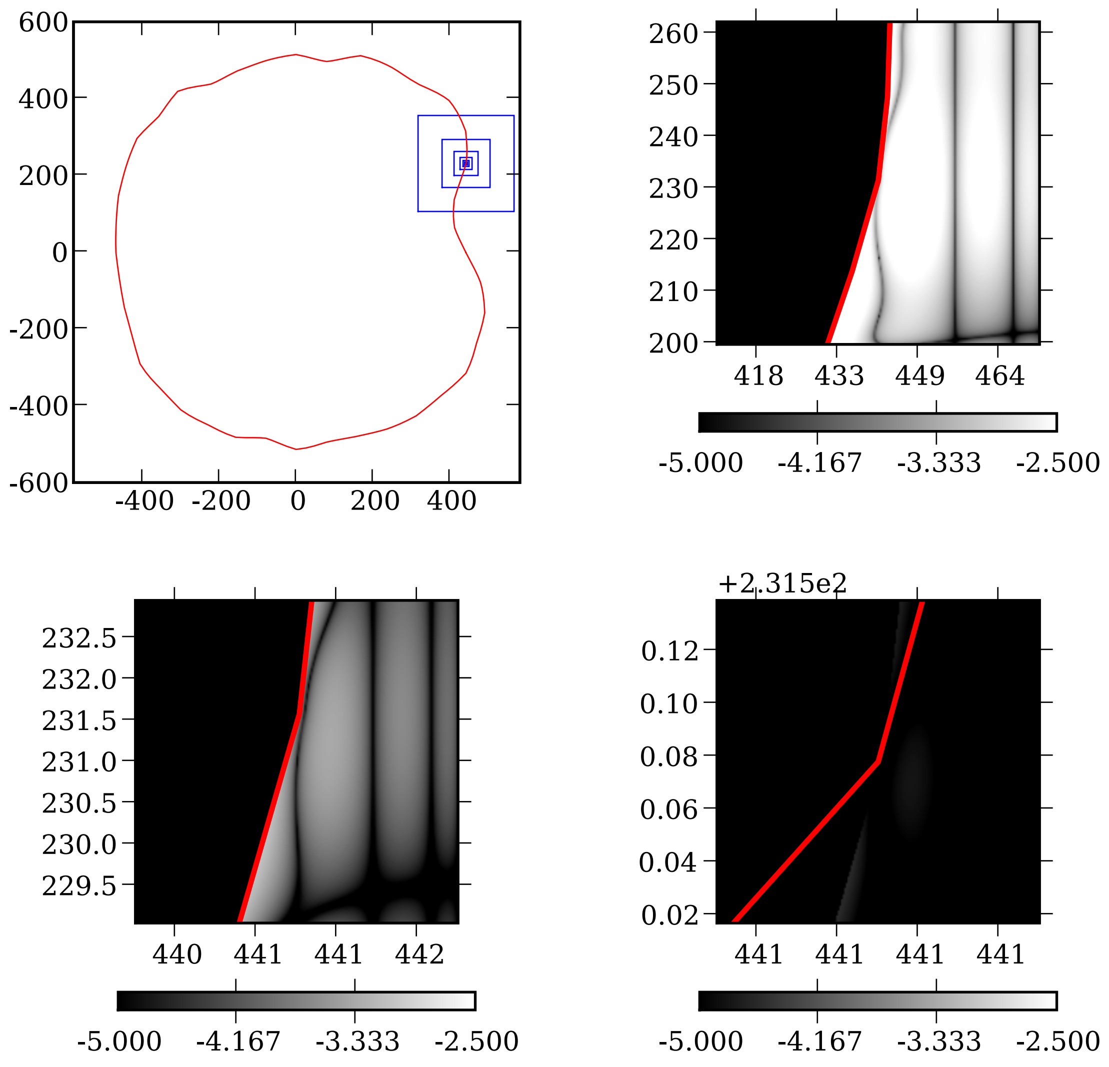

Figure 6   Error as a function of cell size. The shaded plots show a cross section of the logarithm of relative error of gravitational force. The goal is to avoid errors larger than 10^{-5}, so saturated white regions indicate a poor approximation. Upper left: Context for the test. The red outline shows the boundary of asteroid 1998 ML14; the blue boxes indicate the cells we tried. Upper right: A cell with 250 m edge lengths. The saturated white regions show large errors are present. Lower left: A cell with 62.5 m edge lengths. The saturated white regions show large errors are present. Lower right: A cell with 3.91 m edge lengths. The errors here are within tolerance. Each cell used order 6 interpolation, and the true values were taken to be the ones computed by the polyhedral method.

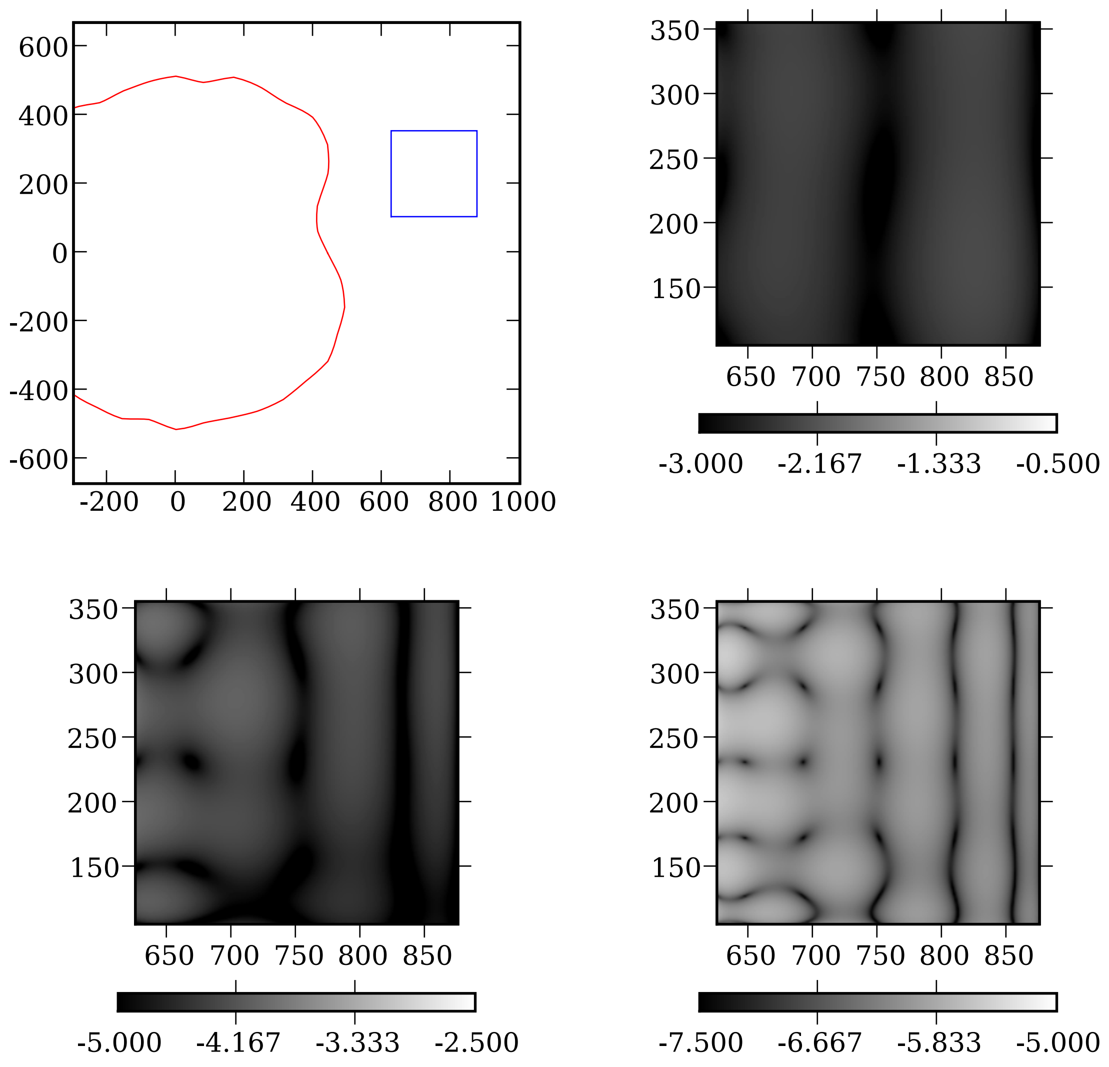

Figure 7 Error as a function of order of interpolation in a distant cell. The shaded plots show a cross section of the logarithm of relative error of gravitational force. Upper left: Context for the test. The red outline shows the boundary of 1998 ML14, the blue box indicates our cell. Upper right: Interpolation at order 2. Lower left: Interpolation at order 4. Lower right: Interpolation at order 6. The true values were taken to be the ones computed by the polyhedral method.

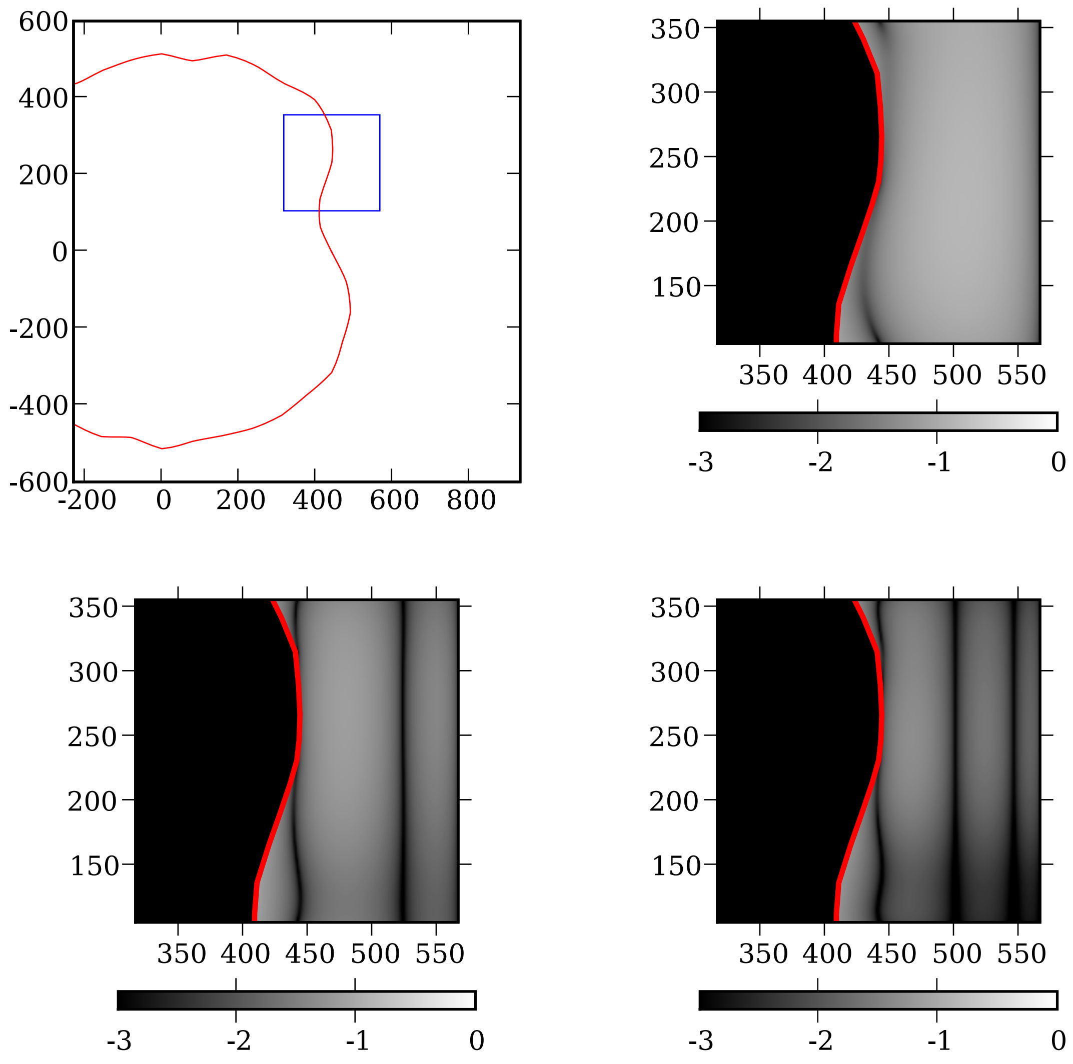

Figure 8 **Error as a function of order of interpolation in a cell close to the asteroid. The shaded plots show a cross section of the logarithm of relative error of gravitational force. Upper left: Context for the test. The red outline shows the boundary of 1998 ML14, the blue box indicates our domain. Upper right: Interpolation at order 2. Lower left: Interpolation at order 4. Lower right: Interpolation at order 6. As we can see all errors reported are outside the acceptable range. The true values were taken to be the ones computed by the polyhedral method.**

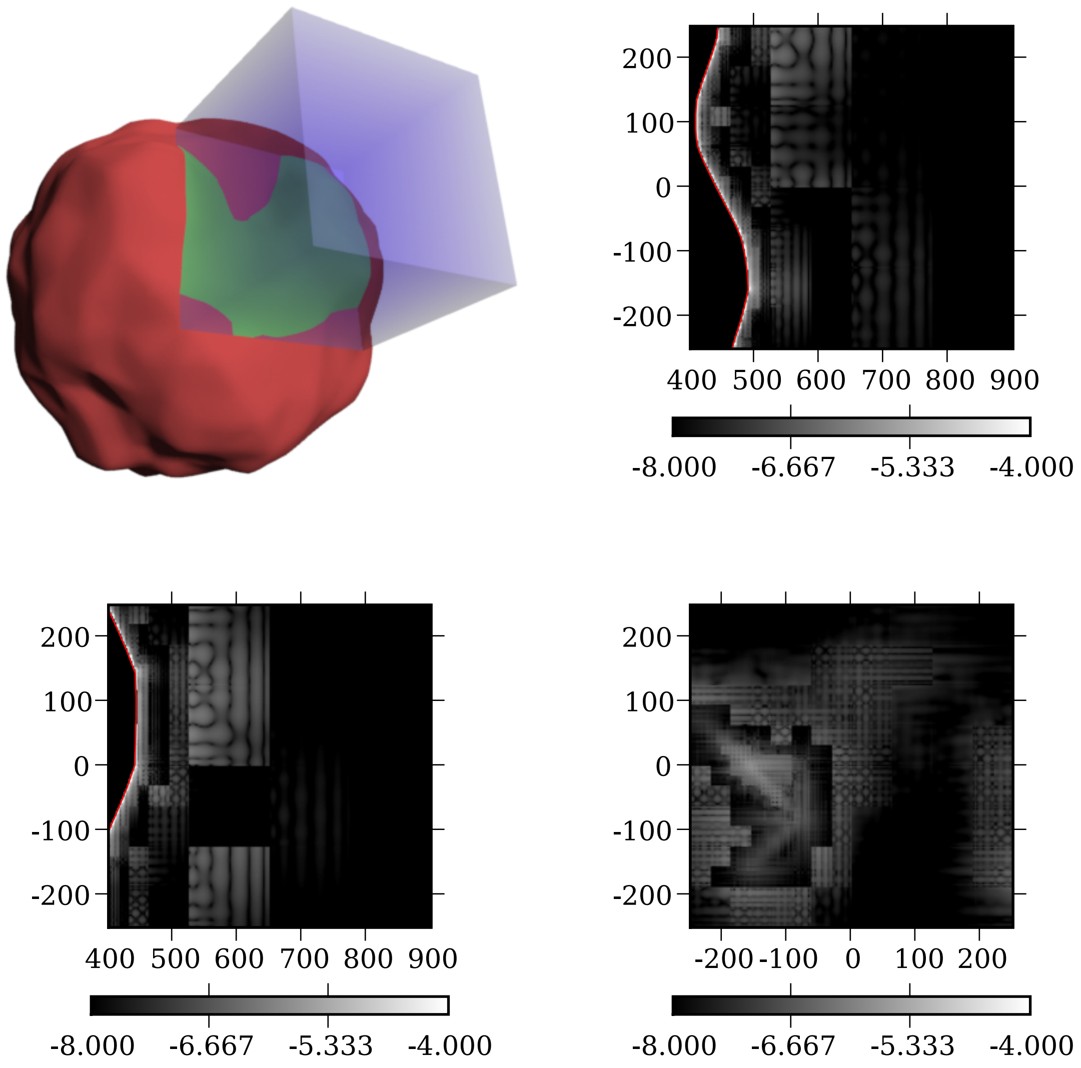

Figure 9 Error of a test octree model representing 1/125 our final domain. The shaded plots show the logarithm of relative error in gravitational force. Upper left: Context for the test. The green region shows where the asteroid is inside the cell. Upper right: Error along the xy-plane at z=0 m. Lower left: Error along the xz-plane at y=0 m. Lower right: Error along the yz-plane at x=500 m. The error in most of the cells in all these is less than 10^{-5} as desired. The only exceptions are in the cells very close to the asteroid in the top right and bottom left figure. These are cells of size about 4 m. Cells used a variable order interpolation depending on their size, orders ranged from 2 to 6. The true values were taken to be the ones computed by the polyhedral method.