STRUCTURE PRESERVING APPROXIMATIONS OF CONSERVATIVE FORCES FOR APPLICATION TO SMALL BODY DYNAMICS

Andrew Colombi, Anil N. Hirani

Department of Computer Science,
University of Illinois at Urbana-Champaign,
201 North Goodwin Avenue, Urbana, IL 61801.

and

Benjamin F. Villac

Mechanical and Aerospace Engineering,
University of California, Irvine
4200 Engineering Gateway, Irvine, CA 92697-3975.

AIAA/AAS Astrodynamics Specialists Conference

Honolulu, HI, August 18–21, 2008

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

Structure Preserving Approximations of Conservative Forces for Application to Small Body Dynamics

Andrew Colombi [*] and Anil N. Hirani [†]
University of Illinois at Urbana-Champaign, Urbana, IL, 61801, USA
Benjamin F. Villac [‡]
University of California, Irvine, CA, 92697, USA

Abstract

Approximation based methods, such as the cubetree algorithm, have proved to be significantly faster than traditional methods for complex force evaluations near small irregular bodies. Such methods also hold the promise of simplifying the inclusion of experimental data to update the force model. However, the cubetree algorithm does not preserve intrinsic properties of gravitational force such as continuity, divergence freedom or exactness. These properties may be needed for trajectory optimization, for the use of geometric (e.g. symplectic) integrators for long term propagation and for other trajectory design problems. This paper presents several adaptive schemes preserving global continuity, exactness or divergence-freedom and discusses the difficulties involved in preserving all of these properties globally.

I. Introduction

With the continuing increase in computing power, large scale problems, which were considered very difficult in the past decade, have now become tractable. In particular optimization methods based on genetic algorithms have recently attracted many researchers in astrodynamics [1, 2] and simulations involving hundred of thousands of trajectory propagations have recently appeared [3, 4]. However, there is still a need to improve the core algorithms, such as ODE integration and force function evaluation, for several reasons.

First, faster elementary methods mean that larger, more realistic systems can be considered. In particular, large simulations generally assume fairly simple dynamics, and are more challenging for complex force models such as small body environments. Secondly, as autonomous navigation becomes a reality, there is an increased demand for fast on-board computational tools [5]. Finally, current research in numerical integration emphasizes the importance of preserving fundamental geometric structures present in the modeled dynamics. Such issues have appeared to be of prime importance for long-term integration, such as encountered in astronomy, and in the analysis of numerical experiments.

Recently we introduced a novel numerical scheme (the *cubetree algorithm*) for the fast evaluation of gravitational force around irregular bodies and showed that it provides a significant speed improvement over other methods [6]. The scheme exploits the availability of large storage capacities to reduce the "on-line" computations. Specifically, by locally interpolating the force field around a

[*]Graduate Student, Department of Computer Science, 201 North Goodwin Avenue, Urbana, IL, 61801; Email: acolombi@gmail.com

[†]Assistant Professor, Department of Computer Science, 201 North Goodwin Avenue, Email: hirani@cs.uiuc.edu, URL: http://www.cs.uiuc.edu/hirani

[‡]Assistant Professor, Mechanical and Aerospace Engineering, 4200 Engineering Gateway, Email : bvillac@uci.edu

small body this algorithm decreased computational effort of spacecraft trajectories integration by a factor of a hundred. While such results are of particular interest for large simulations, such as Monte Carlo analysis, the method may also be useful for smaller, on-board computation due to its relatively light load on the processor. However, the method has not been optimized and requires a significant memory footprint. Also, several desirable mathematical properties of gravitational force have not been considered and are not preserved by the cubetree algorithm.

A. Questions addressed

In this paper we address some of these issues and develop improved approximation schemes for potential energy and force representation around small irregular bodies. In particular, the following questions are considered:

1. *Smoothness*: When several interpolation domains are considered, discontinuities at the boundary represent a fundamental obstacle for theoretical investigation due to the continuous nature of the force represented. These discontinuities may also lead to a deterioration of integration performance.

2. *Exactness*: All conservative forces are the gradient of some potential field. The cubetree method presented in Reference [6] does not respect this qualitative feature. This is especially important for long term simulation where qualitative features of the trajectories are of primary interest.

3. *Divergence Freedom*: The force of gravitation is divergence free. This property has many theoretical implications, but is not necessarily preserved by standard interpolation schemes.

4. *Efficiency*: What interpolation schemes allow for smaller memory footprint while ensuring sufficient accuracy? This question may be addressed in both the approximation method, and the choice of subdivision technique used for partitioning the space around the body.

Note that addressing the first two issues is a necessary step for the application of geometric integrators such as symplectic integrators. Although geometric integrators have been applied to problems with discontinuities and dissipation, the structure of discontinuities or dissipation is part of the theoretical framework in those cases [7]. In an approximation scheme like the cubetree method it is due to the approximation of the force field.

In order to tackle the above issues several modifications of the cubetree algorithm have been considered. In the first section a regularization of cubetree that produces a continuous approximation is presented. While easily implementable, this solution requires tighter tolerances on the approximated function which increases the memory footprint of the model. The third section presents a different approach based on *least squares approximation* using hierarchical B-spline refinements which provided a less stringent error requirement while addressing both continuity and exactness. Finally, we discuss in the fourth section, divergence free approximations and the challenges in obtaining all three properties at the same time globally.

II. Cubetree Algorithm

In order to provide a basis for comparison and to better understand the challenges in addressing the above issues, a review of the cubetree algorithm [6] is presented in Section II.A below. Then we discuss obvious generalizations and inherent difficulties associated with this class of algorithms. Further details on the description, implementation and performance of cubetree can be found in Reference [6].

A. The Cubetree Algorithm

The cubetree algorithm provides an efficient method for approximating the gravitational force generated by a small body in its immediate neighborhood[1], which allows for a fast evaluation of the gravitational forces in such regions. The method consists of two parts: (1) a local approximation scheme for building a force model in a small cuboid domain and (2) a spatial data structure to distribute these local approximations adaptively.

Local force (or potential) approximation is achieved by polynomial interpolation at the tensor product of Gauss-Legendre-Lobatto (GLL) points [8]; see Figure 1. This forms a model for a single cuboid region near a small body. A spatial data structure, called an *octree*, creates a complete approximation for the entire domain by adaptively allocating local approximations. An octree – or quadtree in 2 dimensions – subdivides space by halving its *cells* in a tree structure that cumulatively represents the domain of approximation. Once the domain is divided and each cell has been locally approximated, force at a point $\boldsymbol{x}$ can be quickly calculated by following the tree structure to the appropriate local approximation. Figure 1 shows how a quadtree subdivides a region and how a query is resolved by the tree.

Cubetree approximation is extremely fast compared to the polyhedral method [9, 10]. Each query requires an $O(\log N)$ tree lookup and a constant time polynomial reconstruction. In practice speed-ups between $300\times$ and $100\times$ over the polyhedral method can be expected. To produce the approximation, however, requires significant up front effort: First, an initial model or data set must be available to approximate. Then, significant computational effort is required to construct the model. The model in [6] requires 1150 CPU hours to initialize, and used 64 processors coordinated with the Message Passing Interface (MPI) [11].

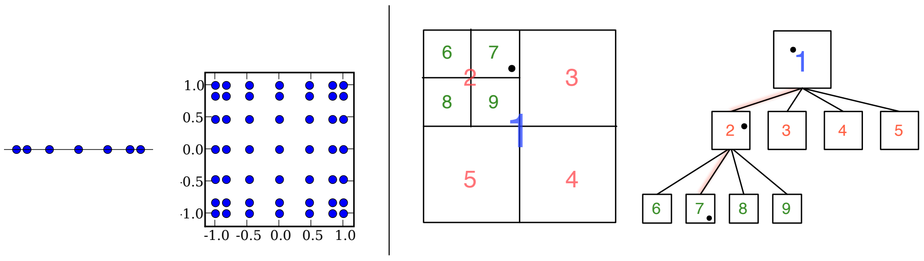


Figure 1: Gauss-Lobatto-Legendre nodes. Left: 1-dimensional Gauss-Lobatto-Legendre nodes, and their 2-dimensional tensor product. Right: A quadtree viewed as a tree and geometrically. Labels show the mapping between geometric and tree views. The point represents a query; the tree view shows the query being resolved. Cells 3 to 9 are leaf cells.

B. Localization and Boundary Mismatches

The initial development of the cubetree algorithm did not explore the choice of the approximation scheme within each cell; however, the cubetree strategy can be employed with different local approximation methods. Any local approximation scheme may be used as long as the following properties are satisfied:

1. The local approximation domains (cells) can cover the entire region of interest (i.e. the neighborhood of a small body).
2. The cells can be subdivided in a way that is suitable for the octree data structure while resulting in a covering of the original cell and subdivided cells of smaller size.

[1]That is, the space that is not represented efficiently by spherical harmonics, namely the region between the surface and the sphere of convergence the (exterior) spherical harmonics.

3. For a fixed cell, the approximation scheme can be adjusted (increased order) to meet a set tolerance.

4. For a fixed approximation order, the error in a local approximation decreases within subdivided cell.

Note that in the case of gravity the third requirement is satisfied by any polynomial interpolation scheme: Gravity is analytic and thus well represented by their Taylor series locally.

The freedom to choose an approximation scheme within each cell enables preservation of any particular property locally – i.e. within a cell. In particular the interpolation scheme considered already preserves continuity within a particular cell; furthermore, exactness can be obtained locally by interpolating potential and computing the force as its negated gradient, rather than directly interpolating the force. This can be achieved using a Hermite interpolation scheme[2]. Finally, divergence freedom can be added by using harmonic polynomials [12] or inner spherical harmonics. For such a scheme, cells domains can be taken as the circumscribing sphere around each cube cell of the cubetree algorithm and the potential represented within each cell.

Thus, a model that locally preserves all the desired structure of gravity is possible; these local properties are, however, not easily extended globally within the framework of the cubetree algorithm. Indeed, requirement (2) above implies that the local approximations will overlap or meet at the boundaries of their respective cells. The approximation functions between two neighboring domains will not necessarily agree on this overlap region. In the original cubetree algorithm, the overlap regions are the faces of the interpolation cube domains. The approximation function are interpolating polynomials which agree at shared interpolation nodes, but may not agree elsewhere on the faces. In particular, when 2 cubes are of different sizes, not all the interpolation nodes are common across a face and the discontinuity of the approximation is more accentuated. This discontinuity is clearly apparent in Figure 9 (a) which shows the approximation error in adjacent cells.

In particular, these discontinuities do not allow for derivatives to be computed in the overlap regions and thus do not allow for the computation of force as the derivative of a potential globally. Note however, that the jumps in discontinuity are within the overall error tolerance, that is, the variations in the approximations are bounded and small. As a result, if the interface region is of measure zero, as in the cubetree algorithm, the approximation function is locally integrable.

In summary, being based on an error tolerance requirement only, the cubetree algorithms does not preserve smoothness (continuity of the approximation and its derivatives), exactness or harmonicity of the force it approximates globally, but can represent such properties locally, almost everywhere.

III. Boundary Regularization

In this section we present a first method to overcome the lack of continuity or smoothness at the boundary of the cubetree algorithm. While this method can be coupled with a Hermite interpolation scheme to provide a continuous and exact method, it requires more demanding error tolerances and results in larger memory footprint. The method, however, may be adequate for some applications and is relatively simple to implement. Moreover, it clarifies the difficulties and necessary conditions for generating an exact scheme.

A. Boundary Matching

As mentioned previously, the cubetree algorithm is built such that the approximated force is within a given tolerance ϵ of the "true" value. In particular in an overlapping region, two approximations, $P_1(x)$ and $P_2(x)$ that approximate the given force or potential, f, within the set accuracy, coexist and each approximation is as good as the other. Linear combinations of these two approximations

[2]For the cubic cells of the original cubetree algorithm, a Hermite interpolation polynomial can be obtained as a tensor product of 1-dimensional Hermite polynomials, in the same way as the Lagrange interpolating polynomials used in [6] have been obtained from 1-dimensional Lagrange polynomials.

that satisfies the error requirement would also be equally fitted for the given objectives. Notably, combinations of the from $P_\lambda(x) = \lambda(x)\, P_1(x) + (1 - \lambda(x))\, P_2(x)$, with $\lambda(x) \in [0, 1]$ and varying from 0 to 1 can be shown to satisfy these requirements. Indeed, $|P_1 - f| \leq \epsilon$ and $|P_2 - f| \leq \epsilon$ implies that $|P_2 - P_1| \leq 2\epsilon$, and $|\lambda \leq 1|$, so that:

$$|P_\lambda(x) - f(x)| \leq |\lambda\,(P_1 - P_2) + P_2 - f| \leq \lambda\,|P_2 - P_1| + |P_2 - f| \leq 3\epsilon\,.$$

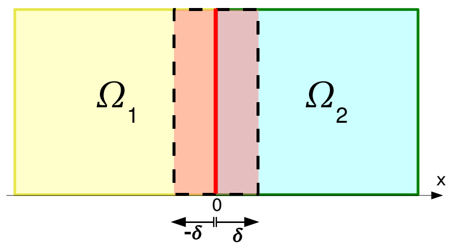
Since $P_\lambda(x)$ equals $P_1(x)$ for $\lambda = 1$ and $P_2(x)$ for $\lambda = 0$, this approximation can be used to effectively provide a transition between P_1 and P_2, while keeping the error under control.

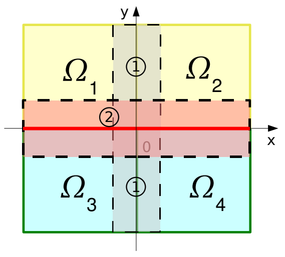
More precisely, denoting Ω_1 and Ω_2 two neighboring cubes in the cubetree approximation (with respective polynomial approximations P_1 and P_2, as shown in Figure 2(a)), we are thus led to the following method to obtain continuity across a boundary:

1. Extend the approximation P_1 to small layer of width δ across the face in the region Ω_2 and similarly for P_2.

2. Linearly match P_1 and P_2 by choosing $\lambda(x)$ to be zero on the layer boundary in Ω_1 and one on the layer boundary in Ω_2.

If the common face is chosen to correspond to the yz-plane of a local coordinate axis, as shown in Figure 2(a), then $\lambda(x)$ can be chosen, for example, to be the linear function $(x/\delta + 1)/2$ to ensure continuity. Smoothness can be ensured by taking $\lambda(x)$ to be smooth, as for example $(1/\pi)\arctan((x/\delta)/(1 - (x^2/\delta^2))) + 1/2$, and the norm in the above estimates to include some derivative information, such as a C^k norm.





a *b*

Figure 2: *a*): Approximation extension to a layer around a common face (in red). *b*): Matching a coordinate at a time.

Note that the boundary layer width must be chosen carefully so as to retain a desired accuracy. First, this extension can be done due to the fact that P_1 and P_2 are in fact defined in all of $\mathbb{R}^3$ (being polynomials), while the control of the accuracy with the layer width, δ comes from continuity of the polynomials. In particular, while $|P_i - f| \leq \epsilon$ in Ω_i, continuity (and compactness of the cubes) implies that we can choose a δ such that $|P_i - f| \leq 2\epsilon$ on $\Omega_i + \delta$[3], for example.

When the cubes meet at a corner, the above regularization can be applied one coordinate at a time. For example, considering the case of Figure 2(b), on $\Omega_\alpha = \Omega_1 \cup \Omega_2$, the above "face" regularization leads to the approximation $P_\alpha(x) = \lambda_\alpha(x)\, P_1(x) + (1 - \lambda_\alpha(x))\, P_2(x)$. Similarly, one obtains an approximation P_β on $\Omega_\beta = \Omega_3 \cup \Omega_4$. Then, one can apply the "face" regularization on the pair $(\Omega_\alpha, P_\alpha)$ and (Ω_β, P_β).

Since each "face regularization" requires a small extension and an associated loss of accuracy, the method requires a slightly better approximation inside the cube to obtain a uniform, continuous approximation across boundaries that still meet a set error tolerance. With the above estimates, the approximation would be within 5ϵ for a face and 17ϵ across a corner, thus requiring ϵ (the

[3]This set is defined as: $\Omega_i + \delta = \{x \in \mathbb{R}^n\ :\ |x - y| < \delta$ for some y in $\Omega_i\}$.

approximation error inside the cubes) to be an order of magnitude smaller than the desired maximum tolerance. This can be taken into account when building the model but results in a larger memory footprint of the model. However, once built, the cost of force evaluation is a small constant overhead that does not penalize the efficiency of the cubetree algorithm. This constant time overhead is associated with a test to determine if the state lies within a boundary layer and, if so, to evaluate the linear combination of P_1 and P_2.

Finally note that if no requirement is imposed on the derivative of the interpolating polynomials P_1 and P_2, the derivative may not satisfy good error properties and the interpolation of the potential with this modified cubetree algorithm would not lead to a good force approximation. On the other hand, using a Hermite interpolation scheme (or one based on harmonic polynomials, as discussed in the previous section), the above regularization offers a method to provide a smooth and exact scheme, albeit at the cost of a larger memory footprint[4]. The exactness is obtained by approximating the potential instead of the force directly. The forces are simply obtained by differentiation.

B. Partition of Unity

The above regularization can be thought of as applying a partition of unity with cube domains. Indeed, the boundary layer matching function, $\lambda(x)$, can be extended beyond the layer to be uniformly equal to 1 inside one of the cube and zero outside the cube augmented by the layer, and the sum of all the λ functions is equal to one at any point of space covered by the cubes. That is we can represent the uniformly continuous approximation as $U \simeq \sum_{\text{cubes } C} \lambda_C(x) P_C(x)$.

The partition of unity λ_C is not unique and other choices are possible and have been applied to the solution of partial differential equations [13, 14, 15]. In particular, the use of harmonic polynomials in conjunction with partition of unity showed faster convergence rate than classic finite element methods for solving Laplace equation [14].

The natural extension of the cubetree algorithm in terms of partition of unity emphasizes two fundamental properties that guide us in the following section. These are:

1. the local nature of the approximation scheme, and thus the reduction of the approximation to the right selection of the approximation function space: while the initial cubetree algorithm is based on interpolation to match given the actual force at selected points (a useful property for model update based on flight data, for example), the interpolation of a function and its derivative at selected points is not the most versatile function space to approximate a given function. Rather, approximation schemes based on integral norms between functions lead to more uniform approximations, as will be discussed in the next section.

2. the importance of the base domain, as opposed to its boundaries: having a covering associated with a partition of unity, the patching is done "automatically" with the associated partition of unity functions.

IV. Exact and Continuous Spline Approximation

While the method discussed in the previous section offers a smooth and exact scheme, it requires the error tolerance to be fixed to stricter standards than what is required by applications and results in larger models. In this section, we present a different approximation scheme that results in models of similar size as the original cubetree while preserving exactness and some degree of smoothness.

To create an exact approximation, we proceeds as previously by approximating the potential rather than the force. Taking the negated gradient of the approximated potential will provide force and exactness is automatic: $\hat{\boldsymbol{F}} := -\nabla \hat{U}$. To ensure smoothness to allow the approximated potential to be derived, we construct $\hat{U}$ in a function space that includes only functions with the desired

[4]the size of a cubetree model depends on the number of coefficient stored per cell and the overall number of cells. In the case of a Hermite interpolation, the degree of the interpolating polynomial is larger than that of a Lagrange polynomial –for equal errors– and thus results in a larger model.

continuity. Put more concretely, choose a function space S with spanning set $\mathcal{B}$ such that $\phi \in \mathcal{B}$ observes,

$$\phi(\boldsymbol{x}) \in C^1 ,$$
$$\phi(\boldsymbol{x}) \text{ and } \nabla\phi(\boldsymbol{x}) \text{ have compact support.}$$

An example of such a spanning set $\mathcal{B}$ with elements satisfying these properties is one generated by Basis Refinement of a B-spline basis.

A. Basis Refinement

Basis Refinement, or CHARMS [16], is an adaptive refinement framework developed for physical simulation. In the CHARMS framework adaptive refinement is performed on functions forming $\mathcal{B}$ – known as *scaling functions* – rather than the elements that describe the domain. Once the scaling functions are refined the domain elements are adapted to fit the needs of the scaling functions (i.e. for the purpose of integration). Figure 3 gives an example of refining degree 3 B-splines in 1D.

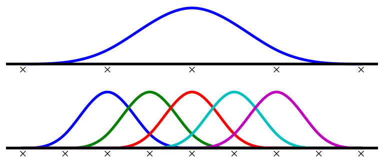


Figure 3: Refinement relation for a degree 3 B-spline. $\times$ below the axis indicate locations of knots. In this refinement every pair of knots in the original is split by a new knot half way between them. The result is a new set of B-splines that are translates and dilates of the original.

The *refinement relation* for scaling functions $\phi \in \mathcal{B}$ is the fundamental unit of adaptivity in CHARMS. A refinement relation gives a recipe for representing ϕ with a linear combination of dilated and translated versions of ϕ, known as ϕ's children $\mathcal{C}(\phi)$; see Figure 3. At the n^{th} iteration of refinement the spanning set $\mathcal{B}_n$ is constructed by replacing elements $\phi \in \mathcal{B}_{n-1}$ with $\mathcal{C}(\phi)$. Since linear combinations of $\mathcal{C}(\phi)$ can represent ϕ, linear combinations of $\mathcal{B}_n$ can represent $\mathcal{B}_{n-1}$. Thus, the approximation space of the n^{th} iteration S_n always contains the previous space: $S_n \supseteq S_{n-1}$. Note that this algorithm does not always produce a basis. For some applications – including this one – the linear independence property of a basis is not necessary. In this case $\mathcal{B}_n$ are merely spanning sets of the space S_n.

The advantage of CHARMS is that one does not need to develop machinery for handling discontinuities at T-junctions. T-junctions arise naturally in adaptive meshing. They get their name from their appearance in 2 dimensional quadrilateral meshes where it is a point in the interior of an edge where 3 quadrilaterals meet. T-junctions are seamlessly handled by the virtues of the spanning set: if its span does not contain discontinuous functions then no discontinuities can arise. Furthermore, as long as the refinement observes the continuity conditions no subsequent spanning set will permit discontinuities. Thus, CHARMS is the foundation upon which we build a continuous adaptive approximation.

Before going into more detail we must introduce our scaling function ϕ, the B-spline Basis function.

B. 1D B-spline Basis

Let $\tau := \{t_j\}$ be a nondecreasing sequence in $\mathbb{R}$ with N elements, t_0 to t_{N-1}, called the *knot vector*. The j^{th} *B-spline* of order k for the knot vector τ is designated $B_{j,k,\tau}$, and defined as,

$$B_{j,0,\tau}(x) := \begin{cases} 1 & \text{if } t_j \leq t < t_{j+1} \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

$$B_{j,k,\tau}(x) := \frac{t - t_j}{t_{j+k} - t_j} B_{j,k-1,\tau}(t) + \frac{t_{j+k+1} - t}{t_{j+k+1} - t_{j+1}} B_{j+1,k-1,\tau}(t)\,. \tag{2}$$

For B-splines to operate in the CHARMS framework they must observe a refinement relation. Not all B-splines observe a refinement relation, however, one common case does: τ with uniform knot spacing and knot multiplicity 1. To refine B_τ add a knot between every pair of knots in τ; more precisely, insert $(t_{j+i} + t_{j+i+1})/2$ after t_{j+i} in τ for $i \in 0, 1, \ldots, k$; see Figure 3.

C. B-spline Basis in K-Dimensions

The K-dimensional tensor product B-spline $B_{\boldsymbol{\tau}}$ is defined by the product of K 1-dimensional B-splines – each with an individual knot vector – where each spline handles one dimension:

$$B_{\boldsymbol{j},k,\boldsymbol{\tau}}(\boldsymbol{x}) = \prod_{d=1}^{K} B_{j_d,k,\tau_d}(x_d)\ \ . \tag{3}$$

In equation (3) $\boldsymbol{\tau}$ is simply a list of knot vectors, one for each dimension, and τ_d is the knot vector for dimension d. Similarly, $\boldsymbol{j}$ is a multi-index where j_d specifies using the j_d^{th} B-spline of τ_d for dimension d. For example, in 3 dimensions equation (3) is

$$B_{\boldsymbol{j},k,\boldsymbol{\tau}}(\boldsymbol{x}) = B_{j_1,k,\tau_1}(x_1) B_{j_2,k,\tau_2}(x_2) B_{j_3,k,\tau_3}(x_3)\,.$$

Refinement in multiple dimensions treats each dimension independently; and in our case we refine each dimension by halving the distance between knots in the knot vector, as described at the end of B. We denote subsequent spaces of refined K-dimensional B-splines with the S_n and $\mathcal{B}_n$ notation.

Now we can define the scaling functions ϕ which constitute a spanning set $\mathcal{B}_n$ for an adaptive approximation of gravitation. Let ϕ be a 3-dimensional B-spline of degree $k = 3$ with uniformly spaced knots of multiplicity 1. Modeling potential with $\sum_i \alpha_i \phi_i$ means force is modeled by $\sum_i \alpha_i \nabla \phi_i$. Thus, the modeled force $\hat{\boldsymbol{F}}$, having lost one degree of continuity from differentiation, is C^1 and exact.

D. Linear Independence of $\mathcal{B}_n$

Refinement by substitution with high order B-splines does not guarantee linear independence of the spanning set $\mathcal{B}_n$ [16, 17]. Even in 1 dimension simple refinements can lead to linear dependence in $\mathcal{B}_n$. Figure 4 gives several 1-dimensional examples.

As Figure 4 indicates, the tricky cases are between different levels of refinement. Linear independence between levels can be guaranteed with additional bookkeeping [17], but for function approximation the extra effort is not necessary. In fact, all we need for function approximation is linear independence within each level of $\mathcal{B}_n$. Within a level the only difficulty is to capture the repeated children between nearby B-splines. As Figure 4, (a) (compare the second and third row) shows, the adjacent B-splines contribute to the children of one another. The simplest way to avoid over adding a given child is to record the included scaling functions of each level of $\mathcal{B}_n$ with a bitmap or set [16]. The problem with the bitmap approach is that it is not adaptive: every function, included or otherwise, is explicitly represented. A set is adaptive, but in practice a set data structure is too inefficient to store individual scaling functions. [5] We will revisit this problem in Section IV.G.

The following sections describe an approximation method that does not require linear independence between the levels of $\mathcal{B}_n$.

[5]For example, the C++ Standard Template Library set has 24 bytes of overhead per function (3 pointers). In our models that would cost about 860 MB of overhead. Given the functions themselves cost about 860 MB (3-dimensional location) we would waste over 1.5 GB.

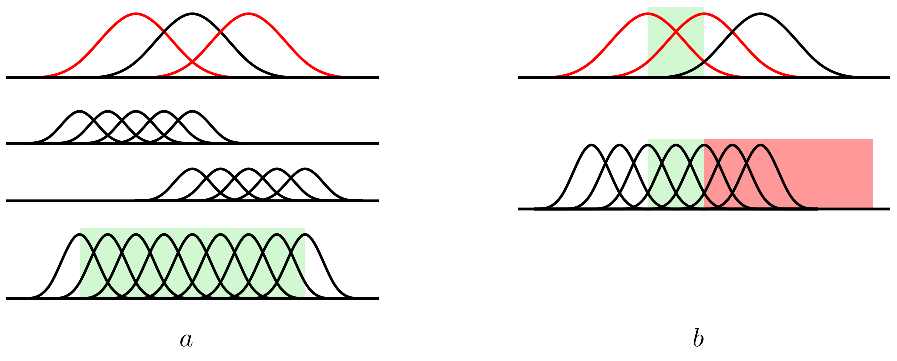


Figure 4: Examples where refinement does not produce a basis. In each diagram the top row shows $\mathcal{B}_0$; red B-splines are to be refined and black B-splines are not. *a*) The second and third rows are the left and right refinements respectively, and the final row shows the combination of these refinements. Green shading indicates the support of the middle scaling function in $\mathcal{B}_0$, notice the shaded region is completely covered by a basis capable of representing the unrefined B-spline. *b*) This example shows that restricting the domain of the approximation can make otherwise linearly independent examples, linearly dependent. The green shaded region is the domain of the approximation. The refinement of the red functions would not normally completely represent the black function because the children of the red functions do not cover the support of the black function – which is depicted as the union of red and green shaded regions. In this case, however, the restriction to the green region means that they do.

E. Least Squares Approximation

Let $U : [\boldsymbol{s}, \boldsymbol{t}] \subset \mathbb{R}^3 \to \mathbb{R}$ and pick a basis $\mathcal{B} = \{\phi_j\}$. A least squares approximation chooses coefficients to minimize the distance between U and its approximation in $\mathcal{B}$ given some norm. In many applications – particularly scattered data interpolation [18] – a discrete norm is made by the collected data to be fit [19]. In our application data points can be placed where ever convenient. This gives us the flexibility to pick any norm we want, e.g. L^2 or H^1. Before going further let us formalize the minimization problem.

Keep U and $\mathcal{B}$ as described above and pick an inner product space in which we wish to minimize $\|U - \sum_j \alpha_j \phi_j\|$, or equivalently

$$\left\|U - \sum_j \alpha_j \phi_j\right\|^2, \tag{4}$$

where $\| \cdot \|$ is the norm induced by the the inner product $\langle \cdot\, , \cdot \rangle$. By differentiating the quantity in (4) above and setting the result to zero we produce a linear system, $Mx = b$, that solves the minimization, where

$$M_{i,j} = \langle \phi_i, \phi_j \rangle \,, \tag{5}$$

$$b_i = \langle U, \phi_i \rangle \,. \tag{6}$$

In this linear system M is often called a *mass matrix.* The next natural question is with what norm shall we minimize $\|U - \sum_j \alpha_j \phi_j\|$. For our application we wish to approximate both U and $\boldsymbol{F}$; thus, the most obvious choice is the H^1 norm,

$$\langle f, g \rangle = \int_\Omega f(\boldsymbol{x})g(\boldsymbol{x}) + \sum_i \frac{\partial}{\partial x_i} f(\boldsymbol{x}) \frac{\partial}{\partial x_i} g(\boldsymbol{x}) d\boldsymbol{x} \,, \tag{7}$$

$$\|f\| = \sqrt{\langle f, f \rangle} \,, \tag{8}$$

as it incorporates both gravitational potential and force.

Implementing either inner product requires numerical integration. Our approximation uses cubic polynomials; thus $\langle \phi_i, \phi_j \rangle$, i.e. creating the mass matrix, will necessitate integrating degree 6 polynomials. A degree 6 polynomial is exactly integrated with Gauss-Legendre quadrature of order 4 [20], making it a natural choice for creating the mass matrix. The right hand side, $\langle U, \phi_i \rangle$, will also use order 4 Gauss-Legendre quadrature. We justify this decision with our goal: to approximate gravitation with cubic polynomials. Therefore, order 4 Gauss-Legendre quadrature will be accurate for the right hand side at the same time as the approximation is accurate. Finally, the integration domain, Ω, is split into a hexahedral mesh that mimics the break points of the piecewise polynomials defining ϕ_i. Whenever ϕ_i overlaps with another scaling function from a finer level, ϕ_j, the hexahedral mesh follows the break points of ϕ_j. This hexahedral mesh defines the quadrature domains over which we evaluate equations (5) and (6).

F. Solving $\mathcal{B}_n$ with Linear Dependence

This process works very well for tensor product B-splines, but will fail once our refinement process begins as the mass matrix is singular when $\mathcal{B}_n$ has linear dependence. To accommodate linear dependence between levels of $\mathcal{B}_n$ we will solve $\mathcal{B}_n$ hierarchically: Each $\mathcal{B}_n$ is formed of nested tiers $\mathcal{T}^p$ for $0 \leq p \leq n$. They are nested in the following sense: $\mathcal{T}^0$ is the coarsest and contains exactly the scaling functions necessary to cover the domain Ω. $\mathcal{T}^p$ contains scaling functions from the p^{th} level that are refinements of scaling functions unique to $\mathcal{T}^{p-1}$, and anything from $\mathcal{T}^{p-1}$ that is not refined. In other words,

$$\phi \in \mathcal{T}^p \implies \phi \in \mathcal{T}^{p-1} \vee (\exists \phi' \in \mathcal{T}^{p-1} \wedge \phi \in \mathcal{C}(\phi') \wedge \phi' \notin \mathcal{T}^{p-2}),$$

where $\mathcal{T}^{-1} := \emptyset$. Although $\mathcal{B}_n$ is defined similarly to $\mathcal{T}^p$, they are not the same. In $\mathcal{B}_n$ there is no restriction on which scaling functions from $\mathcal{B}_{n-1}$ are refined; however, $\mathcal{T}^p$ may only refine $\phi \in \mathcal{T}^{p-1}$ that have already been refined $p-1$ times. Figure 5 gives a 1-dimensional example, and demonstrates the difference between $\mathcal{B}_n$ and $\mathcal{T}^p$.

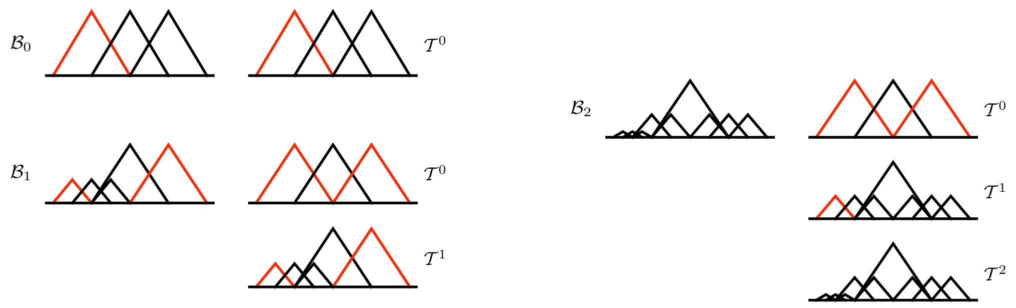


Figure 5: Examples of $\mathcal{T}^p$ that constitute $\mathcal{B}_n$. Red B-splines are to be refined at the next tier. Notice the difference between $\mathcal{T}^1$ and $\mathcal{B}_1$ in the third iteration. This demonstrates that $\mathcal{B}_n$ may refine any scaling function from $\mathcal{B}_{n-1}$, but $\mathcal{T}^p$ only refines those scaling functions unique to $\mathcal{T}^{p-1}$.

To find coefficients for $\mathcal{B}_n$ we iterate through $\mathcal{T}^p$, solving one tier at a time. First we solve $\mathcal{T}^0$ using equations (5) and (6). To solve $\mathcal{T}^p$ assume we know the coefficients of $\mathcal{T}^{p-1}$. Then separate $\mathcal{T}^p$ into $\mathcal{N}$ and $\mathcal{R}$ such that,

$$\begin{aligned} \phi_j \in \mathcal{N} &\implies \phi_j \in \mathcal{T}^{p-1}, \\ \phi_j \in \mathcal{R} &\implies \exists \phi' \in \mathcal{T}^{p-1} \wedge \phi_j \in \mathcal{C}(\phi'). \end{aligned}$$

In other words, $\mathcal{R}$ are scaling functions that come from refining $\mathcal{T}^{p-1}$, and $\mathcal{N}$ are the scaling functions we carry without refinement from one tier to the next. Organize $\phi_j \in \mathcal{T}^p$ such that $\phi_j \in \mathcal{R}$ when $j < |\mathcal{R}|$ and $\phi_j \in \mathcal{N}$ otherwise, i.e. the first $|\mathcal{R}|$ scaling functions of $\mathcal{T}^p$ are the new scaling functions, and the rest are from the previous tier, $\mathcal{T}^{p-1}$. Next, create the mass matrix M of $\mathcal{T}^p$ as usual, equations (5); and remove the scaling functions that come from earlier levels by taking their coefficients from the solution to the previous tier, $\mathcal{T}^{p-1}$. To do this remove the last $|\mathcal{N}|$ rows of M and b, as these equations have already been solved in the in $\mathcal{T}^{p-1}$. This leaves us with under determined system $\bar{M}'x = \bar{b}'$. Using the coefficients from $\mathcal{T}^{p-1}$, we remove the last scaling functions in $\mathcal{N}$ by subtracting their contribution from the right hand side; thus, creating $\bar{b}$. Finally, we solve $\bar{M}x^p = \bar{b}$. Expressed as equations the process is:

$$\begin{aligned}
&1)\ \bar{M}'_{i,j} = M_{i,j} \text{ for } i < |\mathcal{R}|, j < |\mathcal{T}^p|\,,\\
&2)\ \bar{b}_i = b_i - \sum_j \bar{M}'_{i,j} x^{p-1} \text{ for } i < |\mathcal{T}^p|, j < |\mathcal{T}^p|\,,\\
&3)\ \bar{M}_{i,j} = \bar{M}'_{i,j} \text{ for } i < |\mathcal{R}|, j < |\mathcal{R}|\,,\\
&4)\ x^p = \bar{M}^{-1}\bar{b}\,,
\end{aligned}$$

where x^{p-1} is a vector of zeros for the first $|\mathcal{R}|$ entries and coefficients gleaned from solving $\mathcal{T}^{p-1}$ for the last $|\mathcal{N}|$ entries. Finally, the coefficients for $\mathcal{T}^p$ are created by replacing the $|\mathcal{R}|$ zeros in x^{p-1} by x^p. This process is continued until every tier $\mathcal{T}^p$ of $\mathcal{B}_n$ is solved, the coefficients of $\mathcal{T}^{P-1}$ become the coefficients for $\mathcal{B}_n$.

It is worth noting that not only does this process permit linear dependence among the levels of $\mathcal{B}_n$, it also reduces the size of any given linear system solve: In practice we never need the last $|\mathcal{N}|$ rows of M, so they should never be created. This savings can as much as halve the total size of the mass matrix being solved, which is not insignificant. For example, the final iteration of our model required 69 GB, doubling this makes for a very large matrix indeed.

G. Implementation Details

In the subsequent sections we describe implementation details that allow our model to efficiently represent small body gravitation.

G.1. Representation of $\mathcal{B}_n$

This section focuses on representing $\mathcal{B}_n$ in a way suitable to preparing and using our model of small body gravitation. Two concerns dominate the design: memory efficiency during model creation and computation efficiency during force reconstruction. The former is the primary bottleneck of our coefficient fitting stage and dictates the number of iterations of refinement the algorithm will permit. Ideally as much memory as possible is given to the mass matrix, meaning our representation of $\mathcal{B}_n$ should be as lean as possible. The latter is the whole reason for doing this in the first place! Fortunately both problems are addressed by the same data structure.

Our implementation uses patches of B-splines with uniform knot spacing. Each patch $\mathcal{P}_i$ is constructed by a tensor product B-spline representing many B-splines, $\mathcal{P}_i := \{B_{j,3,\boldsymbol{\tau}}\}$, with a single data structure. In this way we can represent any number of B-splines in a cuboid layout with only three data points: the lower bound of the support of the patch, the upper bound of the support of the patch, and the knot spacing for all B-splines of the patch.

Representing $\mathcal{B}_n$ this way has two advantages. First, it makes representing $\mathcal{B}_n$ very compact. For example, the model developed in Section IV.I.1 requires 1.5 MB of storage to represent 18 million B-splines. Our second advantage is in evaluating $\mathcal{B}_n$. The most efficient B-spline evaluation procedures compute neighboring B-splines of a knot vector simultaneously [19]. Using the B-spline patch model captures redundant computation of neighboring ϕ_i whenever they belong to the same patch.

G.2. Refining Patches

Refining a patch is, semantically, the same as refining each B-spline within the patch individually. As each patch is represented by a tensor product B-spline, the new patch is constructed by inserting an additional knot between the knots forming each knot vector of the tensor product. In terms of our data structure this amounts to dividing the `knot_distance` by 2. However, just as with individual scaling functions of Figure 4 (a), some patches will share children. To accommodate redundancy in the children patches must be resized to avoid each other, we call this process *patch splitting*. Consider the 2-dimensional example in Figure 6. From $a \rightarrow b$ two adjacent patches are refined; oblivious to one another, they generate redundant children. To extinguish the repeated children one patch is selected to avoid the other, e.g. $b \rightarrow c$. The number of additional patches created by this process – something we wish to keep low – is reduced by always splitting the patch that contains fewer scaling functions.[6]

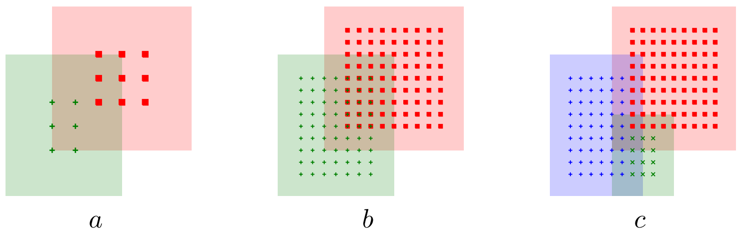


Figure 6: The patch splitting process. Shaded regions show the support of each patch, and markers show the peaks of the scaling functions. At the start, (a), there are two patches – red and green. Each patch is refined such by replacing the present scaling functions with their children, producing (b). A 3×5 block of scaling functions are redundant, so one patch is split to avoid the redundant region, producing the three patches in diagrammed in (c).

Therefore, our algorithm for refinement is: i) Loop through patches and refine those that need refinement. ii) Pair-wise compare patches to avoid overlapping refinements using patch splitting. The pair-wise comparison may be efficiently implemented with a octree data structure [21], but in our experience the cost of the straightforward $O(N^2)$ pair-wise comparisons is small compared to, solving a matrix with 20 million unknowns.

G.3. Computing Gravity with $\mathcal{B}_n$

We have already spoken to the favorable efficiency of evaluating the B-splines of $\mathcal{B}_n$ with patches, but we still need a mechanism for finding the patches that affect a given query. For this we will use a bounding volume hierarchy (BVH) [22].

Bounding volume hierarchies are tree data structures for organizing objects with spatial extent. In this sense they fill the same roll as octrees, but there is an important difference: While octrees subdivide *space*, a BVH subdivides *groups of objects*. Two recursive operations characterize a BVH: `Build` and `Find`. `Build` takes as input a list L of objects and creates one node.

`BoundingVolume` finds the upper and lower bounds of the objects in L. `Split` divides L into two sublists L_ℓ and L_r, which form the input to the next level of the tree. A poor split will mean the bounding volumes of the subsequent levels do not shrink rapidly, and result in poor `Find` performance. Our `Split` sorts L by the widest dimension of its bounding volume; then L_ℓ and L_r split the sorted L evenly. Figure 7 diagrams the process of building a BVH.

`Find` recalls all the objects containing a point query by recursively traversing the BVH. At each level `Find` recursively follows all children that contain the query point. When `Find` reaches a leaf the object contained is appended to the output. Following the `Split` described above, the tree will always be balanced; hence, most queries will only require $O(\lg N)$ effort. Note that the running time will depend on the query and how the bounding volumes are organized within the tree. For example, a perfectly bad `Split` can force `Find` to visit every node in the BVH.

[6]This heuristic was found to globally produce fewer patches than always splitting the patch that locally creates fewer patches.

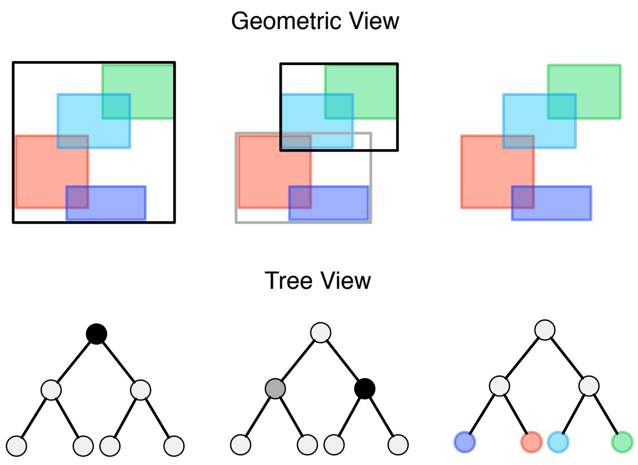


Figure 7: A bounding volume hierarchy. The objects organized are the colored boxes. On the left is a geometric view. The first level shows the bounding box in black, the second level shows it in black and gray, and the last level only the underlying objects remain. On the right is a tree view of the bounding boxes.

To evaluate $\mathcal{B}_n$ efficiently its constituent patches are organized in a BVH. The bounding volume of a patch $\mathcal{P}$ is the union of the supports of ϕ_i (i.e., where it is nonzero) for $\phi_i \in \mathcal{P}$. `Find` retrieves the relevant patches with the BVH and each patch is evaluated.

H. Model Creation

This section concludes the description of model creation.

H.1. Domain Representation and Creating $\mathcal{B}_0$

The domain of our approximation, Ω, plays three roles during model creation: *i*) It provides the domain over which we evaluate our inner product (equation (7)); *ii*) seeds $\mathcal{B}_0$ with B-spline patches necessary to cover Ω; and *iii*) it defines the regions over which we will measure the error in our approximation. The simplest solution is to represent the region around an asteroid as a cube, but this needlessly includes the interior of the asteroid. To avoid as much of the interior as possible we use an octree, and `Subdivide` whenever a cuboid region intersects the asteroid. To avoid subdividing forever a lower bound on the cuboid size is provided.

Once a domain octree Ω has been constructed $\mathcal{B}_0$ is created to cover Ω. Each octree cuboid is considered in isolation and a patch is placed to cover it with a basis of B-splines over Ω. This creates redundant B-splines near the boundaries of octree cells, which are removed using patch splitting. See Figures 6 (*b*) and (*c*) and Figure 8.

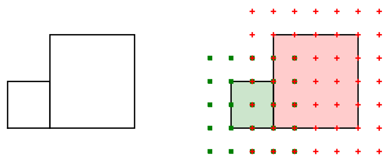


Figure 8: Detailed view of initializing $\mathcal{B}_0$. Left: Ω is the interior of the two boxes. Right: Patches are placed so the domain of patches covers the cuboid it came from. This creates redundancies, which the overlapping markers (crosses and squares) show. Shaded regions show the domain of each patch.

H.2. Error Estimation and Creating $\mathcal{B}_{n+1}$

To estimate relative error we sample the octree domain Ω. Within each cuboid $\omega \in \Omega$ the error is randomly sampled at a rate of one sample every 10 m^3. Any time the relative error of a sample in ω exceeds some threshold every patch intersecting ω is marked for refinement. Any time no samples in ω exceed the threshold ω is removed from the list of domains to check at the next iteration. At the end of the error checking process all patches requiring refinement are refined and patch splitting keeps them from producing redundant scaling functions.

I. Numerical Experiments with 1998 ML14

Now we turn our attention to approximating the gravitational potential and gravitational force of 1998 ML14.

I.1. Notes on Model Creation

The domain of our approximations began at (-1250, -1250, -1250) m and extended 2500 m in each direction, where the origin is the center of mass of the asteroid model. (As a point of reference the radius of 1998 ML14 is $\approx$ 500 m.) The smallest allowed cube in the octree domain was $2500/2^7 \approx$ 19.5 m to an edge. We also limited the largest cube to $2500/2^4 \approx 156$ m to an edge. The relative error threshold for refinement was set to 5×10^{-7}, which is well beneath the requirement for mission design [23]. The Conjugate Gradient solver's relative convergence tolerance [24] was set to 10^{-16}. This model was created in 4500 CPU hours on a parallel computer using the Message Passing Interface (MPI) [11] (64 processors for approximately 80 hours) and occupies 210 MB of memory. $\mathcal{B}_0$ is composed exclusively of patches with knot spacing $2500/2^7$ m, $\mathcal{B}_2$ ends the iterations with a mix of knot spacings at $2500/2^9$ m to $2500/2^7$ m. To cover the region outside the octree spherical harmonics of degree and order 12 were employed. We name this model the Patches of Uniform B-splines Tree, or pubtree for short.

I.2. Continuity at T-junctions

Now we test continuity between levels of refinement. In Figure 9 (a) we plot the error in the cubetree's force at a T-junction; the discontinuity is readily apparent at 935 m. Compare this to the same plot for a T-junction in our B-spline model, Figure 9 (b). No discontinuities exist, in fact the only evidence of a T-junction is a slight shift in the oscillatory pattern at 770 m.

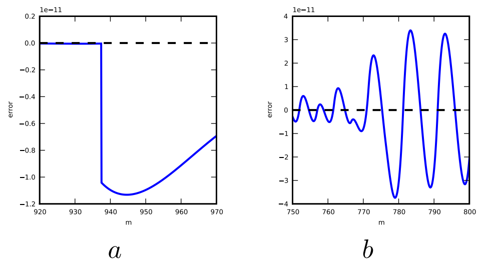


Figure 9: Continuity comparison of cubetree and B-spline method. In each case the absolute error in the x-component of the force is measured along a 50 m ray. The dashed line at $y = 0$ represents the actual force. As the T-junctions do not occur in the same places for each model, different rays are measured. The left graph shows cubetree, and the right graph shows B-spline method.

I.3. Error Measurements

This section explores the structure of refinement by plotting the relative error in potential and force. Portions of the domain are examined in Figure 10, which plot relative error in gravitational potential and gravitational force along the $z = 0$ plane. Note that $\hat{U}$ is about three orders of magnitude more effective than $\hat{\boldsymbol{F}}$. Therefore, for applications where an approximation to potential is not needed a direct approximation to force (as in cubetree) may be easier to create. A white band, extending approximately 60 m from the surface, is visible near the interface of the asteroid and space. This band is where the force approximation fails to make 10^{-6}. The same band for 10^{-5} error it is 40 m from the surface. Future improvements to patch refinement may shrink the band further, though we feel comfortable with accuracy up to 40 m from the surface.

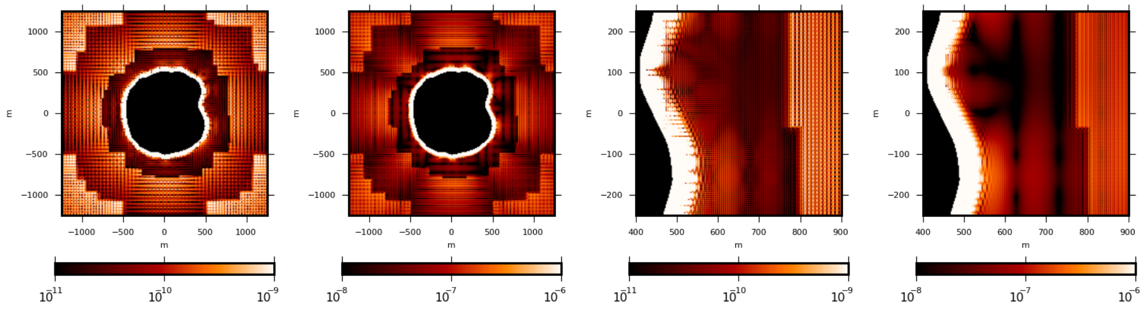


Figure 10: Relative error of pubtree across the complete domain. The interior of the asteroid is not measured and left black. Error is plotted on a log scale. Going from left to right: Error across the entire domain in potential. Error across the entire domain in force. Error near the surface in potential. Error near the surface in force.

Compared to cubetree this model appears to be less accurate despite using similar error bounds. Figure 11 plots cubetree force's relative error in the same region as Figure 10. This effect comes from the granularity and convergence rate of the cubetree compared to the pubtree: In the cubetree model the coarsest refinement is 312.5 m and contains tensor product degree 6 polynomials; pubtree's coarsest refinement is only 156 m and contains tensor products of degree 3 piecewise-polynomials. Therefore, not only does a single refinement of cubetree affect a bigger region, it also has a higher rate of convergence – especially away from the boundary where gravity is smooth. For example, Compare the plots of Figure 9, both plots show the absolute error on either side of a refinement. Pubtree approximately halves the error across the barrier whereas cubetree error falls about one order of magnitude.

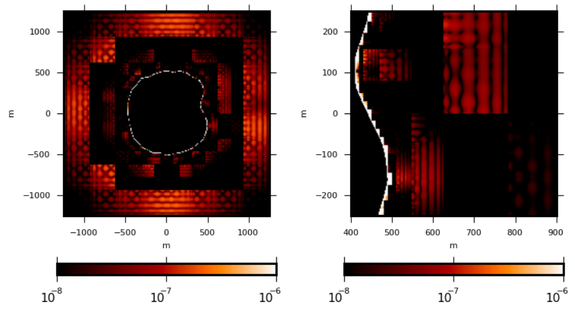


Figure 11: Relative error in the force produced by cubetree. The interior of the asteroid is not measured and left black. Error is plotted on a log scale. Left: The error over the entire domain; Right: Sample of the error in a region abutting the asteroid; compare these to Figure 10.

Cubetree's aggressiveness means a direct comparison between pubtree and cubetree in terms of memory footprint and speed is not entirely fair. Also, quantitative comparisons between trajectories of the models will tend to favor cubetree; as we will see in the following section. Of course quantitative improvements over cubetree was not our goal. Furthermore, both models easily achieve the original goal of 10^{-5}.

I.4. Refinement Sequence

Figure 12 shows the model is develop over 3 iterations. After each iteration regions with high error are reduced. For example, the band of white surrounding the asteroid, which shrinks after each iteration.

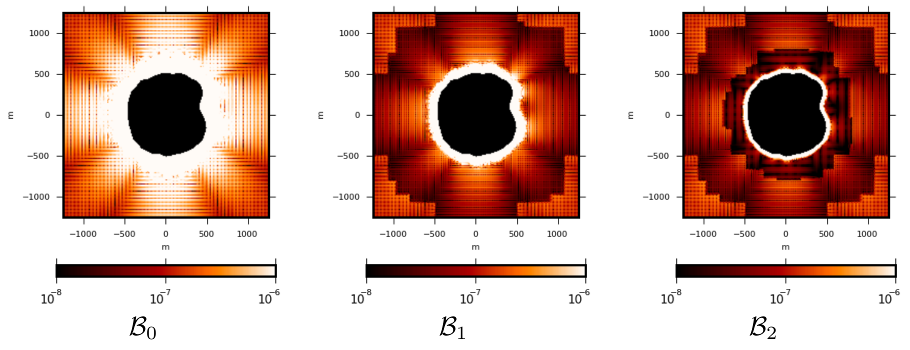


Figure 12: Relative error over the sequence of refinement. The interior of the asteroid is not measured and left black. Error is plotted on a log scale.

I.5. Trajectory Integrations

Our trajectory simulations use the same models as the ones from [6]. In addition we simulate using the pubtree model:

Pubtree trajectories are generated using the pubtree force model. Integration is done with Embedded Runge-Kutta Prince-Dormand (order 8,9) method using relative error tolerance 10^{-13} and absolute error tolerance 10^{-6}.

We integrated a family of retrograde orbits starting between 600 m and 1000 m from the asteroid center. Initial positions were chosen close to the equatorial plane, and initial velocities did not contain large components outside the plane. The magnitude of the velocity was clamped to within 0.65 and 0.75 of escape speed. Simulations ran for 5 days of ballistic motion with each model (pubtree, cubetree, augmented polyhedral, reference), where impacting trajectories were thrown out. The position and velocity of the orbiter was recorded every 5 minutes of simulated time.

This experiment was repeated for 993 trajectories. For each trajectory we measured the maximum difference in position and velocity between the cubetree trajectory and reference trajectory. Figure 13 is a histogram of the errors in position and velocity. For comparison sake the same histograms for cubetree are presented in Figure 13. Cubetree's superior relative error is evident in the histograms, however, the shape of the histograms are similar. On average pubtree trajectories were 270× faster than the augmented polyhedral, compared to 301× for cubetree.

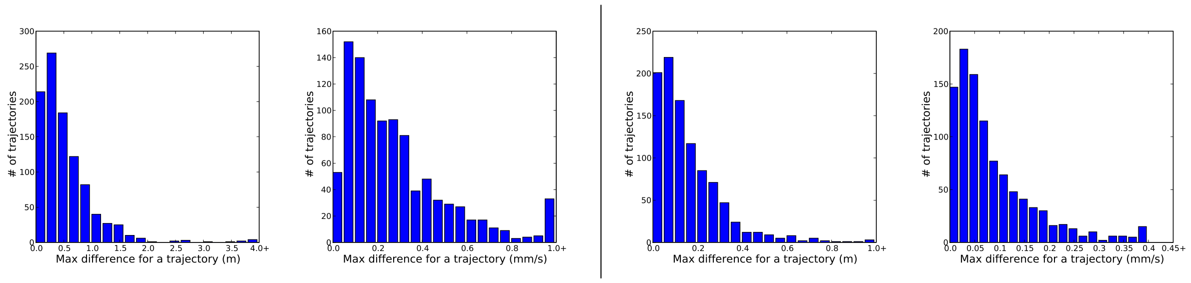


Figure 13: Histograms of errors in position and velocity for 993 retrograde pubtree trajectories integrated for 5 days and observed at 5 minute intervals. Left two are pubtree, right two are cubetree.

J. Discussion

The CHARMS framework, coupled with our hierarchical coefficient fitting, has produced an adaptive approximation to gravitational potential and force. The force is exact and continuous, and the accuracy is within the threshold for the vast majority of the domain.

J.1. Trajectory Integration

During the trajectory integration experiments we found pubtree was approximately $0.9\times$ the speed of cubetree. However, single evaluation tests actually estimate the speed at $2\times$ cubetree! The discrepancy comes from the integrator's adaptive time stepping strategy: On average pubtree trajectories required $1.8\times$ more samples than cubetree. We believe this comes from the higher amplitude of the oscillations in pubtree's force model; see Figure 9. Reducing the amplitude by tightening the error threshold would resolve this, thereby increasing the speed of pubtree.

J.2. Patch Refinement

As we have described it, pubtree employs patch refinement: if the error is bad in one part of the patch the whole patch is refined. This approach becomes problematic once the regions with high error are smaller than the patches. In this case every iteration will require a complete patch to be refined, effectively destroying adaptivity. To wit, our iterations of pubtree ended exactly for this reason.

The solution to this problem is sub-patch refinement implemented with patch splitting and hierarchical error estimation. First, error estimation should follow an octree structure: Each time a cuboid of the error octree is found to contain error it should be subdivided before being checked again. In this way we localize the error. Then, instead of refining whole patches we split the patch so only its scaling functions affecting the localized error are refined. Following this algorithm would reintroduce adaptivity and permit further iterations on pubtree.

J.3. Linear Independence

Although we have pursued a linearly dependent spanning set $\mathcal{B}_n$, with sufficient implementation effort a linearly independent basis could also be devised. A comparison between the methods would be a valuable analysis.

V. Divergence free approximations and challenges in preserving harmonicity

A divergence free approximation of the force can be obtained by projecting an approximate vector field onto the space of divergence free fields. According to the Helmholtz decomposition a vector field in a simply-connected domain can be decomposed into a divergence free and a curl free part:

$$F = F_{\text{div}} + F_{\text{curl}} \quad \text{with} \quad \nabla \cdot F_{\text{div}} = 0 \text{ and } \nabla \times F_{\text{curl}} = 0\,.$$

The presence of holes and handles in a domain results in a non zero harmonic part in the above decomposition. This more general result is called the Hodge (or Helmholtz-Hodge) decomposition [25]. Such decompositions are commonly used in computational incompressible fluid mechanics [26]. Several methods have been proposed in the literature, such as [27, 28, 29], so we do not consider the analysis of divergence free approximations alone.

Note however, that most of these schemes apply only to forces (vector fields) and do not allow a priori for a modification of the potential that would then generate a divergence free field upon differentiation. In Reference [28] the authors do compute the potential for the divergence free part but the techniques as presented there work only for piecewise constant vector fields. In effect, the continuous and exact approximation schemes described in this paper and the work existing on divergence free projections present significant challenges in merging.

To better understand this remark, let us comment on some classic results about harmonic functions [12]. First, note that a force deriving from a potential which is also divergence free satisfies Laplace's equation. That is, the potential is a harmonic function, and is thus analytic and each of its derivative is also an analytic function. Thus an approximation scheme aiming at being both exact and divergence free must provide an analytic approximation.

More importantly, the solution of Laplace equation is unique given continuous data on the boundary of a bounded region with Lipschitz boundary. For example, within the cubetree framework, the knowledge of the function on the boundary of the cubes would imply that the approximated function be completely determined inside the cubes. In particular, this leaves no freedom in matching the derivatives at the boundary. Unless the boundary data of each cube can be prescribed globally, two adjacent cubes sharing a face will only be continuous and not C^1 across the common face.

To illustrate these difficulties in the case of the boundary matching method, let us consider the setting of Section III, while assuming that the approximation polynomials P_i are harmonic. The blending of the functions P_1 and P_2 involves products of functions of the form: $\lambda(x)\,P_i(x)$. The Laplacian of these products is given as:

$$\Delta(\lambda P) = \lambda \Delta P + P \Delta \lambda + 2(\nabla \lambda \cdot \nabla P)\,.$$

Even if one chooses λ to be harmonic, so that the first two terms in the above right hand side vanish, the remaining term imposes a constraint on the derivative, which we have seen is not free to choose. A straight forward way to set all the terms to zero consists of taking λ to be a linear function. In that case however, the smoothness is lost and only a continuous approximation is obtained. Note that while the partition of unity method has been used with harmonic polynomials in Reference [14], the concern was about convergence – that is the measure of the error performed when compared to a true solution – rather than the harmonic properties of the approximation. In particular, these methods, as well as finite elements methods, only consider the problem in its variational formulation, thus loosing some of the smoothness of the problem.

The above properties illustrate the basic fact that being harmonic in a small region has some global implications. In effect, approximating the potential by harmonic functions would consist in solving exactly Laplace equation, which indicates the close link between preserving harmonicity and solving *exactly* Laplace equation for non-convex sets. These remarks in particular indicate the close link between function representation and the numerical solution of partial differential equations. The regularization of the cubetree algorithm, for example, was initially devised independently of the partition of unity method before realizing its equivalence to the construction of a partition of unity used in solving partial differential equations, while the spline approximation has been motivated by higher order finite elements methods.

VI. Conclusions

We have described the continuous and exact approximations of gravitational forces in the neighborhood of small bodies. These approximations preserve some degree of smoothness and were obtained in the setting of adaptive piecewise polynomial approximation initially considered in the cubetree method. While adaptive schemes are necessary for force approximation near small bodies, it has been shown that the complexity of imposing an extra structure on such schemes may, in fact, not be worthwhile for spacecraft applications due to the short integration time span generally encountered. In particular, the distinction between small discontinuities compared to small oscillations in the error become subversive as the error tolerance is tightened.

The ability to represent exactness is however important from a theoretical viewpoint as it illustrates the close link between the force representation problem and the solution of the partial differential equation defining the potential. In particular, it has been shown how the linear dependence among the "basis" functions appearing when refinement patches are considered can be overcome with the introduction of hierarchical refinement; a contribution that may also be useful to higher order finite element methods.

Moreover, these methods open the way for setting a standard for small body close field representation similar to ephemeris models for planetary positions that offer flexibility in updating the models and provide fast force evaluation routines.

References

[1] Lee, S., Fink, W., Russell, R., Von Allmen, P., Petropoulos, A., and Terrile, R., "Evolutionary Computing for Low Thrust Navigation," in *Space 2005*, American Institute of Aeronautics and Astronautics, 2005, Paper AIAA 2005-6835.

[2] Russell, R., "Primer Vector Theory Applied to Global Low-Thrust Trade Studies," *Journal of Guidance, Control and Dynamics*, Vol. 30, No. 2, 2007, pp. 460–472.

[3] Russell, R., "Global Search for Planar and Three-Dimensional Periodic Orbits Near Europa," *Journal of the Astroanutical Sciences*, Vol. 54, No. 2, 2006.

[4] Villac, B., "Using FLI maps for Preliminary Spacecraft Trajectory Design in Multi-Body Environments," *Celestial Mechanics and Dynamics Astronomy*, 2008, submitted.

[5] Gurfil, P., "Simple Satellite Orbit Propagator," in *AAS/AIAA Space Flight Mechanics Meeting, Galveston, Texas*, American Astronautical Society and American Institute of Aeronautics and Astronautics, 2008, Paper AAS 08-167.

[6] Colombi, A., Hirani, A. N., and Villac, B. F., "Adaptive Gravitational Force Representation for Fast Trajectory Propagation Near Small Bodies," *Journal of Guidance, Control and Dynamics*, Vol. 31, No. 4, 2008, pp. 1041–1051, doi:10.2514/1.32559, a publication of the American Institute of Aeronautics and Astronautics.

[7] Fetecau, R. C., Marsden, J. E., and West, M., "Variational Multisymplectic Formulations of Nonsmooth Continuum Mechanics," in *Perspectives and Problems in Nonlinear Science*, edited by E. Kaplan, J. E. Marsden, and K. R. Sreenivasan, Springer-Verlag, 2003.

[8] Karniadakis, G. E., and Sherwin, S., *Spectral/hp Element Methods for Computational Fluid Dynamics*, chap. 2 and 3, Oxford Science Publications, 2005, pp. 15–95.

[9] Werner, R. A., "The Gravitational Potential of a Homogeneous Polyhedron or Don't Cut Corners," *Celestial Mechanics and Dynamical Astronomy*, Vol. 59, No. 3, 1994, pp. 253–278, doi: 10.1007/BF00692875.

[10] Werner, R. A., and Scheeres, D. J., "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, No. 3, 1996, pp. 313–344, doi: 10.1007/BF00053511.

[11] Snir, M., Otto, S., Walker, D., Dongarra, J., and Huss-Lederman, S., *MPI: The Complete Reference*, MIT Press Cambridge, MA, USA, 1995.

[12] Axler, S., Bourdon, P., and Ramey, W., *Harmonic function theory*, Vol. 137 of *Graduate Texts in Mathematics*, Springer-Verlag, New York, second edn., 2001, URL `http://www.axler.net/HFT.html`.

[13] Babuška, I., and Melenk, J. M., "The partition of unity method," *International Journal of Numerical Methods in Engineering*, Vol. 40, 1997, pp. 727–758.

[14] Melenk, J. M., and Babuška, I., "Approximation with harmonic and generalized harmonic polynomials in the partition of unity method," *Computer Assisted Mechanics and Engineering Sciences*, Vol. 4, No. 3/4, 1997, pp. 607–632.

[15] Kumar, M., Singla, P., Chakravorty, S., and Junkins, J. L., "The Partition of Unity Method Applied to the Solution of the Fokker-Planck Equation," in *Proceedings of the 2006 AIAA/AAS Astrodynamics Specialist Conference*, 2006.

[16] Grinspun, E., *The Basis Refinement Method*, Ph.D. thesis, California Institute of Technology, 2003.

[17] Kraft, R., "Adaptive and linearly independent multilevel B-splines," *Surface Fitting and Multiresolution Methods*, 1998, pp. 209–218.

[18] Franke, R., "Scattered Data Interpolation: Tests of Some Methods." *Mathematics of Computation*, Vol. 38, No. 157, 1982, pp. 181–200.

[19] de Boor, C., *A Practical Guide to Splines*, Springer, 2001.

[20] Golub, G., and Welsch, J., "Calculation of Gauss quadrature rules," *Mathematics of Computation*, Vol. 23, No. 106, 1969, pp. 221–230.

[21] Samet, H., *The Design and Analysis of Spatial Data Structures*, Addison-Wesley, Reading, MA, 1990.

[22] Goldsmith, J., and Salmon, J., "Automatic Creation of Object Hierarchies for Ray Tracing," *IEEE Computer Graphics and Applications*, Vol. 7, No. 5, 1987, pp. 14–20.

[23] Cangahuala, L. A., "Augmentations to the Polyhedral Gravity Model to Facilitate Small Body Navigation," in *AAS/AIAA Space Flight Mechanics Meeting, Copper Mountain, Colorado*, American Astronautical Society and American Institute of Aeronautics and Astronautics, 2005, pp. 685–698, Paper AAS 05-146.

[24] Balay, S., Buschelman, K., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Smith, B. F., and Zhang, H., "PETSc Users Manual," Tech. Rep. ANL-95/11 - Revision 2.1.5, Argonne National Laboratory, 2004.

[25] Abraham, R., Marsden, J. E., and Ratiu, T., *Manifolds, Tensor Analysis, and Applications*, Springer–Verlag, New York, second edn., 1988.

[26] Chorin, A. J., and Marsden, J. E., *A mathematical introduction to fluid mechanics*, Vol. 4 of *Texts in Applied Mathematics*, Springer-Verlag, New York, third edn., 1993.

[27] Simard, P., and Mailloux, G., "A projection operator for the restoration of divergence-free vector fields," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 10, No. 2, 1988, pp. 248–256, doi:10.1109/34.3886.

[28] Tong, Y., Lombeyda, S., Hirani, A. N., and Desbrun, M., "Discrete multiscale vector field decomposition," *ACM Transactions on Graphics (Special issue of SIGGRAPH 2003 Proceedings)*, Vol. 22, No. 3, 2003, pp. 445–452, doi:10.1145/882262.882290, URL `http://www.cs.uiuc.edu/hirani/papers/ToLoHiDe2003_SIGGRAPH.pdf`.

[29] Deriaz, E., and Perrier, V., "Orthogonal Helmholtz decomposition in arbitrary dimension using divergence-free and curl-free wavelets," Preprint IM PAN Preprint 680, Institute of Mathematics of the Polish Academy of Sciences, 2007, URL `http://www.impan.gov.pl/Preprints/p680.pdf`.