JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 32, No. 6, November–December 2009

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, Illinois 61801
and
Benjamin F. Villac‡
University of California, Irvine, Ivine, California 92697

DOI: 10.2514/1.42067

Approximation-based methods, such as the cubetree algorithm, have proven 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 the 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 hundreds of thousands of trajectory propagations have recently appeared [3,4]. However, there is still a need to improve the core algorithms, such as ordinary differential equation 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. Second, as autonomous navigation becomes a reality, there is an increased demand for fast onboard 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, a novel numerical scheme (the cubetree algorithm) for the fast evaluation of gravitational force around irregular bodies has been introduced by the authors [6] and shown to provide a significant speed improvement over other methods. The scheme exploits the availability of large storage capacities to reduce the online computations. Specifically, by locally interpolating the force field around a small body, this algorithm decreased computational effort of spacecraft trajectories integration by a factor of 100. Although such results are of particular interest for large simulations, such as Monte Carlo analysis, the method may also be useful for smaller, onboard 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.

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

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.

Exactness: All conservative forces are the gradient of some potential field. The cubetree method presented in [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.

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

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.

To tackle the aforementioned issues, several modifications of the cubetree algorithm have been considered. In Sec. III, 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. Sec. IV presents a different approach based on least-squares approximation using hierarchical B-spline refinements, which provided a less stringent error requirement while addressing

Presented as Paper 7205 at the AIAA/AAS Astrodynamics Specialists Conference, Honolulu, HI, 18–21 August 2008; received 24 December 2008; revision received 8 May 2009; accepted for publication 16 May 2009.

*Graduate Student, Department of Computer Science, 201 North Goodwin Avenue; acolombi@gmail.com.

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

‡Assistant Professor, Mechanical and Aerospace Engineering, 4200 Engineering Gateway; bvillac@uci.edu. Member AIAA.

both continuity and exactness. Finally, in Sec. V, divergence-free approximations and the challenges in obtaining all three properties at the same time globally are discussed.

II. Cubetree Algorithm

To provide a basis for comparison and to better understand the challenges in addressing the preceding issues, a review of the cubetree algorithm [6] is presented in Sec. II.A. Then, obvious generalizations and inherent difficulties associated with this class of algorithms are discussed. Further details on the description, implementation, and performance of cubetree can be found in [6].

A. Cubetree Algorithm

The cubetree algorithm provides an efficient method for approximating the gravitational force generated by a small body in its immediate neighborhood [i.e., the space that is not represented efficiently by spherical harmonics, namely, the region between the surface and the sphere of convergence of the (exterior) spherical harmonics], 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 points [8]; see Fig. 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 two 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 locate the appropriate evaluation formulas. 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 $\mathcal{O}(\log N)$ tree lookup and a constant time polynomial evaluation. In practice, speedups between 300 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 cubetree model. The model in [6] requires 1150 CPU hours to initialize, and used 64 processors coordinated with the Message Passing Interface (MPI) [11].

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).

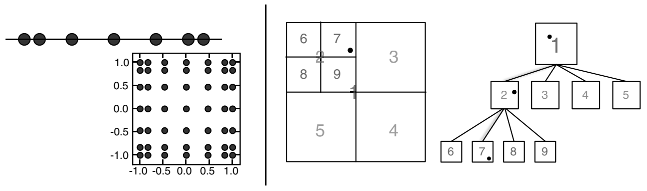

Fig. 1 Cubetree elements illustration. Left: Gauss–Lobatto–Legendre nodes, one-dimensional Gauss–Lobatto–Legendre nodes, and their two-dimensional tensor product. Right: A quadtree viewed as a tree and geometrically. Labels show the mapping between geometric and tree views. The point represents a query;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

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.

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 its Taylor series locally.

The freedom to choose an approximation scheme within each cell enables preservation of any particular property locally, that is, 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.[§] 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, the second requirement listed at the beginning of this subsection implies that the local approximations will overlap or meet at the boundaries of their respective cells, but they will not agree on this overlap region, in general. In the original cubetree algorithm, the overlap regions are the faces of the interpolation cube domains, whereas the approximation functions are interpolating polynomials which agree at shared interpolation nodes, but may not agree elsewhere on the faces. In particular, when two 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 discussed in Sec. IV.I.2 where the approximation error in adjacent cells is demonstrated.

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 algorithm 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, a first method to overcome the lack of continuity or smoothness at the boundary of the cubetree algorithm is presented. Although 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 a 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.

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

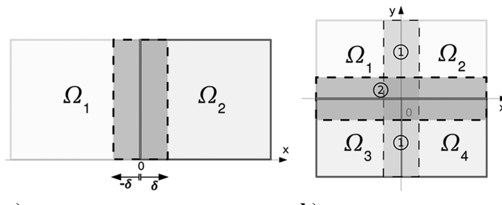

Fig. 2 Approximation around a common face (in bold): a) extension layer (dashed rectangle), b) matching one coordinate at a time.

In particular, in an overlapping region, two approximations $P_1(x)$ and $P_2(x)$, which 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 that satisfy the error requirement would also be equally fitted for the given objectives. Notably, combinations of the form

$$P_\lambda(x) = \lambda(x)P_1(x) + [1 - \lambda(x)]P_2(x)$$

with $\lambda(x) \in [0, 1]$ and varying from zero to one, can be shown to satisfy these requirements. Indeed, $|P_1 - f| \leq \epsilon$ and $|P_2 - f| \leq \epsilon$ implies that $|P_2 - P_1| \leq 2\epsilon$. This and the fact that $|\lambda| \leq 1$, results in

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

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 Fig. 2a), a method to obtain continuity across a boundary consists of the following:

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 Fig. 2a, 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 preceding estimates to include some derivative information, such as a C^k norm.

Note that the boundary-layer width must be chosen carefully so as to retain a desired accuracy. First, this extension can be done because P_1 and P_2 are in fact defined in all of $\mathbb{R}^3$ (being polynomials), whereas 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 imply that a δ can be chosen such that $|P_i - f| \leq 2\epsilon$ on $\Omega_i + \delta$,[¶] for example.

When the cubes meet at a corner, the preceding regularization can be applied one coordinate at a time. For example, considering the case of Fig. 2b, on $\Omega_\alpha = \Omega_1 \cup \Omega_2$, the preceding 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).

Because 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 preceding estimates, the approximation would be within 5ϵ for a face and 17ϵ across a corner, thus requiring ϵ (the 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 preceding regularization offers a method to provide a smooth and exact scheme, albeit at the cost of a larger memory footprint. (The size of a cubetree model depends on the number of coefficients 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.) 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 preceding regularization can be thought of as applying a partition of unity with cube domains. A partition of unity consists of a family of functions Ψ_j with compact support that covers a given domain D in such a way that 1) each point in the domain is contained in the support of only a finite number of functions in the family, and 2) the sum of these functions adds to the unity, one, at that point. The functions Ψ_j are generally assumed to be of a given smoothness, say continuously differentiable, that is, $\Psi_j \in C^1$, for example, so that any function f on D with the same smoothness properties as Ψ_j can be represented as a sum of functions with compact support

$$\sum_j \Psi_j(x) f(x)$$

Partitions of unity are generally used to extend local results about a class of functions to the whole domain of definition D.

In our case, the boundary-layer matching function $\lambda(x)$ can be extended beyond the layer to be uniformly equal to one inside a 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, the uniformly continuous approximation can be represented as

$$U \simeq \sum_{\text{cubes}C} \lambda_C(x) P_C(x)$$

and the family $\{\lambda_c\}$ thus forms a partition of unity.

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 [13–15]. 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's 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:

1) The local nature of the approximation scheme, and thus the reduction of the approximation to the right selection of the approximation function space: although the initial cubetree algorithm is based on interpolation to match 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 in which to approximate a given function. Rather, approximation schemes based on

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

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

Although the method discussed in the preceding 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, a different approximation scheme, which results in models of similar size as the original cubetree while preserving exactness and some degree of smoothness, is presented.

To create an exact approximation, proceed as done 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, $\hat{U}$ is constructed in a function space that includes only functions with the desired continuity. Put more concretely, choose a function space S with spanning set $\mathcal{B}$ such that $\phi \in \mathcal{B}$ observes,

$$\phi(\boldsymbol{x}) \in C^1$$
$$\phi(\boldsymbol{x}) \quad \text{and} \quad \nabla\phi(\boldsymbol{x})$$

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 the refinement of a degree 3 B spline in 1-D.

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 Fig. 3. At the nth iteration of refinement, the spanning set $\mathcal{B}_n$ is constructed by replacing elements $\phi \in \mathcal{B}_{n-1}$ with $\mathcal{C}(\phi)$. Because 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 nth 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 two-dimensional quadrilateral meshes, where the common edge of two squares intersects the edge of a third cell at an interior point. 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 the continuous adaptive approximation considered in this paper is built.

Before going into more detail, the following paragraph introduces the scaling function used: the B-spline basis function ϕ.

B. One-Dimensional 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 jth B spline of order k for the knot vector τ is designated $B_{j,k,\tau}$ and is defined as

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

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

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 Fig. 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 one-dimensional B splines (each with an individual knot vector) where each spline handles one dimension:

$$B_{\boldsymbol{j},k,\boldsymbol{\tau}}(\boldsymbol{x}) = \prod_{d=1}^{K} B_{j_d,k,\tau_d}(x_d); \qquad \boldsymbol{x} = (x_1, \ldots, x_K) \tag{3}$$

In Eq. (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_dth B spline of τ_d for dimension d. For example, in three dimensions, Eq. (3) is

$$B_{\boldsymbol{j},k,\boldsymbol{\tau}}(\boldsymbol{x}) = B_{j_1,k,\tau_1}(x_1) B_{j_2,k,\tau_2}(x_2) B_{j_3,k,\tau_3}(x_3) \tag{4}$$

Refinement in multiple dimensions treats each dimension independently; in our case, each dimension is refined by halving the distance between knots in the knot vector, as described at the end of Sec. IV.B. Subsequent spaces of refined K-dimensional B splines are denoted with the S_n and $\mathcal{B}_n$ notation.

The scaling functions ϕ which constitute a spanning set $\mathcal{B}_n$ for an adaptive approximation of gravitation can now be defined. Let ϕ be a three-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 1 deg 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 one dimension, simple refinements can lead to linear dependence in $\mathcal{B}_n$. Figure 4 gives several one-dimensional examples.

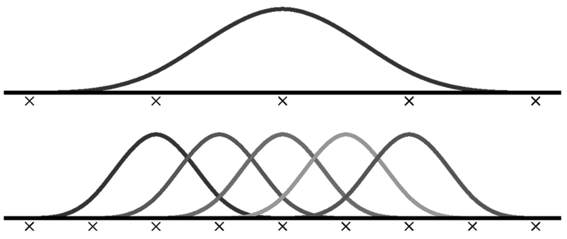
Fig. 3 Refinement relation for a degree 3 B spline. The ×s below the axis indicate locations of knots. In this refinement, every pair of knots in the original is split by a new knot halfway between them. The result is a new set of B splines that are translations and dilations of the original.

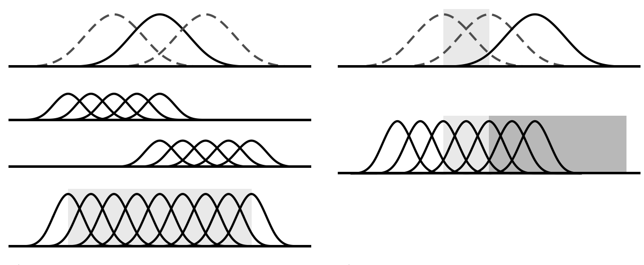

Fig. 4 Examples where refinement does not produce a basis. See Sec. IV.D for details.

As Fig. 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 that is needed 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 Fig. 4a shows (compare the second and third rows), the adjacent B splines contribute to the children of one another.

In Fig. 4, the top row shows dashed B splines are to be refined and black B splines are not. In Fig. 4a, the second and third rows are the left and right refinements, respectively, and the final row shows the combination of these refinements. Lighter 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. The example in Fig. 4b shows that restricting the domain of the approximation can make otherwise linearly independent examples, linearly dependent. The lighter shaded region is the domain of the approximation. The refinement of the dashed functions would not normally completely represent the black function because the children of the dashed functions do not cover the support of the black function (which is depicted as the union of darker and lighter shaded regions). In this case, however, the restriction to the lighter region means that they do.

The simplest way to avoid duplicating 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.[**] This problem will be revisited in Sec. IV.G.

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

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 the application presented in this paper, data points can be placed wherever convenient, which provides the flexibility of picking any norm that suits the approximation objectives. For example, one can use the norm in L^2, the space of square integrable functions, or in H^1, the subspace of L^2, in which first derivatives are also square integrable. Before going further, let us formalize the minimization problem.

Keep U and $\mathcal{B}$ as described in the preceding paragraph and pick an inner product space in which

$$\|U - \sum_j \alpha_j \phi_j\|$$

[**]For example, the C + + Standard Template Library set has 24 B of overhead per function (three pointers). In the model considered in this paper, that would cost about 860 MB of overhead, the same size as the storage requirement for the functions themselves (three-dimensional location)!

is sought to be minimized, or, equivalently,

$$\|U - \sum_j \alpha_j \phi_j\|^2 \tag{5}$$

where $\|\cdot\|$ is the norm induced by the inner product $\langle\cdot,\cdot\rangle$. By differentiating the quantity in Eq. (5) and setting the result to zero, a linear system that solves the minimization is produced: $Mx = b$, where

$$M_{i,j} = \langle\phi_i, \phi_j\rangle \tag{6}$$

$$b_i = \langle U, \phi_i\rangle \tag{7}$$

In this linear system, M is often called a mass matrix. The next natural question is with what norm shall minimize

$$\|U - \sum_j \alpha_j \phi_j\|$$

For the application of this paper, both U and $\boldsymbol{F}$ should be approximated; thus, the most obvious choice is the H^1 norm,

$$\langle f, g\rangle = \int_\Omega f(\boldsymbol{x})g(\boldsymbol{x}) + \sum_i \frac{\partial}{\partial x_i} f(\boldsymbol{x}) \frac{\partial}{\partial x_i} g(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \tag{8}$$

$$\|f\| = \sqrt{\langle f, f\rangle} \tag{9}$$

as it incorporates both gravitational potential and force.

Implementing either inner product requires numerical integration. The approximation used in this work is based on cubic polynomials; thus $\langle\phi_i, \phi_j\rangle$, that is, 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. This decision can be justified by looking at the initial goals: to approximate gravitation with cubic polynomials. Therefore, order 4 Gauss–Legendre quadrature will be as accurate (for the right-hand side) as the gravity approximation. 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 mesh defines the quadrature domains over which Eqs. (6) and (7) are evaluated.

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

This process works very well for tensor product B splines, but will fail once the refinement process begins as the mass matrix is singular when $\mathcal{B}_n$ has linear dependence. To accommodate linear dependence between levels of $\mathcal{B}_n$, the function sets $\mathcal{B}_n$ are determined hierarchically: each $\mathcal{B}_n$ is formed of nested tiers $\mathcal{T}^p$ for $0 \le p \le n$. 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 pth 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,

$$\phi \in \mathcal{T}^p \Rightarrow \phi \in \mathcal{T}^{p-1} \quad \text{or} \quad [\exists \phi' \in \mathcal{T}^{p-1} \quad \text{and} \quad \phi \in \mathcal{C}(\phi') \quad \text{and} \quad \phi' \mathcal{T}^{p-2}]$$

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 one-dimensional example, and demonstrates the difference between $\mathcal{B}_n$ and $\mathcal{T}^p$.

To find the coefficients for $\mathcal{B}_n$, the tiers are solved one at a time by iterating through $\mathcal{T}^p$ according to the following procedure. First, solve $\mathcal{T}^0$ using Eqs. (6) and (7). Then, solve recursively $\mathcal{T}^p$ by assuming the coefficients of $\mathcal{T}^{p-1}$ and separating $\mathcal{T}^p$ into two sets $\mathcal{N}$ and $\mathcal{R}$, defined such that

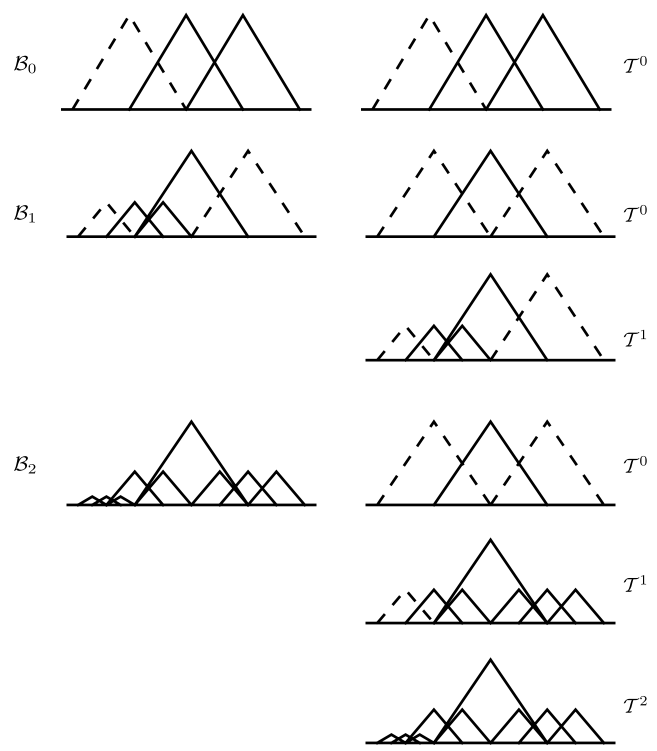

Fig. 5 Examples of tiers $\mathcal{T}^p$ that constitute a given spanning set $\mathcal{B}_n$. Dashed B splines show which are to be refined at the next tier. Notice the difference between $\mathcal{T}^1$ and $\mathcal{B}_1$ in the third iteration. This demonstrates that $\mathcal{B}_n$ may refine any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

$$\phi_j \in \mathcal{N} \Rightarrow \phi_j \in \mathcal{T}^{p-1} \qquad \phi_j \in \mathcal{R} \Rightarrow \exists \phi' \in \mathcal{T}^{p-1} \quad \text{and} \quad \phi_j \in \mathcal{C}(\phi')$$

In other words, $\mathcal{R}$ contains the scaling functions that come from refining $\mathcal{T}^{p-1}$, whereas $\mathcal{N}$ contains those which are carried without refinement from one tier to the next. Then, the functions $\phi_j \in \mathcal{T}^p$ are organized such that the first $|\mathcal{R}|$ scaling functions of $\mathcal{T}^p$ are the new scaling functions, and the rest are from the previous tier $\mathcal{T}^{p-1}$. That is, $\phi_j \in \mathcal{R}$ when $j < |\mathcal{R}|$ and $\phi_j \in \mathcal{N}$ otherwise. Next, the mass matrix M of $\mathcal{T}^p$ is created as usual via Eq. (6), and the scaling functions that come from earlier levels are removed by taking their coefficients from the solution to the previous tier, $\mathcal{T}^{p-1}$. This is achieved by removing the last $|\mathcal{N}|$ rows of M and b, because these equations have already been solved in $\mathcal{T}^{p-1}$. This results in the underdetermined system $\bar{M}'x = \bar{b}'$. Then, using the coefficients from $\mathcal{T}^{p-1}$, remove the last scaling functions in $\mathcal{N}$ by subtracting their contributions from the right-hand side, thus creating $\bar{b}$. Finally, solve the linear system $\bar{M}x^p = \bar{b}$.

Expressed as equations, the process can be summarized as follows:

1)

$$\bar{M}'_{i,j} = M_{i,j} \quad \text{for } i < |\mathcal{R}|, \qquad j < |\mathcal{T}^p|$$

2)

$$\bar{b}_i = b_i - \sum_j \bar{M}'_{i,j} x^{p-1} \quad \text{for } i < |\mathcal{T}^p|, \qquad j < |\mathcal{T}^p|$$

3)

$$\bar{M}_{i,j} = \bar{M}'_{i,j} \quad \text{for } i < |\mathcal{R}|, \qquad j < |\mathcal{R}|$$

4)

$$x^p = \bar{M}^{-1}\bar{b}$$

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, and 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 the given linear systems to solve: in practice, the last $|\mathcal{N}|$ rows of M are never needed, and so they need not 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 the model required 69 GB; doubling this makes for a very large matrix indeed.

G. Implementation Details

In the subsequent sections, implementation details that allow the proposed model to efficiently represent small-body gravitation are described.

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

This section focuses on representing $\mathcal{B}_n$ in a way suitable to preparing and using the approximation to small-body gravitation presented in the previous section. Two concerns dominate the design: memory efficiency during model creation and computation efficiency during force reconstruction. The former is the primary bottleneck of the 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 the chosen 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.

The 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, $\mathcal{P}_i := \{B_{j,3,\tau}\}$, with a single data structure. In this way, any number of B splines can be represented 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 Sec. IV.I.1 requires 1.5 MB of storage to represent 18 million B splines. The second advantage is in evaluating $\mathcal{B}_n$. The most

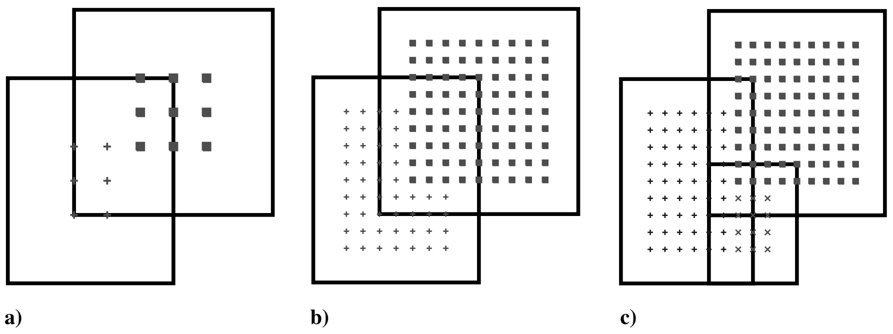

Fig. 6 The patch splitting process. Rectangles show the support of each patch, and markers show the peaks of the scaling functions: a) there are two patches; b) result of each patch being refined by replacing the present scaling functions with their children; c) a 3 × 5 block of scaling functions are redundant, so three patches are produced by one patch being split to avoid the redundant region.

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.

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 the chosen data structure, this amounts to dividing the `knot_distance` by two. However, just as with individual scaling functions of Fig. 4a, some patches will share children. That is, the basis refinement of CHARMS, used to properly approximate the gravity field at a given location, causes some overlap of the patches. Thus, to accommodate redundancy in the children, some patches must be resized to avoid each other; a process referred to as patch splitting in the following.

Consider the two-dimensional example in Fig. 6. From $a \rightarrow b$, two adjacent patches are refined; oblivious to one another, they generate redundant children. To eliminate the repeated children, one patch is selected to avoid the other, for example, $b \rightarrow c$. The number of additional patches created by this process, which should be kept as small as possible to avoid large memory footprints, is reduced by always splitting the patch that contains fewer scaling functions. (This heuristic was found to globally produce fewer patches than always splitting the patch that locally creates fewer patches.)

Therefore, the proposed algorithm for refinement is 1) loop through patches and refine those that need refinement, and 2) pairwise compare patches to avoid overlapping refinements using patch splitting. The pairwise comparison may be efficiently implemented with an octree data structure [21], but, in our experience, the cost of the straightforward $\mathcal{O}(N^2)$ pairwise comparisons is small compared to solving a matrix with 20 million unknowns.

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

Although the favorable efficiency of evaluating the B splines of $\mathcal{B}_n$ with patches has already been mentioned, a mechanism for finding the patches that affect a given query still needs to be defined. For this, a bounding volume hierarchy (BVH) [22] will be used.

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: whereas 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. The function `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 previously, the tree will always be balanced; hence, most queries will only require $\mathcal{O}(\log N)$ 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.

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.

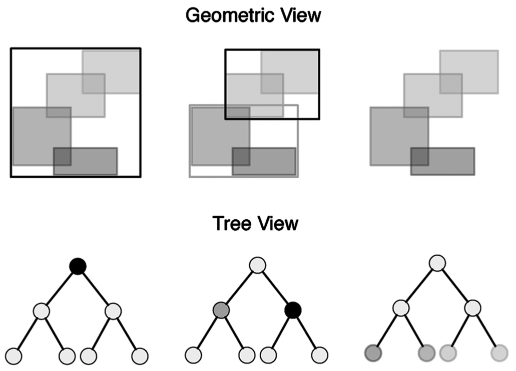

Fig. 7 A bounding volume hierarchy. The objects organized are the shaded boxes. On the left is a geometric view. The first level shows the bounding box in black, the second level shows it in black and gray, and the last level only the underlying objects remain. On the right is a tree view of the bounding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

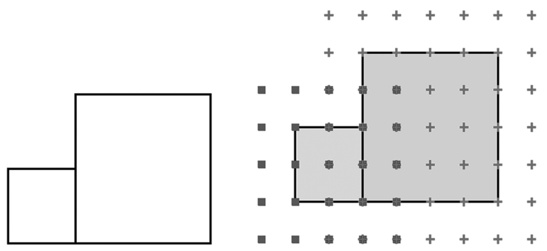

Fig. 8 Detailed view of initializing $\mathcal{B}_0$. Left: Ω is the interior of the two boxes. Right: Patches are placed so the the domain of patches covers the cuboid it came from. This creates redundancies, which the overlapping markers (crosses and squares) show. Shaded regions show the domain of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

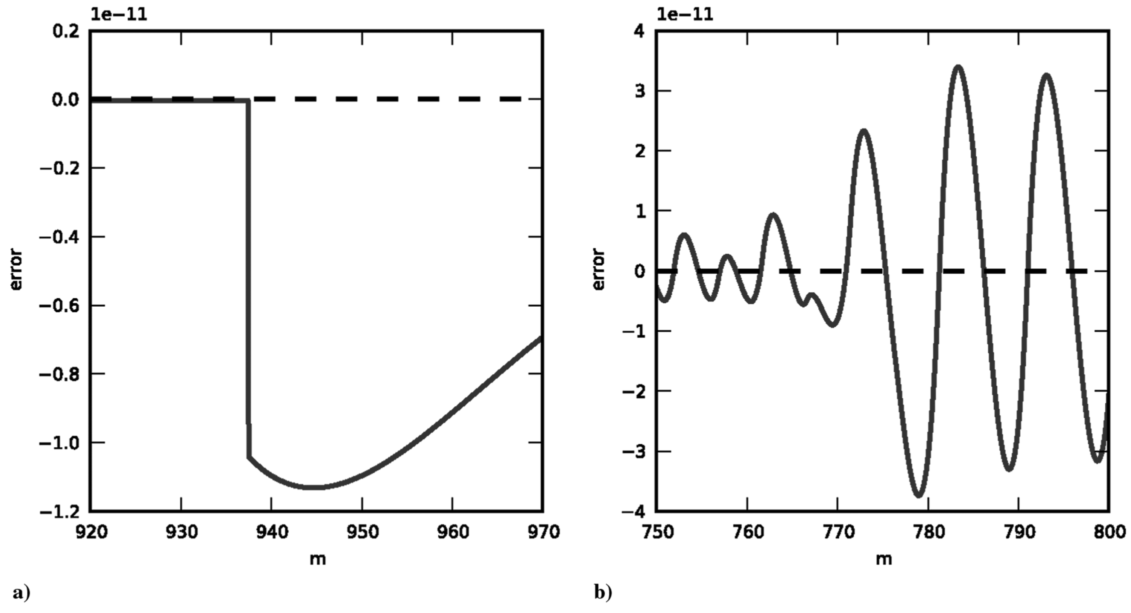

Fig. 9 Continuity comparison of cubetree and B-spline method. In each case, the absolute error in the x component of the force is measured along a 50 m ray. The dashed line at $y = 0$ represents the actual force. As the T junctions do not occur in the same places for each model, different rays are measured.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

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

The domain of approximation Ω plays three roles during a model creation: 1) it provides the domain over which the inner product [Eq. (8)] is evaluated; 2) it seeds $\mathcal{B}_0$ with B-spline patches necessary to cover Ω; and 3) it defines the regions over which the error of the computed approximation is measured. 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, an octree is used, 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 Figs. 6b, 6c, and 8.

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

To estimate relative error, the octree domain Ω is sampled. 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, let us turn our attention to approximating the gravitational potential and gravitational force of asteroid 1998 ML14. The polyhedral model for the chosen asteroid is the same as the one used in [6].[††]

1. Notes on Model Creation

The domain of approximation 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. The largest cube was also limited 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 MPI [11] (64 processors for approximately 80 h) 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 $2500/2^9$–$2500/2^7$ m. To cover the region outside the octree, spherical harmonics of degree and order 12 were employed. The resulting model will be referred as patches of uniform B-splines tree, or pubtree for short, in the remainder.

2. Continuity at T Junctions

Next, continuity between levels of refinement is tested. In Fig. 9a, the error in the cubetree's force at a T junction is plotted and the discontinuity is readily apparent at 935 m. Comparing this to the same plot for a T junction in the B-spline model, Fig. 9b, one can see that no discontinuities exist; in fact, the only evidence of a T junction is a slight shift in the oscillatory pattern at 770 m.

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 Fig. 10, which plots relative error in gravitational potential and gravitational force along the $z = 0$ plane. Note that $\hat{U}$ is about 3 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 is 40 m from the surface. Future improvements to patch refinement may shrink the band further, though the achieved accuracy seemed sufficient to demonstrate the model and for many applications.

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 Fig. 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

[††]Data available online at http://www.psi.edu/pds/resource/rshape.html, EAR-A-5-DDR-RADARSHAPE-MODELS-V2.0 [retrieved 10 June 2007].

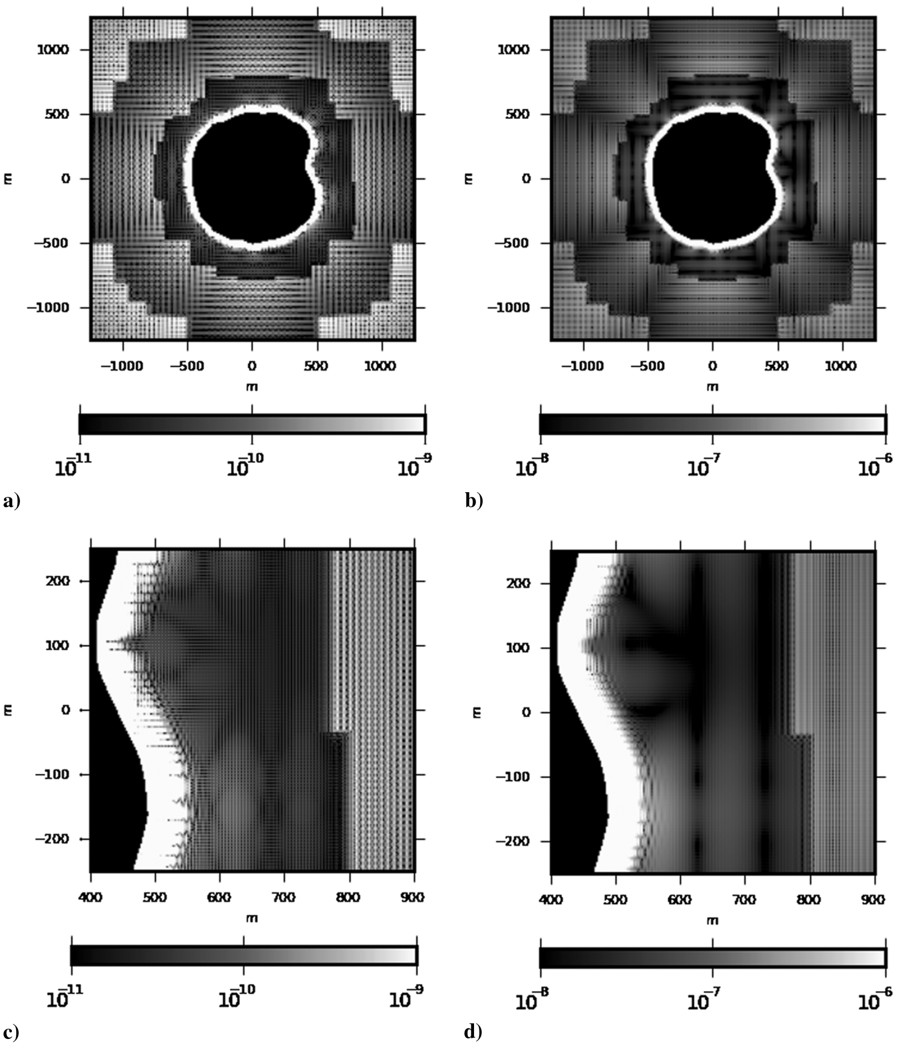

Fig. 10 Relative error of pubtree across the complete domain. The interior of the asteroid is not measured and is left black. Error is plotted on a log scale. Error a) across the entire domain in potential, b) across the entire domain in force, c) near the surface in potential, and d) near the surface in force.

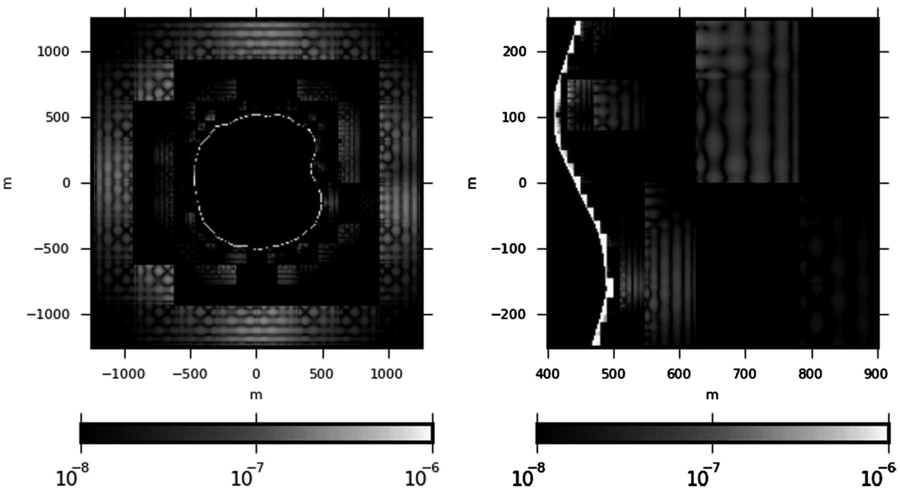

Fig. 11 Relative error in the force produced by cubetree. The interior of the asteroid is not measured and is left black. Error is plotted on a log scale. Left: the error over the entire domain. Right: sample of the error in a region abutting the asteroid. Compare these to Fig. 10.

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 Fig. 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 1 order of magnitude.

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 will be seen in the following section. Of course, quantitative improvements over cubetree was not the goal of this research. Furthermore, both models easily achieve the target accuracy of 10^{-5}.

4. *Refinement Sequence*

Figure 12 shows the model as developed over three iterations. After each iteration, regions with high error are reduced (e.g., the band of white surrounding the asteroid, which shrinks after each iteration).

5. *Trajectory Integrations*

The following trajectory simulations use the same models as the ones from [6], as well as the pubtree model presented in this paper.

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

A family of retrograde orbits starting between 600 and 1000 m from the asteroid center has been integrated. 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 five 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 min of simulated time.

This experiment was repeated for 993 trajectories. For each trajectory, the maximum difference in position and velocity between the cubetree trajectory and reference trajectory has been measured. Figure 13 is a histogram of the errors in position and velocity. For comparison's sake, the same histograms for the cubetree model are presented in Fig. 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 times faster than the augmented polyhedral, compared to 301 times for cubetree.

J. Discussion

The CHARMS framework, coupled with the hierarchical coefficient fitting that was developed in this paper, 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.

1. *Trajectory Integration*

During the trajectory integration experiments, pubtree was found to be 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 Fig. 9. Reducing the amplitude by tightening the error threshold would resolve this, thereby increasing the speed of pubtree.

2. *Patch Refinement*

As described in previous sections, 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, the iterations of pubtree ended exactly for this reason.

The solution to this problem is subpatch 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, the error is localized. Then, instead of refining whole patches, they are only split so that the scaling functions affecting the localized error are refined. Following this algorithm would reintroduce adaptivity and permit further iterations on pubtree.

3. *Linear Independence*

Although a linearly dependent spanning set $\mathcal{B}_n$ has been pursued in this research, 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

The previous section described exact and continuous approximation schemes, but did not address harmonicity of the gravitational potential. The fact that the Laplacian of the potential is zero in free space results in a divergence-free gravitational force. Although the converse is not necessarily true, a first requirement on an approximation scheme consists of requiring the force approximation to be divergence free.

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:

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

The presence of holes and handles in a domain results in a nonzero harmonic part in the preceding 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

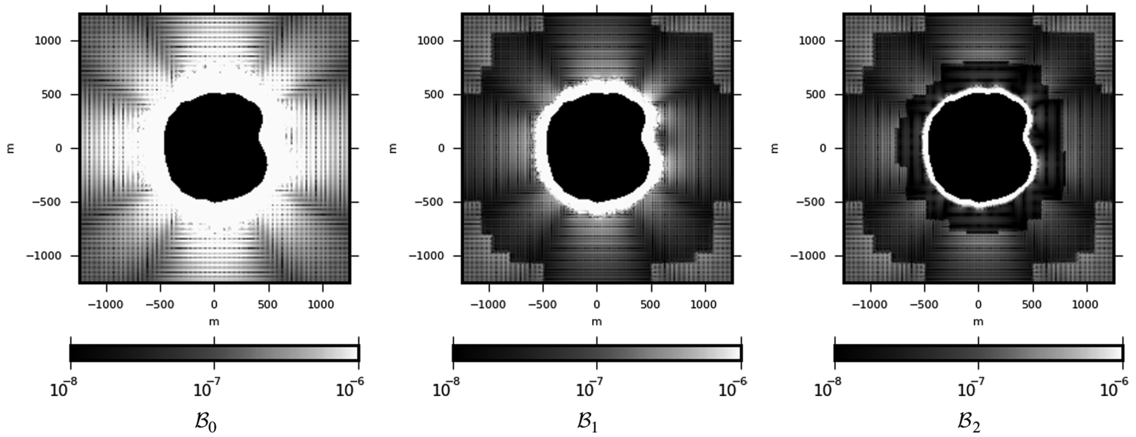

Fig. 12 Relative error over the sequence of refinement. The interior of the asteroid is not measured and is left black. Error is plotted on a log scale.

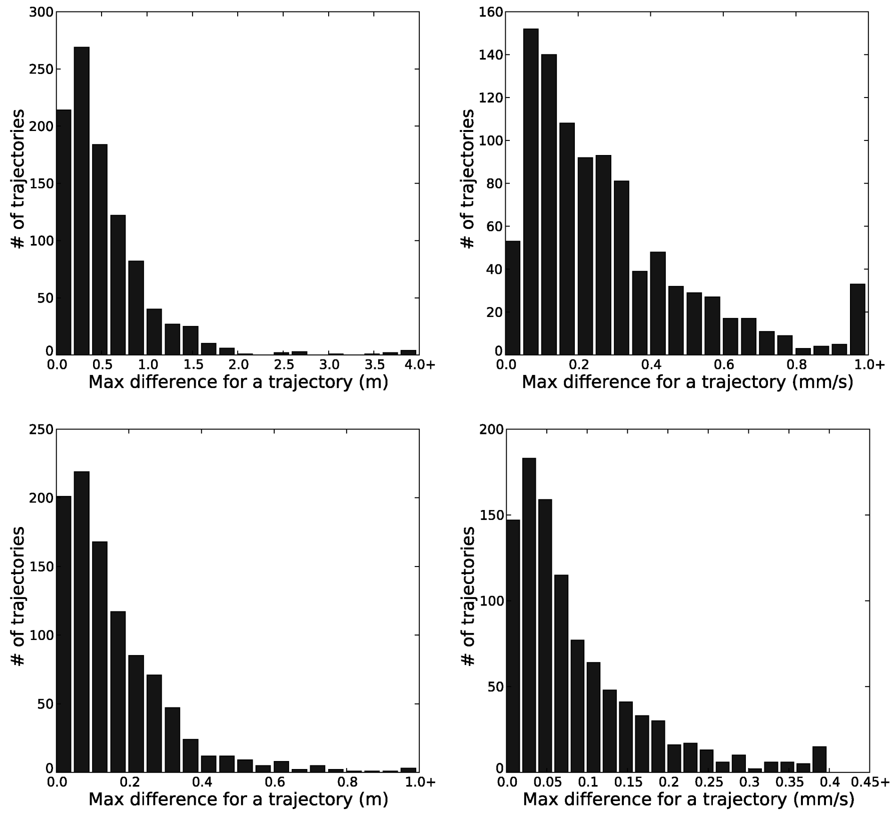

Fig. 13 Histograms of errors in position and velocity for 993 retrograde pubtree trajectories integrated for five days and observed at 5 min intervals. Top pair of graphs are pubtree position and velocity errors, whereas the bottom pair correspond to the cubetree position and velocity errors.

proposed in the literature, such as [27–29], and so the analysis of divergence-free approximations alone has not been considered in this paper. However, these projection can be performed after the fact, on the already-developed approximation schemes presented in the previous questions. That is, the imposition of a zero divergence of the vector field can appear as a correction on the force evaluated from the previous approximation schemes.

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. Tong et al. [28] 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 that is also divergence-free satisfies Laplace's equation. That is, the potential is a harmonic function, and is thus, together with all of its derivatives, an analytic function. Thus, an approximation scheme aiming at being both exact and divergence free must provide an analytic approximation.

More important, the solution of Laplace's 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 Sec. 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 preceding right-hand side vanish, the remaining term imposes a constraint on the derivative, which is not free to choose, as discussed in Sec. II. A straightforward 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, although the partition of unity method has been used with harmonic polynomials in [14], the concern was about convergence (i.e., 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 losing some of the smoothness of the problem.

The preceding properties illustrate the basic fact that being harmonic in a small region has some global implications. In effect, approximating the potential by a harmonic function would consist of solving exactly Laplace's equation, which indicates the close link between preserving harmonicity and solving exactly Laplace's equation for nonconvex 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, whereas the spline approximation has been motivated by higher-order finite elements methods.

VI. Conclusions

Continuous and exact approximations of gravitational forces in the neighborhood of small bodies has been described. These approximations preserve some degree of smoothness and were obtained in the setting of adaptive piecewise polynomial approximation initially considered in the cubetree method.

Although 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 timespan 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. Although such model building is initially computationally expensive, the locality inherent in these adaptive schemes make model updates and force evaluations extremely fast. As discussed in the paper, trajectory integrations can be 2 orders of magnitude faster than the polyhedral method.

References

[1] Lee, S., Fink, W., Russell, R., Von Allmen, P., Petropoulos, A., and Terrile, R., "Evolutionary Computing for Low Thrust Navigation," AIAA Paper 2005-6835, 2005.

[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. doi:10.2514/1.22984

[3] Russell, R., "Global Search for Planar and Three-Dimensional Periodic Orbits Near Europa," *Journal of the Astronautical Sciences*, Vol. 54, No. 2, 2006, pp. 199–226.

[4] Villac, B., "Using FLI Maps for Preliminary Spacecraft Trajectory Design in Multi-Body Environments," *Celestial Mechanics and Dynamical Astronomy*, Vol. 102, Nos. 1–3, 2008, pp. 29–48. doi:10.1007/s10569-008-9158-1

[5] Gurfil, P., "Simple Satellite Orbit Propagator," American Astronautical Society Paper 08-167, 2008.

[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

[7] Fetecau, R. C., Marsden, J. E., and West, M., "Variational Multisymplectic Formulations of Nonsmooth Continuum Mechanics," *Perspectives and Problems in Nonlinear Science*, edited by E. Kaplan, J. E. Marsden, and K. R. Sreenivasan, Springer–Verlag, New York, 2003, pp. 229–263.

[8] Karniadakis, G. E., and Sherwin, S., *Spectral/hp Element Methods for Computational Fluid Dynamics*, Oxford Science Publ., New York, 2005, Chaps. 2–3, 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 Derived and Compared with Harmonic and Mascon Gravitation Representations of Asteroid 4769 Castalia," *Celestial Mechanics and Dynamical Astronomy*, Vol. 65, No. 3, Sept. 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, 1995.

[12] Axler, S., Bourdon, P., and Ramey, W., *Harmonic Function Theory*, Graduate Texts in Mathematics, 2nd ed., Vol. 137, Springer–Verlag, New York, 2001, pp. 1–26, 73–106, http://www.axler.net/HFT.html.

[13] Babuška, I., and Melenk, J. M., "The Partition of Unity Method," *International Journal for Numerical Methods in Engineering*, Vol. 40, No. 4, 1997, pp. 727–758. doi:10.1002/(SICI)1097-0207(19970228)40:4<727::AID-NME86>3.0.CO;2-N

[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, Nos. 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," *Proceedings of the 2006 AIAA/AAS Astrodynamics Specialist Conference*, American Astronautical Society Publ. Office, San Diego, CA, 2006, pp. 21–24.

[16] Grinspun, E., "The Basis Refinement Method," Ph.D. Thesis, California Inst. of Technology, Pasadena, CA, 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. doi:10.2307/2007474

[19] de Boor, C., *A Practical Guide to Splines*, Springer, New York, 2001, Chap. 14.

[20] Golub, G., and Welsch, J., "Calculation of Gauss Quadrature Rules," *Mathematics of Computation*, Vol. 23, No. 106, 1969, pp. 221–230. doi:10.2307/2004418

[21] Foley, J. D., Dam, A. V., Feiner, S. K., and Hughes, J. F., *Computer Graphics: Principles and Practice*, 2nd ed., Addison–Wesley, Reading, MA, 1995, Chap. 12, pp. 551–557.

[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. doi:10.1109/MCG.1987.276983

[23] Cangahuala, L. A., "Augmentations to the Polyhedral Gravity Model to Facilitate Small Body Navigation," American Astronautical Society Paper 05-146 2005.

[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 User's Manual," TR ANL-95/11, Rev. 2.1.5, Argonne National Lab., 2004.

[25] Abraham, R., Marsden, J. E., and Ratiu, T., *Manifolds, Tensor Analysis, and Applications*, 2nd ed., Springer–Verlag, New York, 1988, Chap. 7, p. 539.

[26] Chorin, A. J., and Marsden, J. E., *A Mathematical Introduction to Fluid Mechanics*, Texts in Applied Mathematics, 3rd ed., Vol. 4, Springer–Verlag, New York, 1993, Chap. 1, p. 37.

[27] Simard, P., and Mailloux, G., "A Projection Operator for the Restoration of Divergence-Free Vector Fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 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*, Vol. 22, No. 3, 2003, pp. 445–452, http://www.cs.uiuc.edu/hirani/papers/ToLoHiDe2003_SIGGRAPH.pdf. doi:10.1145/882262.882290

[29] Deriaz, E., and Perrier, V., "Orthogonal Helmholtz Decomposition in Arbitrary Dimension Using Divergence-Free and Curl-Free Wavelets" (Preprint 680), Inst. of Mathematics of the Polish Academy of Sciences, 2007, http://www.impan.gov.pl/Preprints/p680.pdf.