JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 39, No. 1, January 2016

Efficient Interpolation of High-Fidelity Geopotentials

Nitin Arora* and Ryan P. Russell†
University of Texas at Austin, Austin, Texas 78712

DOI: 10.2514/1.G001291

With a goal of efficiently trading higher-memory footprints for faster runtimes, a high-fidelity interpolation method is presented for approximating a scalar quantity and associated gradients in the global three-dimensional domain external to a sphere. The new "Fetch" interpolation model uses the Junkins weighting function strategy to achieve continuity and smoothness. An overlapping grid strategy ensures a singularity-free domain while minimizing associated memory costs. Local interpolating functions are judiciously chosen with a new adaptive order-based selection of local polynomials that minimizes coefficient storage subject to a radially mapped residual tolerance. Analytic inversions of the normal equations associated with each candidate polynomial allow for rapid solutions to the least-squares process without resorting to the conventional numerical linear system solvers. The gradient and higher-order partial derivatives are computed directly with no memory cost, and they are smooth and continuous to a user-specified order. The method is specifically applied to interpolate the GRACE GGM03C geopotential model up to the degree and order of 360. Highly tuned interpolation models of various resolutions are presented and discussed in detail. Released Fetch interpolation models of the geopotential include resolutions of 33×33, 70×70, 156×156, and 360×360. The memory requirements span from 120 to 2360 MB, and the expected speedups over spherical harmonics evaluations span from approximately 3- to 800-fold. The minimum break-even resolution for runtime speeds is approximately the degree and order of 13. The models are valid up to an altitude of 60 Earth radii and are globally continuous to order three, and thus may be of interest to a variety of science and engineering applications. The runtime evaluation codes along with model coefficients are available on the Internet.

Nomenclature

B	=	least-square inverse matrix
$\boldsymbol{C}$	=	coefficient vector for a node polynomial
$\boldsymbol{D}_w$	=	weight function normalization matrix
$\boldsymbol{D}_{\alpha,\beta,\gamma}$	=	node polynomial normalization matrix
d	=	degree of a local node polynomial
$\hat{e}_1, \hat{e}_2, \hat{e}_3$	=	unit vectors in polar angle, azimuth, and radial directions
m	=	total number of measurements per cell in each direction
N	=	total number of coefficients of a general node polynomial
$N_{i,j,k}$	=	total number of coefficients of a node polynomial, associated to the i, j, k cell node
$N_{\max}$	=	maximum number of coefficients for a node polynomial
$O_{\max}$	=	maximum degree of a node polynomial
Q_d	=	major candidate polynomial of degree d
$Q_{d,z}$	=	minor candidate polynomial of degree d and index z
R_e	=	mean radius of the Earth
U_F	=	final composite potential computed using a weighted average
$U_{i,j,k}$	=	local polynomial associated to the i, j, k cell node
$w_{i,j,k}$	=	weight function associated to the i, j, k cell node
$\boldsymbol{x}$	=	absolute position vector of the evaluation point in spherical coordinates
$\boldsymbol{x}_{i,j,k}$	=	absolute position vector of the i, j, k cell node in spherical coordinates
$\boldsymbol{x}_w$	=	normalized position vector in spherical coordinates of the evaluation point in the cell domain
$\boldsymbol{y}$	=	normalized (−1 to 1) position vector of the evaluation point in spherical coordinates
δ	=	polar angle (0 to 180 deg)
δ, λ, r	=	spherical coordinate set (where r is greater than or equal to one)
$\eta_{0\ldots4}$	=	user-controlled model residual tolerance constants
λ	=	azimuth (equivalent to geocentric longitude, 0–360 deg)
σ	=	Total loop counter
ϕ	=	geocentric latitude (δ-90)
Ω	=	Hermite weighting function

Received 9 March 2015; accepted for publication 15 April 2015; published online 6 October 2015.

*Research Fellow, Department of Aerospace Engineering and Engineering Mechanics; currently Mission Design and Concept Formulation, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109; Nitin.Arora@jpl.nasa.gov. Member AIAA.

†Assistant Professor, Department of Aerospace Engineering and Engineering Mechanics; ryan.russell@utexas.edu. Associate Fellow AIAA.

I. Introduction

HIGH-FIDELITY trajectory computation using conventional spherical harmonics (SH) gravity fields is computationally slow and difficult to implement if starting from scratch [1–3]. The problem is expected to compound with the release of new higher-order gravity field models [4,5]. With the current computational resources, it is typically not feasible to account for most the accurate geopotential models in many high-fidelity trajectory simulations, such as monitoring the space catalog [6]. Accordingly, there is a pressing need for a new class of gravity models that can achieve orders of magnitude in speedup over SH while still preserving accuracy and robustness of the SH approach.

Various alternatives to SH have been proposed and can be loosely divided into two classes: 1) discrete mass models, and 2) interpolation models. Discrete mass models (deemed as any model using point masses, or surface or volume mass distributions) require only a limited amount of global data [7–12] and are straightforward to evaluate. They can be easily coupled with SH models for the benefit of an adaptive local resolution. Recently, there has been a growing interest in robotic and potential human exploration of small irregular-shaped bodies [13]. The volume-based and surface-based discrete mass models are therefore compelling because they are valid in the entire domain, whereas SH representations fail to converge inside the reference radius [14,15]. In particular, the polyhedral method [16] is a robust and elegant solution for irregular small bodies, although the extra computational requirements are cumbersome. Recently, a modern high-fidelity geopotential mascon model has been demonstrated to

provide an order of magnitude of speed improvements over SH when implemented on a common graphics processing unit (GPU) [17]. Recent modifications to the SH model [18,19] allow for evaluation inside of the circumscribing sphere (below the traditional SH reference radius) and present an attractive alternative to the volume-based and surface-based discrete mass models for nonspherical bodies.

The second, and more popular, class of alternative potential formulations includes the interpolation models. Applicable to both irregular- and near-spherical-shaped bodies, methods in this class expedite computations by effectively trading computer memory for runtime speed. Essentially first proposed by Junkins in 1976 [20], geopotential interpolation methods have been bolstered recently by the abundant memory resources of common computers. Depending on the method, a variety of techniques and basis functions have been employed, including weighting functions [21,22], wavelets [23], splines [23,24], octrees [25], pseudocenters [26], and three-dimensional (3-D) digital modeling [27]. Each interpolation method balances accuracy with efforts to minimize runtime speed and memory footprint while achieving exactness, continuity, and smoothness as appropriate.

The 3-D geopotential interpolation model developed in this study is motivated primarily by the works of Junkins [20], Junkins and Engels [22], Hujsak [26], Colombi et al. [25], Oltrogge [27], and Beylkin and Cramer [23], among others. Over 30 years ago, Junkins demonstrated that a one order-of-magnitude speedup was possible at the expense of a modest investment in memory [20]. This was a remarkable feat, considering the quality of computers at the time. His model encompassed the region around Earth out to 1.2 radii and achieved roughly six digits of accuracy at the expense of storing 30,000 coefficients. Hujsak revisited the problem approximately 20 years later, introducing the concept of interpolating the pseudocenter coordinates instead of the potential or acceleration [26]. With a simpler interpolating scheme and improved hardware, he fit a 70×70 field (for the northern hemisphere at altitudes between 400 and 1500 km) with only 5 MB of storage. The algorithm requires the calculation overhead equivalent of a 5×5 field, leading to approximately a 100-fold speedup compared to SH for the 70×70 resolution. Columbi et al. recently applied similar concepts using modern tools for calculating gravity fields for highly nonspherical bodies such as comets and asteroids [25]. Their methods are demonstrated to provide approximately 100-fold speedups, except their gains are compared to the expensive polyhedral methods. Beylkin and Cramer [23] showed recent progress in the efficient storage of multiresolution interpolating functions, and Jones et al. subsequently published the first global, modern 3-D interpolation geopotential model called the cubed-sphere model [28]. Using multiple concentric shells and Chebyshev basis functions, their method demonstrated 30-fold speedups compared to SH for their highest-resolution 150×150 model. Their break-even model in terms of runtime is said to be approximately at the resolution of 20×20. The memory footprint of their 150×150 model is 856 MB. Their model suffers from small discontinuities across shell boundaries, and it requires the storage of extra coefficients for acceleration and higher-order derivatives.

The main drawback of 3-D geopotential interpolation gravity models is their memory requirements. As memory and processor technology has advanced significantly over the years, a renewed interest in memory-intense numerical methods is increasingly justified. Given the potential benefits of leveraging the recent increase in available memory for speed improvements, we propose a new hybrid model (called Fetch) that attempts to take advantage of all the previous models while finding innovative solutions to correct their respective problems. Accordingly, the proposed Fetch model improves on shortcomings of the aforementioned interpolation models as well as the prototype Fetch model [29]. As with the previous Fetch model, priority is placed on a solution method that 1) is continuous and smooth across the global domain to an arbitrary order of derivatives, 2) is adaptive in terms of local vs global resolution, 3) has a residual error profile that is in the noise of the accuracy of the underlying base model (SH in this study), and 4) is singularity free. No previous or existing gravity field interpolation model can claim a complete handle on each of these ambitious priorities.

The new Fetch model optimally trades memory for speed and takes advantage of a weighted interpolation scheme (developed by Junkins et al. [21]) to achieve global continuity. Continuity in zeroth- and higher-order derivatives is a desirable feature for any force model. Global continuity in the first derivative is necessary to accurately represent a conservative field, and continuity in higher-order derivatives may be important, depending on the specific application. Most 3-D interpolation-based methods are not globally continuous or require complex algorithms to achieve continuity. The tricubic interpolation method by Lekien and Marsden [24] is an example that achieves first-order continuity but at the expense of reduced accuracy and performance. Furthermore, in their method, the interpolants must be of the same degree globally, and accurate high-order derivatives of the fitting data are required for the coefficient generation process. The cubed-sphere model is discontinuous even to the zeroth order across the shell boundaries, although the discontinuity is small in the published models. Also, like the tricubic method, the degree of the interpolants for the cubed-sphere model are fixed globally.

The classic singularity and associated numerical problems near the poles (when converting to Cartesian frame) are avoided in the Fetch model using a two-level overlapping global grid structure with an additional weighting function to ensure continuity between the grids. This approach to remove the singularity only requires extra coefficients in the overlapping region ($\sim$0.2 to 5% of the domain for high- and low-resolution models, respectively), and it is therefore more memory efficient than storing an extra term throughout the entire domain. The new model also introduces a novel polynomial selection strategy based on an ordered listing of potential polynomial basis functions, leading to significant improvements in terms of memory efficiency. The method uses an algebraic manipulator to produce high-order analytic inverses for the resulting least-squares problems to ensure accurate and rapid coefficient evaluation. High-order polynomial approximations are feasible via a master-worker implementation of the parallel coefficient generation algorithm [implemented using the message-passing interface (MPI)] and by using a least-squares fitting algorithm on a nonuniform grid. The original Junkins weighting function method [20] is modified to handle this new memory-saving nonuniform grid. The interpolated geopotential is exact, in the sense that accelerations and higher-order derivatives are calculated as exact derivatives of the interpolated function. Furthermore, the memory requirements scale approximately linearly with the degree of the base field, whereas the speedups are faster than quadratic in the same argument. This favorable memory scaling makes the new Fetch model attractive for use with high-order gravity fields.

We present details on the Fetch interpolation approach when applied to the geopotential application. Information on how to use the Fetch gravity model is given, which is followed by a comprehensive performance profiling via a direct comparison with a state-of-the-art singularity-free SH implementation. The Fetch algorithm is developed to be general, in a sense, so that a user can control the coefficient generation trade of memory vs speedup with a minimum number of parameters. We apply the algorithm using the GGM03C SH gravity field to compute four Fetch models, along with the qualitative characteristics of each. Specifically, the highest-resolution Fetch model discussed in this study corresponds to the complete 360×360 GMM03C SH field and it required on the order of 12,000 CPU hours[‡] to compute all of its coefficients. The highly optimized memory burden of this model was 2.36 GB, leading to as much as three orders of magnitude in runtime speedups over the GGM03C SH model. On the other end of the spectrum, an 8×8 Fetch model was found to approximately match the runtime speed of its associated SH counterpart. The next section gives an overview of the Fetch gravity model.

[‡]Data available online at http://www.tacc.utexas.edu [retrieved 2015].

II. Fetch Gravity Model Overview

A localized representation of the gravity field follows naturally from the fact that there are uneven gravity undulations over the Earth surface. To separate the dominant global effects from the smaller local undulations, we formulate the geopotential approximation as given by Eq. (1):

$$U \simeq -\frac{\mu}{r} + U_{J_2} + U_F \tag{1}$$

where U represents the total potential from a specified degree and order of the GGM03C spherical harmonics field, U_{J_2} represents the potential due to the J_2 term, and U_F is the interpolated local potential obtained from the Fetch model. The J_2 term in Eq. (1) is three orders of magnitude more significant than all the higher-order terms combined. Removing of U_{J_2} [given by Eq. (2)] from higher-order terms provides an extra three to four digits of accuracy in the potential interpolation at an almost negligible computational cost. Most previous gravity field interpolation efforts have also exploited the benefit of removing the low-frequency terms [14,17,20,23,25,26,28]. If applicable, solid Earth tides or any other time-dependent low-frequency terms could also be removed from the static interpolation. In this study, however, only the mean J_2 term is removed:

$$U_{J_2} = -\frac{\mu}{r}\left[J_2\left(\frac{R_e}{r}\right)^2 P_2(\sin(\phi))\right] \tag{2}$$

where μ and R_e are the reference gravitational parameter and radius of the Earth, respectively; r and ϕ are the magnitude and geocentric latitude of the position vector, respectively; and P_2 is the second-degree Legendre polynomial. The U_F term in Eq. (1) represents the final composite polynomial, and it is computed using a weighted average of the eight-node polynomials for which the centers depend on the spacing of the neighboring eight cells [see Fig. 1 and Eq. (3)] [20,21]. The coefficients for each of the eight local node polynomials are computed via least-squares fits of the local geopotential (with the two-body and J_2 terms removed). More details are provided later on the least-squares solutions and candidate forms for the local node polynomials:

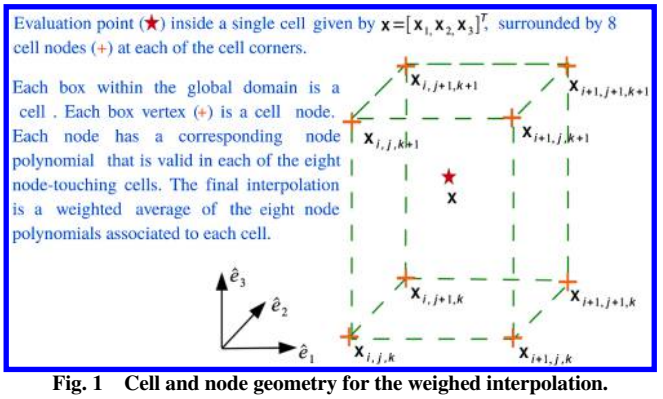

Fig. 1 Cell and node geometry for the weighed interpolation.

Figure 1 shows a cell and cell nodes geometry. The vector $\boldsymbol{x}$ gives the position of the evaluation point. The vector $\boldsymbol{x}_{\alpha,\beta,\gamma}$ is the location of the α, β, γ node, where α, β, γ are dummy indices for the $\hat{e}_1$, $\hat{e}_2$, $\hat{e}_3$ directions, respectively. For each cell (e.g., $\boldsymbol{x}_{\alpha,\beta,\gamma}$) node, there exists a unique local node polynomial (e.g., $U_{\alpha,\beta,\gamma}$). Unlike the original Junkins method [20], the node polynomials are not centered at the node location [see Eq. (3)] but, instead, at the midpoint of the neighboring six nodes. Each node polynomial has the argument $\boldsymbol{y}$, which is normalized by $\boldsymbol{D}_{\alpha,\beta,\gamma}$ to be valid from -1 to 1 in each direction. Each cell is part of the valid domain of eight nodes. The choice of centering each node polynomial at the midpoint of its six neighboring nodes is taken to allow Chebyshev measurement spacing during the least-square polynomial fitting, as will be discussed later in the paper. The $w_{\alpha,\beta,\gamma}$ are Hermite weight functions [20] valid in the normalized (0, 1) space. The definition of $\boldsymbol{D}_w$ effectively normalizes the $\boldsymbol{x}_w$ vector to have components between zero and one for use in the weight functions $w_{\alpha,\beta,\gamma}$. The weight functions are always centered at the lower, left, front corner of each cell. The order of the weight functions can be chosen as arbitrarily high to ensure any degree of user-desired continuity or smoothness. The gradient of the potential is obtained by taking partial derivatives of the resulting polynomial, and higher-order derivatives are available at a modest cost. Fitting only the potential function is memory efficient and leads to an exact formulation, in the sense that accelerations and higher-order derivatives are calculated as exact derivatives of the interpolated function. In contrast, many previous models [20,23,25,26] ignore exactness constraints, and they fit directly the acceleration and any higher-order derivatives [28] leading to increased memory requirements.

$$\begin{aligned}
U_F &= \sum_{\alpha=i}^{i+1}\sum_{\beta=j}^{j+1}\sum_{\gamma=k}^{k+1} w_{\alpha,\beta,\gamma}(\boldsymbol{x}_w)U_{\alpha,\beta,\gamma}(\boldsymbol{y}) \\
\boldsymbol{x}_w &= \boldsymbol{D}_w^{-1}(\boldsymbol{x}-\boldsymbol{x}_{i,j,k}) \\
\boldsymbol{y} \equiv \boldsymbol{x}_{u_{\alpha,\beta,\gamma}} &= \boldsymbol{D}_{\alpha,\beta,\gamma}^{-1}(\boldsymbol{x}-\boldsymbol{x}_{p_{\alpha,\beta,\gamma}}) \\
\boldsymbol{x}_{p_{\alpha,\beta,\gamma}} &= \left[\frac{(\boldsymbol{x}_{\alpha+1,\beta,\gamma}+\boldsymbol{x}_{\alpha-1,\beta,\gamma})\cdot\hat{e}_1}{2}, \frac{(\boldsymbol{x}_{\alpha,\beta+1,\gamma}+\boldsymbol{x}_{\alpha,\beta-1,\gamma})\cdot\hat{e}_2}{2}, \frac{(\boldsymbol{x}_{\alpha,\beta,\gamma+1}+\boldsymbol{x}_{\alpha,\beta,\gamma-1})\cdot\hat{e}_3}{2}\right]^T \\
\boldsymbol{D}_w &= \begin{bmatrix} (\boldsymbol{x}_{i+1,j,k}-\boldsymbol{x}_{i,j,k})\cdot\hat{e}_1 & 0 & 0 \\ 0 & (\boldsymbol{x}_{i,j+1,k}-\boldsymbol{x}_{i,j,k})\cdot\hat{e}_2 & 0 \\ 0 & 0 & (\boldsymbol{x}_{i,j,k+1}-\boldsymbol{x}_{i,j,k})\cdot\hat{e}_3 \end{bmatrix} \\
\boldsymbol{D}_{\alpha,\beta,\gamma} &= \begin{bmatrix} \frac{(\boldsymbol{x}_{\alpha+1,\beta,\gamma}-\boldsymbol{x}_{\alpha-1,\beta,\gamma})\cdot\hat{e}_1}{2} & 0 & 0 \\ 0 & \frac{(\boldsymbol{x}_{\alpha,\beta+1,\gamma}-\boldsymbol{x}_{\alpha,\beta-1,\gamma})\cdot\hat{e}_2}{2} & 0 \\ 0 & 0 & \frac{(\boldsymbol{x}_{\alpha,\beta,\gamma+1}-\boldsymbol{x}_{\alpha,\beta,\gamma-1})\cdot\hat{e}_3}{2} \end{bmatrix}
\end{aligned} \tag{3}$$

It is worth reemphasizing that the Junkins et al. weighting function scheme [21] allows for arbitrary functions to be used for the local node interpolants. In the current study, simple polynomials are chosen to minimize runtime. The number of coefficients in each node polynomial is allowed to vary adaptively in each direction in order to maximize efficiency (defined as high accuracy and low memory).

The process of generating and using the Fetch model is implemented in two phases. Phase 1 is performed once offline and consists of parallel computation of the local node polynomial coefficients on a global 3-D grid. To deal with the singularity at the poles when working in the geocentric coordinates, a second rotated global grid is introduced. Details on the singularity problem and the rotated grid solution are given in the next section. Phase 2 is the runtime interpolation and involves a weighted evaluation of the local node polynomials. In practice, users of a Fetch model only interface with the runtime interpolation. The order of the weight functions (and, accordingly, the degree of continuity of the final composite function across the boundaries) is independent from the coefficient generation and can be selected at runtime. The next section gives a brief overview of the coefficient generation process.

III. Phase 1: Coefficient Generation

The Fetch interpolation model relies on a global discretization scheme for achieving adaptivity and continuity [21]. The whole solution space is divided into 3-D cells in the spherical coordinate system consisting of polar angle δ, azimuth λ, and radial r directions. The geopotential within each cell is fit (in a least-squares sense) via node polynomials that are locally valid over the eight cells touching that node. Hence, the problem of fitting the global geopotential is reduced to finding localized polynomial coefficients corresponding to each cell node via least-squares fitting.

A. Surface Discretization

Spherical coordinates are used for discretization of the whole solution domain outside the sphere given by the Earth's mean radius. The weighted interpolation scheme (used to achieve global continuity) requires a uniform discretization in each dimension. In other words, the cutting planes that partition the global domain into local cells must all be mutually orthogonal, although the spacing between the planes can vary. Hence, the spacing of the cutting planes [specified by the diagonal components of $\boldsymbol{D}_w$ in Eq. (3)] is a degree of freedom and can be used to improve efficiency. The polar angle, azimuth (δ, λ) space is ultimately chosen to have equal spacing for simplicity. Furthermore, the adaptive degree selection of the local polynomials affords the rationale that the smaller cells (in the Cartesian sense) near the poles will require polynomials of lower order to achieve the same accuracy as the larger cells near the equator. In addition, the two-grid strategy used to remove the singularity also serves to keep a more uniform grid size in the δ, λ space when mapped to the surface. The cutting plane spacing in the radial direction is highly adaptive, as the rapid undulations near the surface require shallow spacing, whereas the high altitudes can be modeled with relatively distant spacings.

B. Singularity

A spherical coordinates-based grid suffers from singularities at the poles when converting to and from the Cartesian coordinate space. A variety of methods is possible for dealing with the singularity, such as that used by Pines [1]. To avoid the burden of the extra dimension, we choose to remain with spherical coordinates but use a rotated grid near the polar regions. Similarly, the cubed-sphere model uses rotated mappings on the polar region "cube faces" to handle the singularity [28]. In the current approach, the global grid is divided into two subgrids (the primary grid and the rotated grid), both in the classic spherical coordinates. The primary grid suffers from singularities at the poles, but its domain of validity is constrained to lie sufficiently far from both poles. Technically, according to [2] and from observation, the main grid is valid and numerically stable using spherical coordinates in all regions outside of a few degrees away from the poles. However, in this application, we enforce a more conservative domain, in order to avoid the small surface patches (and thereby save memory) that result from the uniform (δ, λ) grid. For maximum memory efficiency, we find that a polar angle range of $x < \delta < (180 - x)$ is reasonable, where x can be chosen anywhere from ~25 to ~50 deg. Starting from the poles, a rotated grid is designed that has singularities at two points on the equator of the original grid (due to a 90 deg rotation about the x axis). Hence, the rotated grid serves to remove the singularity at the poles in the original grid. Precise continuity between the two grid submodels is maintained via a Hermite weighting function applied in a narrow overlapping band, as illustrated in Figs. 2 and 3, and will be discussed in detail later.

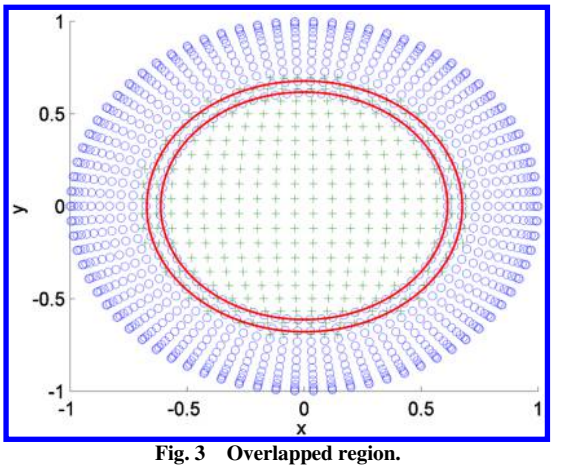

Fig. 3 Overlapped region.

Figure 2 shows the final surface discretization for the primary and the rotated grids. The active region for both the grids is shown as the darker shade, whereas the inactive region is lighter. The overlapping regions are indicated by the horizontal rings (and bounded by the solid lines in Fig. 3). Evaluation inside the overlap region requires that both submodels are computed and a one-dimensional (1-D) weighting function dependent only on the polar angle is used to ensure continuity across the submodel boundaries.

Note that, for the coefficient generation phase, each submodel only requires coefficients for its respective domain, including the overlapped areas. Therefore, the two-submodel strategy creates almost no overhead during coefficient generation. Figure 3 shows a projection onto the Cartesian xy plane of the primary grid (shown with circles "○") overlaid with a rotated grid (shown with plus signs "+") along with the overlapped region (bounded by the solid lines).

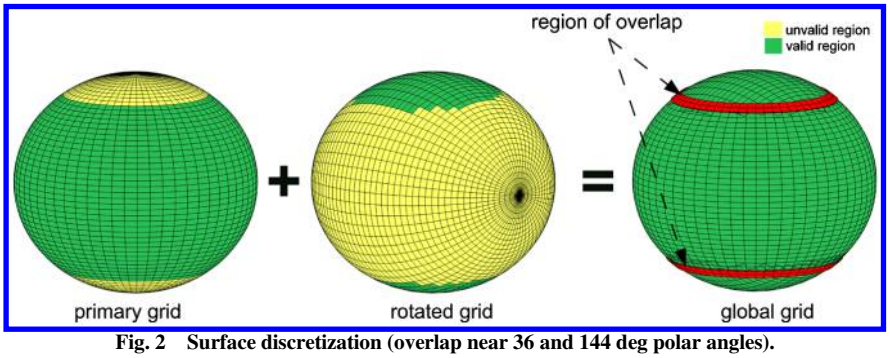

Fig. 2 Surface discretization (overlap near 36 and 144 deg polar angles).

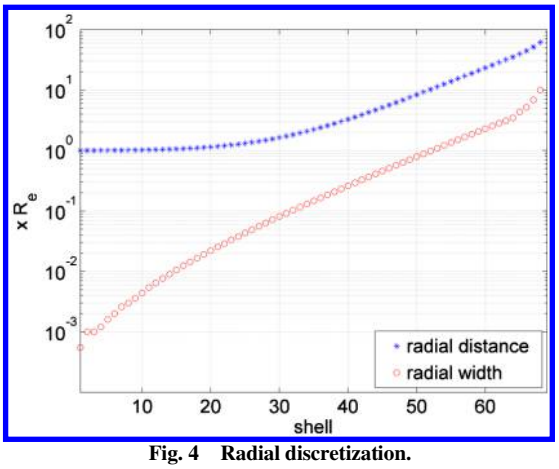

Fig. 4 Radial discretization.

C. Radial Discretization

Unevenly spaced shells are used to space the cutting planes in the radial direction extending all the way to the distance (approximately $230R_e$ for the GGM03C model) beyond which the contribution from the SH terms higher than J_2 becomes less than machine double precision. The shells are densely packed near the surface where the field changes rapidly, and they spread out as altitude increases. Having closely packed shells decreases the number of coefficients per cell but increases the number of cells required for the global model. On the other hand, choosing a large radial step size increases the number of coefficients per cell, making runtime function evaluation slower. Hence, there is a tradeoff between runtime memory and speed requirements. Figure 4 shows the radial shell placement and width selected for the model used in this study, consisting of 68 shells. Starting from a log-based distribution, manual tuning (trial and error) is performed to obtain the final shell spacing.

D. Adaptive Polynomial Selection

The Junkins weighted interpolation approach [20] affords the freedom to choose any degree polynomial for the local approximation functions for a given node. This benefit is used in the polynomial fitting process by adaptively choosing a different degree polynomial for each cell node (see Fig. 1):

$$Q_d = \sum_{a=0}^{d} \sum_{b=0}^{a} \sum_{c=0}^{a-b} C_\sigma y_1^{a-b-c} y_2^c y_3^b \tag{4}$$

Equation (4) gives a general 3-D polynomial considered during the fitting process. Vector $\boldsymbol{y}$ represents the normalized (−1 to 1) position of the evaluation point, $\boldsymbol{C}$ is the coefficients array, C_σ stands for an element of $\boldsymbol{C}$, σ is the total cumulative loop counter, and d stands for the total degree of the polynomial. One approach to select the basis polynomial is to pick a value of d after a careful trade study and then directly use the polynomial obtained from Eq. (4) over the whole solution domain. Junkins [20] and Junkins and Engels [22] adopted this strategy during early pioneering studies [20,22]. Given the computation power present at that time and the intended application of his models, it was the most practical choice.

In our case, we want a broad range of candidate polynomials in order to finely tune the memory requirements of the resulting global model. Inserting $d = 10$ into Eq. (4) results in a polynomial with $N = 286$ coefficients. A set of candidate polynomials can be extracted from the general polynomial by selecting any nonrepeating subset of the 286 terms. As an example, the number of possible ways to generate a set of 200 candidate polynomials (with no repeating coefficients) is $286!/200!(286-200)!$ or on the order of $\sim\mathcal{O}(10^{74})$. To overcome this intractable problem, we extract a subset of polynomials by incrementally removing the highest-order terms (total degree equal d) subject to a dependency on r. Removing such terms is justified because the radial grid spacing is variable, thus partially absorbing the need for adaptivity in those directions. Note that the summations in Eq. (4) are intentionally ordered such that the removed terms are always the trailing terms:

$$\begin{aligned} Q_2 = {} & C_1 + C_2 y_1 + C_3 y_2 + C_4 y_3 + C_5 y_1^2 + C_6 y_1 y_2 + C_7 y_2^2 \\ & + C_8 y_1 y_3 + C_9 y_2 y_3 + C_{10} y_3^2 \end{aligned} \tag{5}$$

The algorithm to generate the candidate polynomial set starts by generating nine major candidate polynomials Q_d from Eq. (4) for d ranging from 2 to 10. Next, we divide each of the nine major polynomials into minor candidate polynomials $Q_{d,z}$ by incrementally removing terms. Equation (5) shows the first major candidate polynomial for $d = 2$. It has four terms that are dependent on y_3, out of which three (the three of total degree two) are removed in succession to generate three minor candidate polynomials [Eq. (6)]. We only need to delete the leading terms because the loop counter b changes slower than the loop counter for c [see Eq. (4)]:

$$\begin{aligned} Q_{2,3} = {} & C_1 + C_2 y_1 + C_3 y_2 + C_4 y_3 + C_5 y_1^2 + C_6 y_1 y_2 + C_7 y_2^2 \\ & + C_8 y_1 y_3 + C_9 y_2 y_3 \\ Q_{2,2} = {} & C_1 + C_2 y_1 + C_3 y_2 + C_4 y_3 + C_5 y_1^2 + C_6 y_1 y_2 + C_7 y_2^2 \\ & + C_8 y_1 y_3 \\ Q_{2,1} = {} & C_1 + C_2 y_1 + C_3 y_2 + C_4 y_3 + C_5 y_1^2 + C_6 y_1 y_2 + C_7 y_2^2 \end{aligned} \tag{6}$$

Equation (6) shows the three minor polynomials $Q_{2,3}$, $Q_{2,2}$, and $Q_{2,1}$ corresponding to the major polynomial Q_2. The formula to calculate the number of terms to be removed n_d from a major polynomial of degree d is given by Eq. (7):

$$n_1 = 1 \quad n_i = n_{i-1} + i \qquad \forall\ i = 2 \ldots d \tag{7}$$

The total number of minor polynomials equals to the sum of the number of terms removed for each major polynomial. This results in a

Algorithm 1 Candidate polynomial set generation

```
1: procedure POLYGET(O_max)        ▷ Input highest degree of the candidate polynomianl ≥ 2
2:   i←0
3:   n(1)←1
4:   for j = 2 to O_max, do        ▷ Major polynomial loop
5:     n(j)←n(j − 1) + j           ▷ Calculate number of terms to be dropped [Eq. (7)]
6:     for k = n(j) − 1 to 0, do   ▷ Minor polynomial loop
7:       i←i + 1
8:       C_(N−k)...N←0             ▷ Set (N − k) to N coefficients to zero
9:       P_i←Q_j,k                 ▷ Compute minor polynomial with last k + 1 coefficients set to zero
10:    end for
11:    i←i + 1
12:    P_i←Q_j                     ▷ Compute major polynomial with N coefficients
13:  end for
14:  return P_1...i                ▷ The final set of candidate polynomials
15: end procedure
```

total set of 219 minor polynomials. Finally, the sum of 9 major and 219 minor polynomials together gives us a total set of 228 candidate polynomials $P_{1\ldots228}$. Algorithm 1 outlines the complete candidate polynomial generation procedure.

During fitting, each of the candidate polynomials are ordered in terms of the number of total coefficients. Starting with the candidate polynomial with the fewest coefficients (seven in the current study), the least-squares problem is evaluated. If a candidate polynomial solution is found with residuals acceptable according to user-prescribed tolerances, then the polynomial is selected and the process is stopped for that node. Candidate polynomials with a greater number of coefficients are not evaluated. In this manner, the residuals of the local solutions will remain just below the user-supplied tolerance, leading to a uniform global residual distribution. In addition, because the candidate polynomials are evaluated in order of increasing memory footprint, each cell has a minimized memory requirement (for a given cell dimension and grid spacing). Therefore, the coefficients corresponding to each local node are chosen so as to minimize its memory footprint, subject to user-provided residual constraints.

E. Localized Least-Square Approximation

The local approximation for each node polynomial is obtained via a least-squares fit. As stated in the previous section, the minimum and maximum number coefficients allowed are fixed to 7 and 286, respectively. Given a candidate polynomial, the conventional least-squares method for generating the coefficients is summarized by Eq. (8):

$$\boldsymbol{C} = B\boldsymbol{u} \tag{8}$$

Here, $\boldsymbol{C}$ is a $N \times 1$ vector of coefficient estimates, $\boldsymbol{u}$ is a $m^3 \times 1$ vector representing the measurements, and B denotes the $N \times m^3$ least-squares inverse matrix [Eq. (9)]:

$$B = (H^T H)^{-1} H^T \tag{9}$$

The matrix H represents the $m^3 \times N$ sensitivity matrix, m^3 represents the number of measurements (noting there are m measurements in each of the three directions), and N represents the total number of coefficients per node polynomial. The measurements are spaced according to the Chebyshev node distribution function given by Eq. (10) in each direction [30]. This choice of placement helps to minimize the well-known Runge's phenomena and increases the robustness of the fit. As the radial shells are unevenly spaced, centering the node polynomials at the cell boundary (or cell nodes) destroys the Chebyshev measurement spacing and leads to nonuniform residual distributions over the node domain. Hence, to allow for Chebyshev spacing, the node polynomials are centered at the midpoints of the neighboring six nodes as measured from the current node. We note that our method differs from Junkins formulation [20], as his polynomials are centered at the cell nodes, which is a special case in our formulation when we choose constant cell sizes. Figure 5 shows the Chebyshev node distribution for $m = 11$ for a normalized two-dimensional (2-D) node domain:

$$x_i = \cos\left(\frac{2i-1}{2\ m}\pi\right), \qquad i = 1\ldots m \tag{10}$$

The H matrix contains the first-order partials of the candidate polynomial with respect to the estimated coefficients. The functional form for the rth row of H is given as follows:

$$H_{(r,:)} = \left.\frac{\partial P_i}{\partial \boldsymbol{C}}\right|_{\rho_r} \tag{11}$$

where $\boldsymbol{\rho}_r$ is the measurement location, and P_i is the ith candidate polynomial. Computing the analytic partials and evaluating them at the specific measurement locations, the full H matrix for each candidate polynomial is computed using a symbolic manipulator program (Maple) and results in a $m^3 \times N$ matrix of rational fraction entries.

Each node polynomial is normalized between -1 to 1 along each of its dimensions in its node domain. If the number of measurements m^3 and their respective positions (defined by the Chebyshev spacing) do not change; then, the fully evaluated H matrix is invariant across all nodes for a given candidate polynomial. This static property of the H matrix (as pointed out by Junkins [20] and Lekien and Marsden [24]) is only true if the number of measurements and their relative positions ($\boldsymbol{\rho}_r$ for all measurements) are the same across all node domains. Given that the H matrix is static, the inversion matrix that solves the least-squares problem in Eq. (9) can be computed analytically just once. In other words, once the B matrix is solved analytically, there is no need to numerically solve a linear system in order to obtain the least-squares coefficient fit for a given candidate function. Therefore, in order to test a single candidate polynomial across the global domain, each cell simply requires one matrix multiplication [Eq. (8)]. It is further emphasized that the H matrix, when evaluated with a symbolic manipulator is composed of rational fractions. Accordingly, B from Eq. (9) can be analytically computed and the results are also in the form of rational fractions. Typical numerical problems associated with solving ill-conditioned linear systems are avoided. Quoting from [22], the "one inverse property can lead to order of magnitude savings in computer time for large data sets."

As an example, the matrix B for the candidate polynomial having seven coefficients ($N = 7$) and with $m = 11$ is dense with explicit cosine functions present in most of its terms [owing to the Chebyshev spacing from Eq. (10)]. The cosine function is accurate at least up to the IEEE floating point standard. Furthermore, the cosine terms are computed in Maple at an extended precision, and the result is stored to 17 digits. Equation (12) gives the first row of this matrix. The actual size of the matrix is 7×11^3:

$$B(1,:) = \left[\begin{array}{c} \frac{7}{14641} - \frac{36}{14641}\cos(1/11\pi) + \frac{8}{14641}\cos(3/11\pi) + \frac{8}{14641}\cos(\frac{5}{11}\pi) - \frac{8}{14641}\cos(4/11\pi) - \frac{8}{14641}\cos(2/11\pi) \\ -\frac{2}{1331}\cos(1/22\pi) \\ -\frac{2}{1331}\cos(1/22\pi) \\ -\frac{2}{1331}\cos(1/22\pi) \\ \frac{4}{14641} + \frac{36}{14641}\cos(1/11\pi) - \frac{8}{14641}\cos(3/11\pi) - \frac{8}{14641}\cos(\frac{5}{11}\pi) + \frac{8}{14641}\cos(4/11\pi) + \frac{8}{14641}\cos(2/11\pi) \\ \frac{2}{1331}\cos(1/11\pi) + \frac{2}{1331} \\ \frac{4}{14641} + \frac{36}{14641}\cos(1/11\pi) - \frac{8}{14641}\cos(3/11\pi) - \frac{8}{14641}\cos(\frac{5}{11}\pi) + \frac{8}{14641}\cos(4/11\pi) + \frac{8}{14641}\cos(2/11\pi) \end{array}\right]^T \tag{12}$$

For this study a Maple worksheet is used to analytically invert the B matrices corresponding to all possible candidate polynomials. The worksheet takes the maximum polynomial degree $O_{\max}$ and m as inputs and writes the B matrices to a file for later use by the coefficient generation routines. Analytic inversion of the normal matrix $H^T H$ of

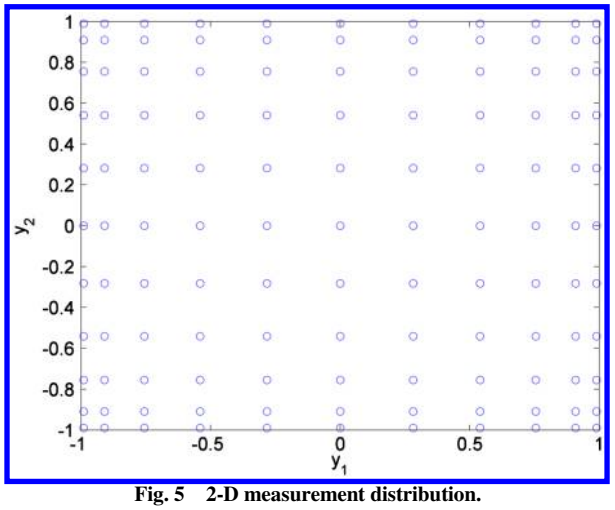

Fig. 5 2-D measurement distribution.

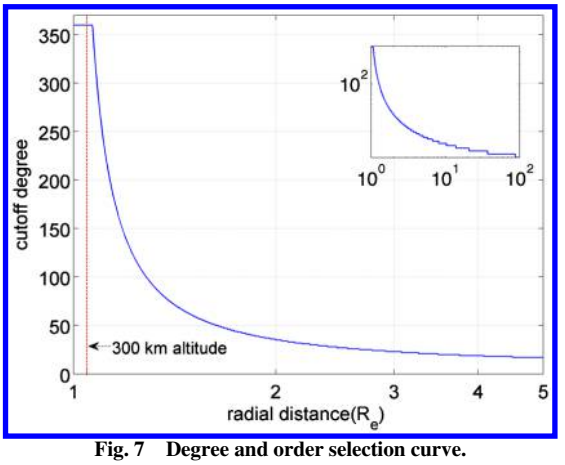

Fig. 7 Degree and order selection curve.

dimension $N \times N$ is computationally intensive for large values of N (100 or more) and may suffer from numerical ill conditioning. The analytical approach provides us with the best possible method to tackle any ill conditioning if present. Numerical roundoff errors (though reduced) will still be present, but the numerical ill conditioning arising from inverting an ill-conditioned matrix is avoided. It is emphasized that these inversion matrices only need to be computed just once for a given interpolant and a fixed measurement spacing scheme. The number of observations in each direction is kept at a minimum of $O_{\max} + 1$ in order to preserve a well-posed (overconstrained) least-squares problem. Once the measurements are obtained and the measurement vector is formed, only a single matrix multiply is needed to obtain the least-squares coefficients. Exact and precomputed B matrices allows for a fast, accurate, and numerically well-conditioned coefficient generation process.

For the current study, the value of m is fixed at 11 for all the node domains and the maximum polynomial degree ($O_{\max}$) is fixed at 10. Figure 6 gives the structure of $(H^T H)^{-1}$ matrix for $m = 11$ and $N = 286$. Here, the black dots represent the nonzero terms in the matrix. The size of this matrix is 286×286. Maple version 13 was used for this study, and it took approximately 36 h on a single workstation to generate the B matrices for all 228 candidate polynomials.

F. Scalable SH Degree Selection

Once all the B matrices are generated, the measurement $\boldsymbol{u}$ for each node domain must be computed in order to obtain the coefficients for all the candidate functions using Eq. (8). The measurement vector requires evaluation of the SH function at each of the m^3 locations within each node domain. Noting that there are ~8 million node domains for the global 360×360 resolution case, obtaining the measurement vector for $m = 11$ requires evaluation of the SH function on the order of 10 billion times. Fortunately, the full-precision SH field is not necessary at all altitudes, as the high-order terms become undetectable to machine precision as the altitude increases.

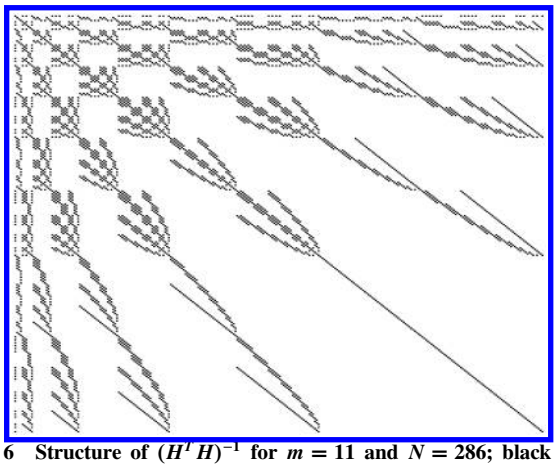

Fig. 6 Structure of $(H^T H)^{-1}$ for $m = 11$ and $N = 286$; black dots denote nonzero terms.

Accordingly, a scalable SH degree selection method is adopted. Figure 7 shows the altitudes where the contributions of higher-order terms of the SH expansion (given by the GGM03C model) become less than a normalized 10^{-15} (near-machine precision for double). For example, at a radial distance of $10R_e$ (equivalently at an altitude of $9R_e$), only terms up to the degree and order of 30 are necessary for the computation of the measurement vector. Using 10^{-15} as a cutoff tolerance is overly conservative considering that the maximum normalized residuals in the target SH field fits (to be discussed later) are always substantially larger. The equation used to generate the curve in Fig. 7 is given in Eq. (13) and is found via a nonlinear least-squares curve fit of a large sampling of measurements taken across the global domain. Equation (13) takes in normalized radial distance r as input and returns the degree and order value F_{siz}, which is the maximum degree and order SH gravity field necessary for that radial distance:

$$F_{\text{siz}} = \min\left[360, \text{floor}\left(\frac{Z_1}{\ln(r)^{Z_2} + Z_3} + Z_4\right) + 1\right] \tag{13}$$

where the coefficient values are given in Table 1. It is evident from Fig. 7 that, for most of the global domain, the high-order spherical harmonics terms are not significant, even in double-precision arithmetic. As the evaluations move radially outward, the measurement vector computation time decreases rapidly, thereby significantly speeding up the coefficient generation process. Note that Eq. (13) can be used to speed up the computation of SH for general applications as well.

G. Continuity

The Fetch model achieves continuity in any order by using the weighted interpolation scheme (developed by Junkins [20] and Junkins et al. [21]), which leads to eight interpolant function evaluations instead of one for each composite function call. The eight interpolants applicable to each cell are evaluated in their own -1 to 1 normalized domain, and the weighting is done in the overlapping space inside a cell, normalized from 0 to 1. The normalization allows the use of Hermite weighting functions, which enable continuous higher-order derivatives. The weight functions as they appear in Eq. (3) can be found in equations 5a–5g of [20]. Equation (14) summarizes the final Fetch approximation for the potential and first derivative with respect to x_1, x_2, and x_3, which are the spherical components of position vector defining the evaluation point. $\boldsymbol{C}$ stands for the coefficients of the node polynomial under consideration; $w_{i,j,k}$ are the Hermite weight functions at the i, j, k node as specified by

Junkins [20]; $\boldsymbol{x}_w$ is the normalized position vector having components between 0 and 1 [see Eq. (3)]; and $N_{i,j,k}$ is the number of coefficients for the i, j, k node polynomial. These partial derivatives are further converted into final Cartesian acceleration by applying the inverse of the classic spherical coordinate transformation. Higher-order derivatives are obtained by taking further derivatives of Eq. (14) and applying the chain rule:

$$
\begin{aligned}
U_F &= \sum_{\alpha=i}^{i+1}\sum_{\beta=j}^{j+1}\sum_{\gamma=k}^{k+1} w_{\alpha,\beta,\gamma}(\boldsymbol{x}_w)U_{\alpha,\beta,\gamma}(\boldsymbol{y}) \\
\frac{\partial U_F}{\partial x_1} &= \sum_{\alpha=i}^{i+1}\sum_{\beta=j}^{j+1}\sum_{\gamma=k}^{k+1}\left[\frac{\partial w_{\alpha,\beta,\gamma}(\boldsymbol{x}_w)}{\partial \boldsymbol{x}_w}\frac{\partial \boldsymbol{x}_w}{\partial x_1}U_{\alpha,\beta,\gamma}(\boldsymbol{y}) + w_{\alpha,\beta,\gamma}(\boldsymbol{x}_w)\frac{\partial U_{\alpha,\beta,\gamma}(\boldsymbol{y})}{\partial \boldsymbol{y}}\frac{\partial \boldsymbol{y}}{\partial x_1}\right] \\
\frac{\partial U_F}{\partial x_2} &= \sum_{\alpha=i}^{i+1}\sum_{\beta=j}^{j+1}\sum_{\gamma=k}^{k+1}\left[\frac{\partial w_{\alpha,\beta,\gamma}(\boldsymbol{x}_w)}{\partial \boldsymbol{x}_w}\frac{\partial \boldsymbol{x}_w}{\partial x_2}U_{\alpha,\beta,\gamma}(\boldsymbol{y}) + w_{\alpha,\beta,\gamma}(\boldsymbol{x}_w)\frac{\partial U_{\alpha,\beta,\gamma}(\boldsymbol{y})}{\partial \boldsymbol{y}}\frac{\partial \boldsymbol{y}}{\partial x_2}\right] \\
\frac{\partial U_F}{\partial x_3} &= \sum_{\alpha=i}^{i+1}\sum_{\beta=j}^{j+1}\sum_{\gamma=k}^{k+1}\left[\frac{\partial w_{\alpha,\beta,\gamma}(\boldsymbol{x}_w)}{\partial \boldsymbol{x}_w}\frac{\partial \boldsymbol{x}_w}{\partial x_3}U_{\alpha,\beta,\gamma}(\boldsymbol{y}) + w_{\alpha,\beta,\gamma}(\boldsymbol{x}_w)\frac{\partial U_{\alpha,\beta,\gamma}(\boldsymbol{y})}{\partial \boldsymbol{y}}\frac{\partial \boldsymbol{y}}{\partial x_3}\right] \\
U_{\alpha,\beta,\gamma}(\boldsymbol{y}) &= \sum_{a=0}^{O_{\max}}\sum_{b=0}^{a}\sum_{c=0}^{a-b} C_\sigma y_1^{a-b-c} y_2^c y_3^b \qquad \forall\ \sigma \le N_{\alpha,\beta,\gamma} \\
\frac{\partial U_{\alpha,\beta,\gamma}(\boldsymbol{y})}{\partial y_1} &= \sum_{a=1}^{O_{\max}}\sum_{b=0}^{a}\sum_{c=0}^{a-b} C_\sigma (a-b-c) y_1^{a-b-c-1} y_2^c y_3^b \qquad \forall\ \sigma \le N_{\alpha,\beta,\gamma}, a-b-c>0 \\
\frac{\partial U_{\alpha,\beta,\gamma}(\boldsymbol{y})}{\partial y_2} &= \sum_{a=0}^{O_{\max}}\sum_{b=0}^{a}\sum_{c=1}^{a-b} C_\sigma c y_1^{a-b-c} y_2^{c-1} y_3^b \qquad \forall\ \sigma \le N_{\alpha,\beta,\gamma} \\
\frac{\partial U_{\alpha,\beta,\gamma}(\boldsymbol{y})}{\partial y_3} &= \sum_{a=0}^{O_{\max}}\sum_{b=1}^{a}\sum_{c=0}^{a-b} C_\sigma b y_1^{a-b-c} y_2^c y_3^{b-1} \qquad \forall\ \sigma \le N_{\alpha,\beta,\gamma}
\end{aligned} \tag{14}
$$

Figure 8 summarizes the weighting function technique for two equally spaced dimensions. Equation (15) shows the final weighted evaluation of the potential function in the small region of subgrid overlap (see Fig. 3):

$$U_\Omega = U_{\text{pri}}(\delta,\lambda,r)\Omega(\phi) + [1-\Omega(\delta)]U_{\text{rot}}(\delta,\lambda,r) \tag{15}$$

In Eq. (15), δ refers to normalized polar angle, which varies between zero and one across the overlapped region. U_{pri} and U_{rot} are the interpolated geopotentials evaluated on the primary and the rotated grids, respectively. U_Ω is the final weighted geopotential in the overlapping region, and Ω represents a seventh degree 1-D Hermite weight function that achieves third-order continuity (determined by the outer term δ^4) and is given by Eq. (16):

$$\Omega(\delta) = \delta^4(35 - 84\delta + 70\delta^2 - 20\delta^3) \tag{16}$$

Equations (15) and (16) can be further differentiated with respect to δ using the chain rule and by using Eq. (3) to obtain continuous derivatives up to the desired order in the overlap region. Hence, due to the weighting evaluation and the overlapping techniques, continuity and exact higher-order derivatives of the interpolated geopotential are maintained globally. Exact higher-order derivatives possess attractive dynamical properties, especially for applications such as orbit determination and trajectory optimization. Figure 9 illustrates the continuity by showing the potential and its first three derivatives with respect to Cartesian x for the subgrid overlapping region. Figure 9 (left) illustrates continuity in all four cases, achieved by choosing a seventh-degree Hermite weighting function that is continuous up to the third order (smooth up to the second order). Kinks and discontinuities in the higher-order derivatives are illustrated in Fig. 9 (right) and are a result due to the use of a lower-order (first-order continuous) weight function.

H. Residual Tolerances

The final acceptable residual level (when compared to SH fitting function) for each node domain is a function of four subtolerances: the rms and maximum of the potential residual and the rms and maximum of the acceleration residuals (norm of the difference of the Cartesian acceleration vectors). The potential residual tolerance is adaptively determined based on the radial distance of the current point in the node domain and the degree and order of the SH function being fit. The target value is chosen to conservatively mirror the expected errors of the SH function. The estimated accuracies of the GGM03C solution are given in [4] and the associated release notes. The accumulated error as a function of SH degree is replicated in Fig. 10. For example, up to the degree and order of 70, the accumulated error for the geoid height is $\sim$6 mm or $\sim 10^{-9}$ in normalized units. Up to the degree and order of 360, the accumulated normalized error is $\sim 3\times 10^{-8}$. Therefore, the confidences of the potential evaluations at the surfaces of 360×360 and 70×70 fields are approximately eight and nine digits of accuracy, respectively. The accumulated error curve in Fig. 10 serves as a baseline target for the interpolation residuals for geopotential evaluation at the surface. To map the target residuals to different altitudes, Eq. (17) includes the (R_e/r) term similar to the structure of the interpolant. For a given size of the SH field, the covariance matrix terms (σ_c and σ_s) from the GGM03C solution are obtained [4]. To be conservative, the baseline target potential residual τ_U is always kept below a user-defined constant η_0, which is equal to 5×10^{-9} for the current study:

$$\tau_U(r) = \min\left(\eta_0, \sqrt{\left[\sum_{i=0}^{d}\left(\frac{R_e}{r}\right)^{2i}\sum_{j=0}^{i}(\sigma_c(i,j)^2 + \sigma_s(i,j)^2)\right]}\right) \tag{17}$$

For a given degree and order of the SH field d, a residual scaling graph is obtained using Eq. (17). Figure 11 illustrates the curve for various degree and order truncations. The R_e/r term dominates at the high altitudes, making the baseline target residuals approximately the

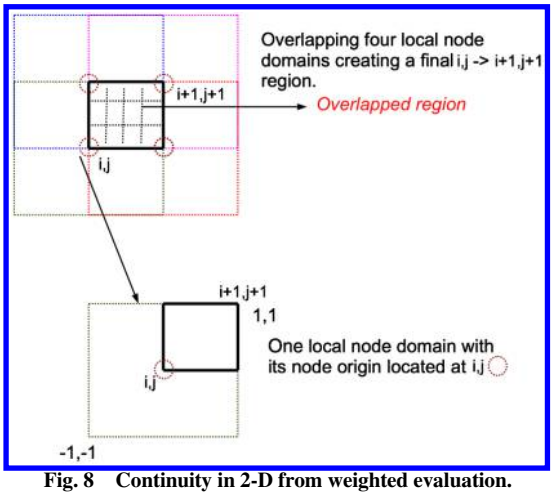

Fig. 8 Continuity in 2-D from weighted evaluation.

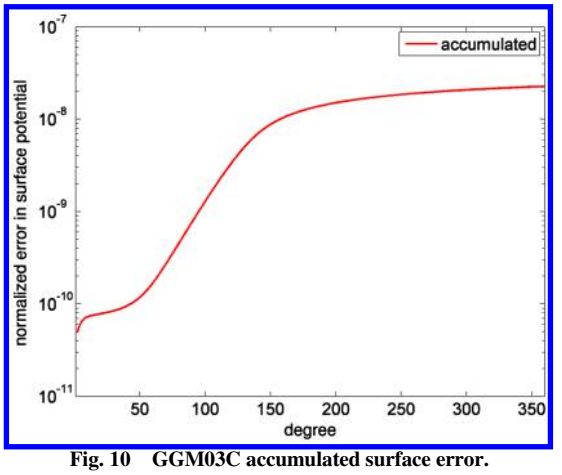

Fig. 10 GGM03C accumulated surface error.

same for all four cases shown. This adaptive scaling in the radial direction provides a context for targeting interpolation residuals, allowing for a highly optimized memory footprint of the global model. The baseline target value of τ_U multiplied by a user-defined constant (η_1) serves as the final potential rms residual tolerance used during the fitting process. The acceleration rms residual tolerance is directly obtained by multiplying the potential rms residual tolerance by a user-defined constant (η_2). The maximum residual tolerances on both the potential and acceleration are obtained by multiplying the corresponding per-node domain rms values (computed during the fitting process), by user-defined constants, η_3 and η_4, respectively. Equation (18) gives the four subtolerances that are required to be met in order for a candidate polynomial to be selected. Here, $\Upsilon_{1\ldots4}$ represent the final potential rms, acceleration rms, potential maximum, and the acceleration maximum residual tolerances, respectively. It should be noted that κ_U and $\kappa_{\nabla U}$ denote the per-node domain rms values for the potential and acceleration, respectively:

$$\begin{aligned}\Upsilon_1 &= \tau_U \eta_1\\ \Upsilon_2 &= \Upsilon_1 \eta_2\\ \Upsilon_3 &= \kappa_U \eta_3\\ \Upsilon_4 &= \kappa_{\nabla U} \eta_4\end{aligned} \tag{18}$$

It can be argued that, because the residuals of the interpolation model are within the published accuracy of the SH model, neither the fitted SH model nor the interpolation model is more likely to represent the true geopotential. Despite such an argument, the additional constraint to limit residuals in the accelerations is included in order to maintain consistency in a uniform manner for the acceleration results from both the GGM03C SH and the Fetch models. No constraints are placed on higher-order derivatives because there is no justification for exact consistency. To the contrary, applications such as orbit determination and trajectory optimization require exact derivatives of the function being used and not the function being approximated.

I. Parallel Coefficient Generation

In spite of various algorithmic optimizations, as stated in the previous sections, the coefficient generation process for a complete

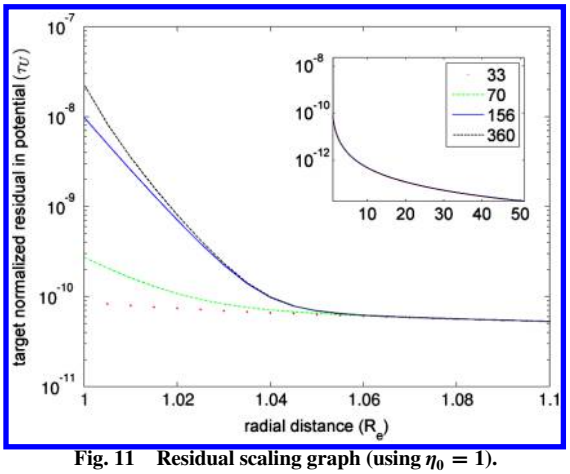

Fig. 11 Residual scaling graph (using $\eta_0 = 1$).

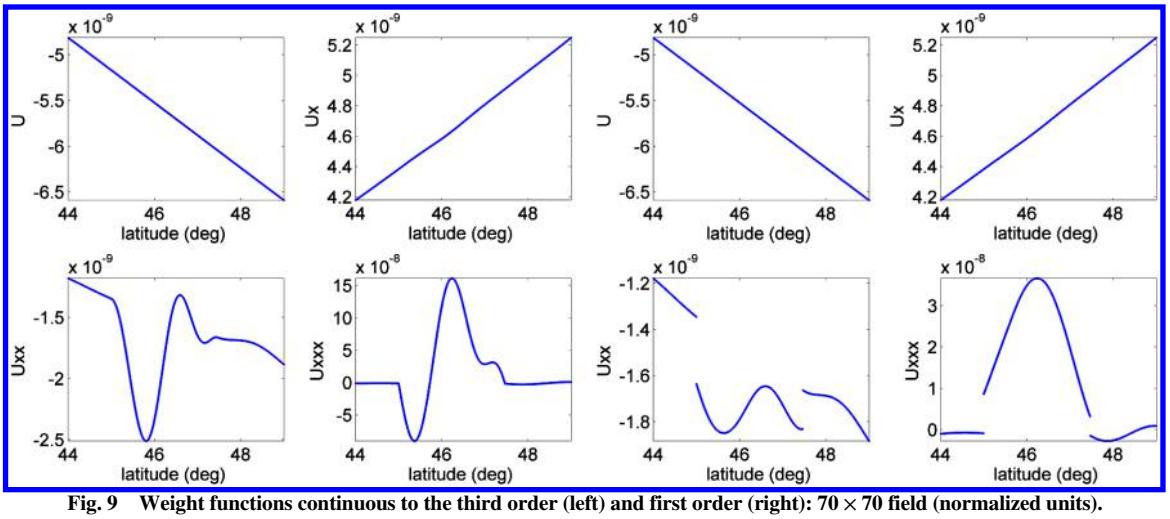

Fig. 9 Weight functions continuous to the third order (left) and first order (right): 70×70 field (normalized units).

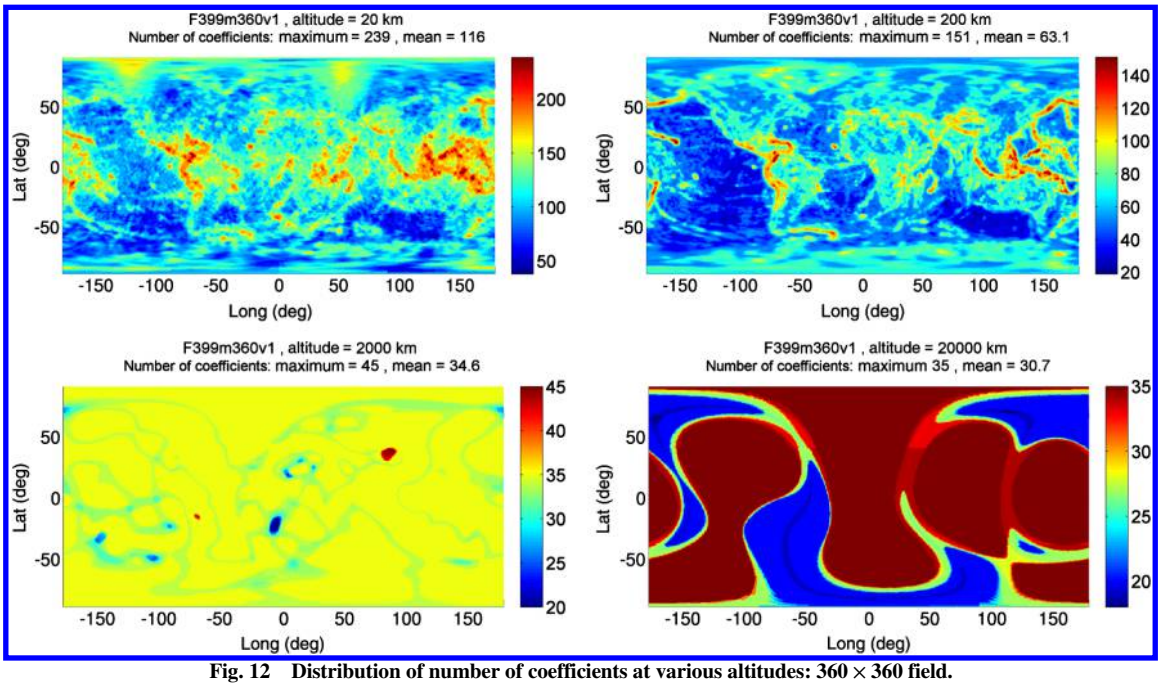

Fig. 12 Distribution of number of coefficients at various altitudes: 360 × 360 field.

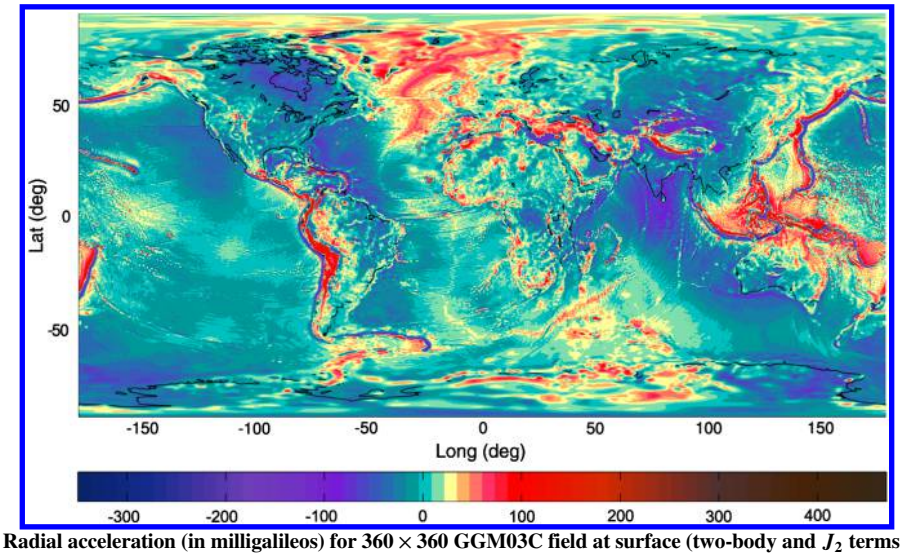

Fig. 13 Radial acceleration (in milligalileos) for 360 × 360 GGM03C field at surface (two-body and J_2 terms removed).

360 × 360 global model would require on the order of 12,000 CPU hours using a modern workstation (2.27 GHz Intel Xeon E5520 processor). Note that the 360 × 360 deg and order model contains approximately eight million nodes, and up to 228 candidate functions are evaluated for each node, where each candidate function contains up to 286 coefficients. Furthermore, each node includes $11^3 = 1331$ measurements, and each SH call can include up to ∼130,000 terms. As the coefficients need to be generated just once, the process is parallelized using the MPI programming model. The main computation burden comes from SH field computation and increases near the surface where higher-order terms are significant. Depending on the radial distance and the local geopotential characteristics of the node under consideration, the fitting times can vary significantly, leading to nonhomogeneity in computation over the global domain.

To tackle the nonhomogeneity, a parallel algorithm using a master-worker strategy is implemented for computing the coefficients. The first CPU thread is made the master thread and is responsible for gathering and distributing work (nodes to be fit) to all other CPUs or worker threads. The worker thread comes back to the master thread after fitting its share of allocated nodes and waits for more nodes. Each worker thread writes its own separate coefficient file, and the master thread is responsible for joining all the files into the final coefficient binary file. The master-worker algorithm leads to a 1.37-fold speedup over the case of static assignments for each thread. The algorithm also has attractive features like load balancing and autoresume, which are useful if computation is interrupted for any reason. The parallel version of the code is implemented in FORTRAN 2003 and can be compiled using either the Intel MPI compiler or the OPEN-MPI compiler. The complete process, as described, for fitting a 360 deg and order field across the full domain

Table 1 Parameter values in Eq. (13)

Parameter	Value
Z_1	20.37185996049265
Z_2	1.165605077780336
Z_3	1.625250344642757E − 02
Z_4	4.461932685300188

Table 2 Fetch models

Model name	SH field	η_1	η_2	η_3	η_4	Γ	Memory, MB	No. cells	Average no. coefficients/cell	Expected speedup
F399m33v1	33×33	1.00	10.00	11.00	10.00	2.00	121	175,536	90.2	3
F399m70v1	70×70	1.00	10.00	11.00	11.00	1.53	360	564,075	83.1	15
F399m156v1	156×156	1.00	12.00	11.00	11.00	1.21	898	2,015,076	57.9	91
F399m360v1	360×360	1.00	20.00	11.00	8.00	0.92	2360	7,615,254	41.0	784

takes 11 h on a cluster of 1100 processors (3.33 GHz Intel Xeon X5680 processor). For the current study, the Texas Advanced Computing Center's Lonestar Linux Cluster is used, consisting of 1888 compute nodes with 6 cores per node, resulting in a total of 22,656 cores and a peak compute performance of 302 trillion floating points per second.

Figure 12 shows a surface distribution for a number of polynomial coefficients of the optimally selected polynomials at altitudes ranging from 20 to 20,000 km for the 360×360 model. At all altitudes, the (δ, λ) grid size remains the same; hence, we observe a reduction in the number of coefficients as altitude is increased. The regions with higher numbers of coefficients correspond to the regions where the geopotential changes rapidly. Figure 13 shows the contour of the radial acceleration [in milligal (mGal) with two-body and J_2 terms removed] evaluated at the surface for 360×360 field using the Fetch model. The computed coefficients for the primary and rotated grids, along with the required metadata, are stored in one dense binary file.

J. Model Classification

Given the SH base field, the Fetch model is generated by specifying the four residual constants (defined previously as $\eta_{1\ldots4}$) and a grid resolution control parameter called Γ:

$$s = \Gamma\sqrt{\frac{180}{d}} \tag{19}$$

In Eq. (19), s is the cell edge length in degrees, and d is the degree and order of the SH base field. Γ provides a dial to control the memory vs speedup trade. The value of Γ directly affects the polynomial fitting accuracy and Fetch runtime performance. A high value results in a final Fetch model with a smaller memory footprint but slower runtime performance, and vice versa. Typical values for Γ lie between 0.5 to 2 for various SH base fields considered in this study. Table 2 lists the various Fetch models generated for this study and are classified based on the degree and order of the SH field that is fit, residual tolerances ($\eta_{1\ldots4}$), and Γ. The naming convention includes the SPICE body number [31], the SH model degree and order, and the version number of the current release. The F, m, and v stand for Fetch, model, and version, respectively. The memory footprint increases almost linearly with the increase in the SH degree and order. The expected speedup is representative of low-altitude orbiter simulations (see Sec. V.C for more details).

Table 3 Available Fetch routines

Routine name	Task performed
init_Fetch	Initializes Fetch + coefficient file
get_Fetch	Computes potential + derivatives

Table 4 Test hardware/software

Component type	Component
CPU	Intel Core i7 950 at 3.07 GHz
RAM (memory)	6.0 GB
Compiler	Intel FORTRAN 12.0
Operating system	Ubuntu 12.10

Table 5 Performance evaluation regions

Band identification	Start altitude, km	End altitude, km
Band 1	36.0	65.0
Band 2	65.0	1,000.0
Band 3	1,000.0	2,550.0
Band 4	2,550.0	6,378.0
Band 5	6,378.0	19,135.0

Figure 14 shows various Fetch models with their memory footprint (left vertical axis) and expected speedup (right vertical axis) as a function of Γ. Two classes of Fetch models corresponding to SH field sizes 33×33 and 70×70 are shown. Increases in both the speedup and memory are observed with a decreasing value of Γ. For fairness, the residual tolerances ($\Upsilon_{1\ldots4}$) within each class of models are kept constant. Only the models satisfying the residual tolerances are plotted, thereby imposing an upper limit on Γ.

IV. Phase 2: Runtime Evaluation

The runtime implementation of a Fetch model is designed to be as simple as possible. The steps required at runtime are 1) initialize

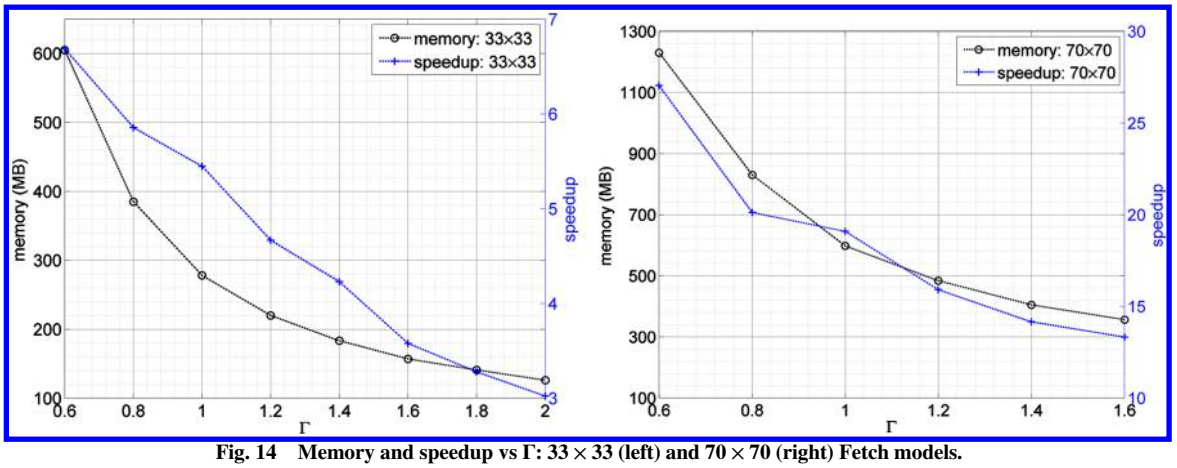

Fig. 14 Memory and speedup vs Γ: 33×33 (left) and 70×70 (right) Fetch models.

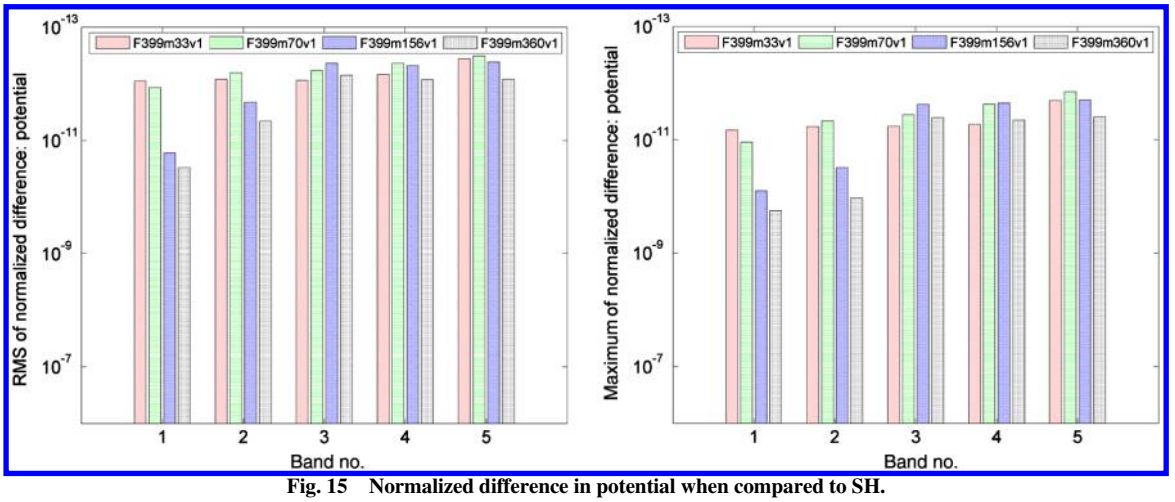

Fig. 15 Normalized difference in potential when compared to SH.

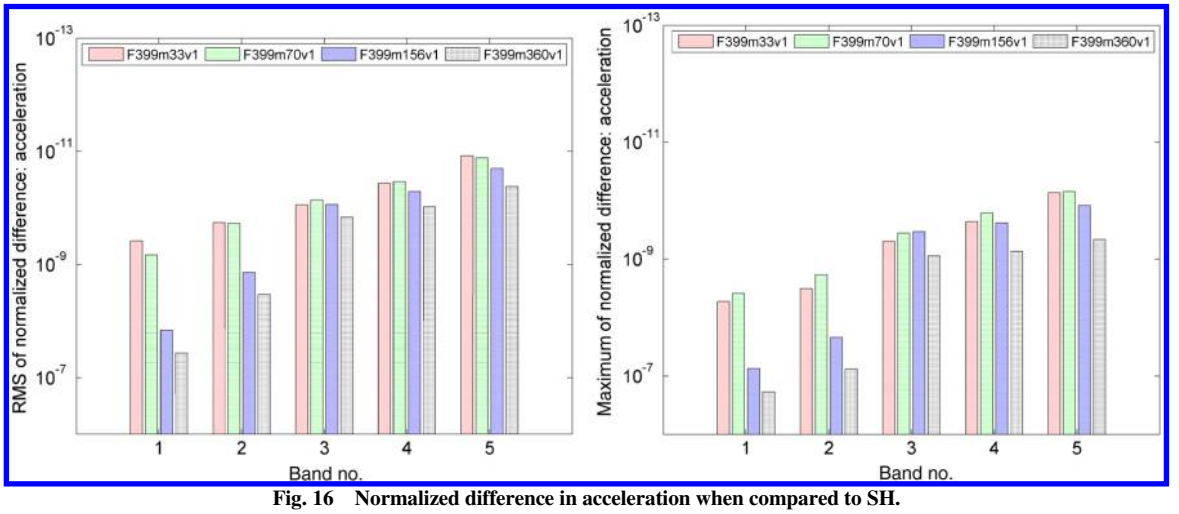

Fig. 16 Normalized difference in acceleration when compared to SH.

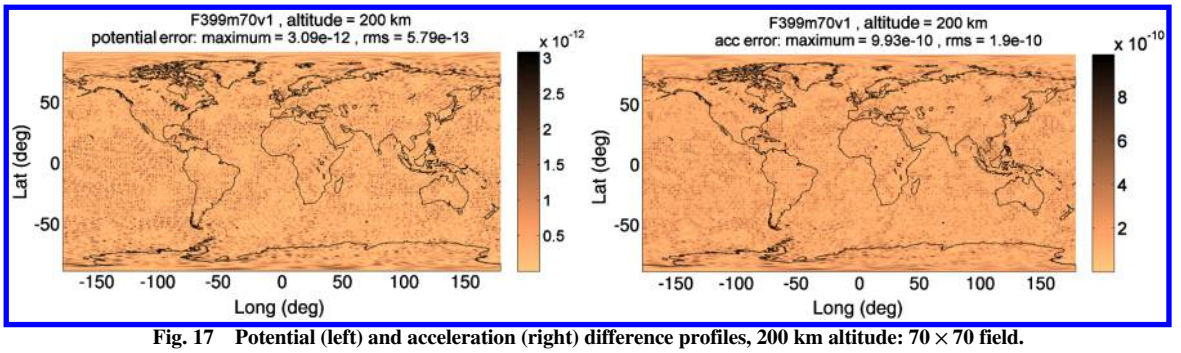

Fig. 17 Potential (left) and acceleration (right) difference profiles, 200 km altitude: 70×70 field.

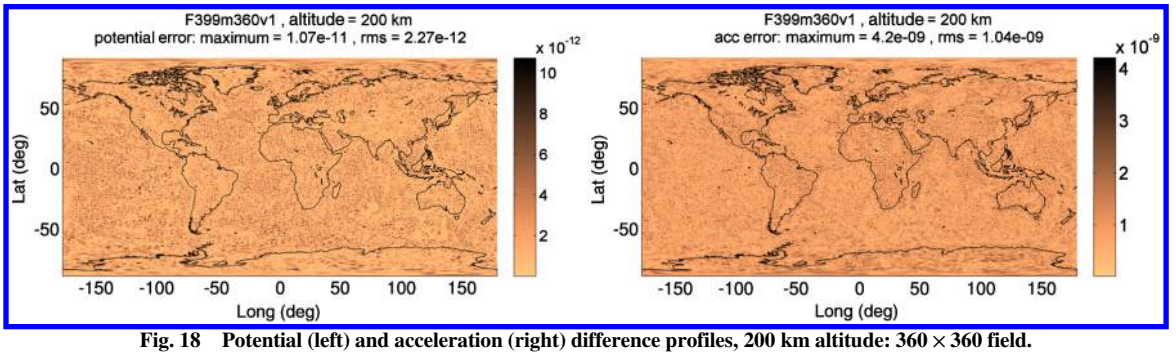

Fig. 18 Potential (left) and acceleration (right) difference profiles, 200 km altitude: 360×360 field.

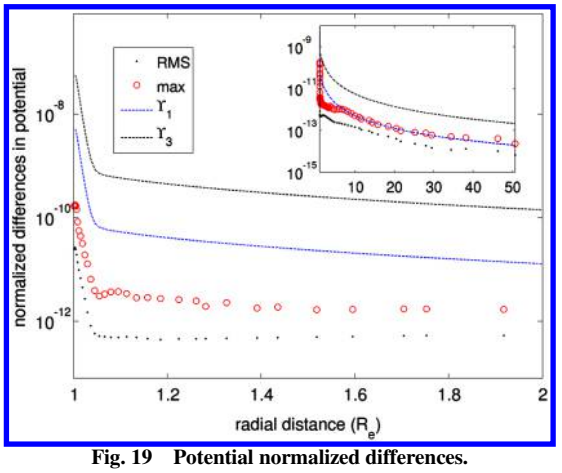

Fig. 19 Potential normalized differences.

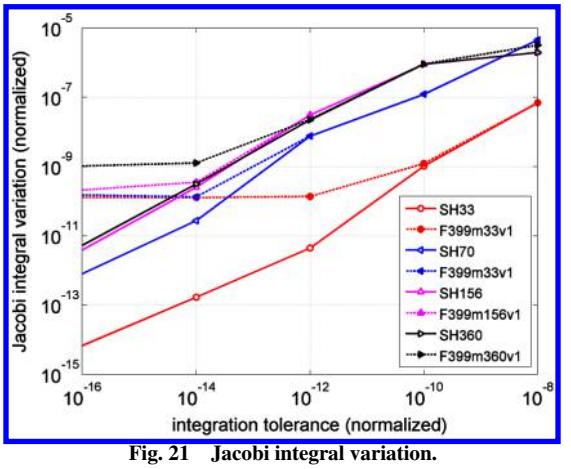

Fig. 21 Jacobi integral variation.

Fetch (e.g., load the binary coefficient file), and 2) call Fetch runtime routines to retrieve the potential and any desired derivatives.

The first step initializes Fetch and loads the coefficient file requested by the user to the computer's random access memory (RAM). This step is executed only once during the model initialization phase. This step can take anywhere from less than 1 s to up to ∼20 s, depending on the size of the Fetch model being loaded and hardware of the computer. Once the model is initialized, various calls to Fetch routines can proceed. Inside the core runtime routines, the coefficients are efficiently tracked and the correct eight neighboring node polynomials are identified and evaluated, followed by a final weighted evaluation of the composite Fetch runtime function.

A. Coefficient Lookup

All routines take the spacecraft geocentric Cartesian position vector as part of their input, and subsequently identify the 3-D cell housing that position vector. Normally, such a task would require a lookup table; however, the uniform surface grid spacing along with the precomputed radial shell distances can be exploited to identify the correct cell using only a double to integer conversion function. The coefficients are stored in a manner such that only one node position lookup is required in order to gather all of the neighboring eight node positions.

B. Fetch Runtime Routines

Table 3 lists the runtime routines that have been implemented for the current release. The main "get_Fetch" routine takes the geocentric Cartesian vector, body GM, body radius, J_2, and required derivative order as inputs and gives the interpolated potential plus derivatives as output. The accelerations are computed by taking the gradient of the interpolated geopotential function with respect to spherical coordinates, and then performing the necessary transformations to the Cartesian coordinates [32]. Similar coordinate transformations and chain rules are required for any necessary higher-order derivatives. In this implementation, runtime routines are provided to include up to third-order derivatives (with respect to Cartesian coordinates) of the potential. Again, it is emphasized that

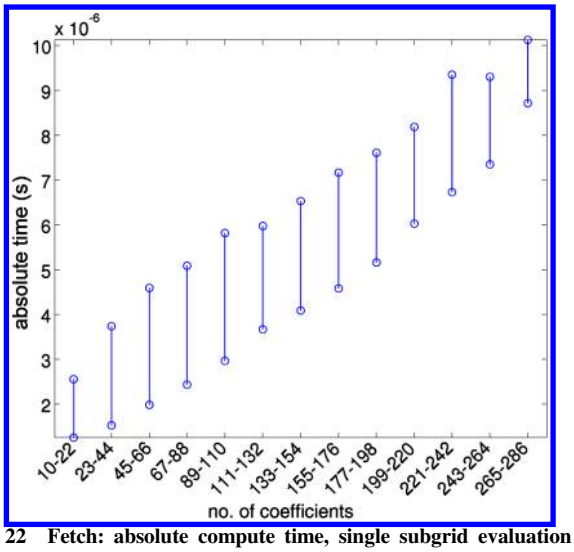

Fig. 22 Fetch: absolute compute time, single subgrid evaluation of potential and acceleration.

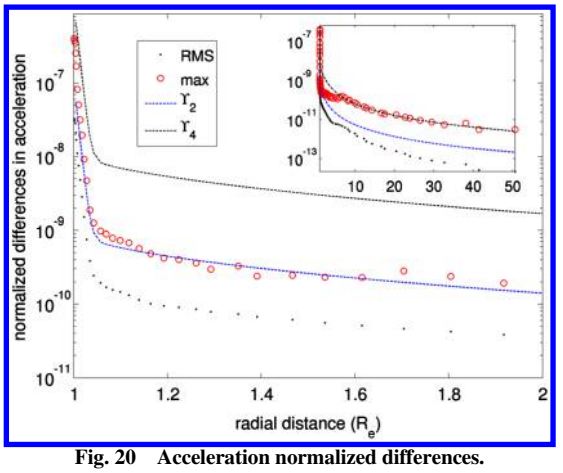

Fig. 20 Acceleration normalized differences.

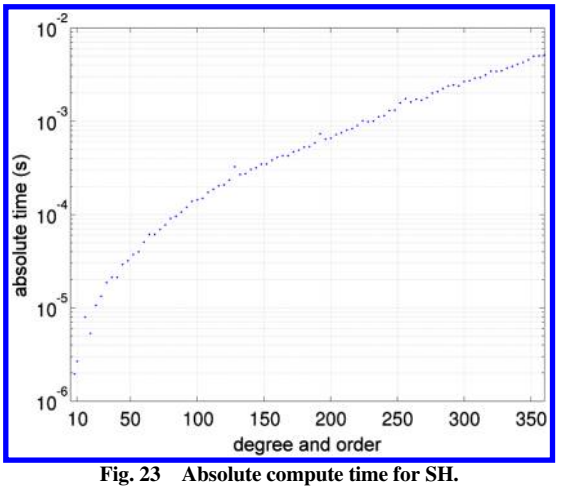

Fig. 23 Absolute compute time for SH.

Table 6 Path classification

Path no.	Type	Perigee altitude, km	Apogee altitude, km	Inclination, deg	No. revolutions	Function calls
1	Low altitude, circular, inclined	200	200	65	32.5	65,000
2	Low altitude, circular, near polar	500	500	85	30.5	46,000
3	Medium altitude, circular, near polar	1,350	1,350	85	25.6	39,500
4	High altitude, circular, near polar	4,050	4,050	85	16.4	24,700
5	Low perigee, highly eccentric, inclined	150	$5R_e$	65	6.5	12,600

the singularity issue with the spherical coordinates conversions is handled through the use of the two overlapping grids.

V. Runtime Performance

In this section, the performances of the Fetch models are evaluated against an efficient CPU implementation of the Pines SH model [1,3]. It is noted that both the Pines SH algorithm and the Fetch model are singularity free. Even though the Fetch model extends beyond the moon, the performance comparisons are limited to an altitude of approximately $5R_e$. All four of the Fetch models from Table 2 are considered for performance comparison. Table 4 gives the specifications of the runtime test hardware and compiler.

A. Comparison with Fitting SH Function

Comparisons are performed by dividing the evaluation region into five altitude bands, listed in Table 5. For each band, sample direct calls to the Pines SH and Fetch models are made and differences between the Fetch and SH models are evaluated. The number of sample points for each band varies between 6500 and 50,000, and they are selected heuristically depending on the degree and order of the field.

Figures 15 and 16 show the rms and maximum of the differences in potential and acceleration corresponding to each of the bands and for the various fidelity models. As expected, the rms of the differences in potential and acceleration is always found to be less than the target residual from the fitting process and the maximum of the differences is generally one order of magnitude greater. The observed differences in lower-fidelity fields are much smaller due to the lower target residuals associated with the lower field fits.

Figure 17 and 18 show the differences (in normalized units) in potential and acceleration at a 200 km altitude for a Fetch model interpolating a 70×70 SH model and a 360×360, respectively. As expected, the maximum differences in the potential and acceleration are found to be less than their computed runtime residual tolerances. Looking at Fig. 10, the estimated rms accuracies of the GGM03C 360×360 model are approximately 3×10^{-8} at the surface (and get mapped to a slightly lower value when evaluated at 200 km); therefore, the Fetch model, with a maximum geopotential difference of 10^{-11}, is conservatively (noting the rms of the difference is even lower) three orders of magnitude below the noise level of the SH function being fit. The cubed-sphere model is also three orders of magnitude below the noise of the SH model [28]. Therefore, both the cubed-sphere and Fetch models can be considered sufficiently accurate (if not overly accurate) with respect to the true geopotential.

Figures 19 and 20 shows the comparison to SH statistics for a complete sweep of the global domain for the F399m156v1 model. Each point corresponds to a shell and contains the maximum or rms of the differences in potential and acceleration over all the cells, with 27 random evaluations within each cell. Near the surface, Υ_2 generally is the limiting one of the four residual criteria $\boldsymbol{\Upsilon}$; therefore, the rms of the differences in potential is significantly smaller in these regions.

Note that the general polynomials used for the interpolation basis are agnostic to the function being fit. In this case, the fitting function is a conservative gravity field and enjoys several mathematical features not present in the polynomial fitting functions. The following are examples:

1) The spherical harmonics function is a solution to Laplace's equation.

2) The equation of motion for a spacecraft in a conservative gravity field admits the well-known Jacobi integral.

In future works, a custom basis function could be implemented that preserves such features; however, in its current form, the Fetch model will only preserve such constants according the precision of the fit. To demonstrate the impact, Fig. 21 shows the variation of the Jacobi integral as a function of integration tolerance with both SH and Fetch, respectively. A 300-km-altitude reference orbit with an inclination of 89 deg is integrated for one day (the whole Earth is covered once in body-fixed coordinates). It can be seen that both methods preserve the Jacobi constant to the same precision for lower integration tolerances. For tighter tolerances, Fetch reaches a limit on number of digits preserved, depending on the order of the fit. In this case, at the tightest tolerance, the Fetch method preserves the Jacobi integral to between 9 and 10 digits in all four models, wheeras the SH method varies between 11 and 14 digits for its corresponding models.

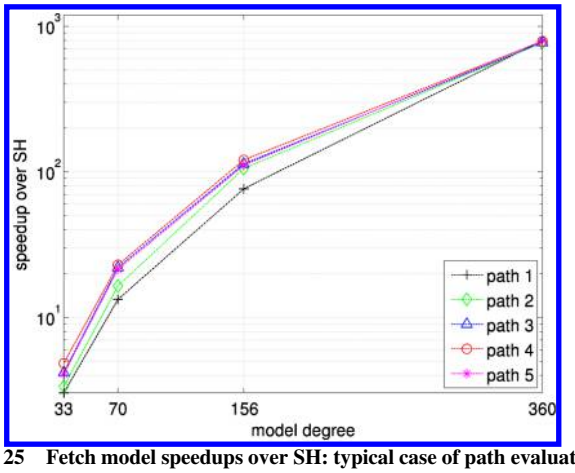

Fig. 25 Fetch model speedups over SH: typical case of path evaluation such that the sequential calls to Fetch are from the same or neighboring cells.

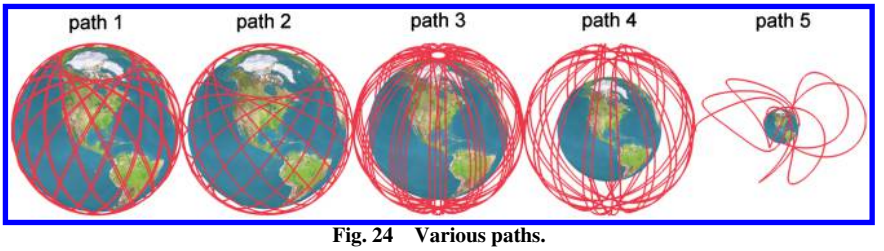

Fig. 24 Various paths.

Despite this drawback for the Fetch model, it is emphasized that, in practice, such errors in the Jacobi constant are smoothed over when integrating a full trajectory. The residuals in position and velocity are only 0.56 m and 0.64 mm/s, respectively, when comparing the Fetch and SH propagations for the 360×360 resolution case and the tightest tolerance from Fig. 21.

B. Speed Comparison

As Fetch trades memory for runtime performance, the speed at which memory (RAM) can be accessed has a major impact on the model's runtime performance. Randomized reads from memory are slower than sequential reads or reads from adjacent memory banks. Hence, the runtime performance of Fetch varies significantly, depending on how the coefficients are being accessed.

Fetch's runtime performance is independent of the SH degree and order. Figure 22 shows the minimum and maximum absolute evaluation times vs number of coefficients. To generate the minimum evaluation time data, multiple points per cell are evaluated to mimic the near-sequential access behavior. The maximum evaluation time data are generated by evaluating, in sequence, nonadjacent cells with the same number of coefficients. Given the computational nature of SH, its evaluation time increases quadratically as the degree and order of the SH field increases (see Fig. 23). From the timing data, we can compute the expected runtime speedup corresponding for a given number of coefficients. Higher-order Fetch models demonstrate nearly three orders of magnitude in speedup. The minimum break-even resolution for runtime speeds is approximately the degree and order of 13. It is noted that the Fetch runtime performance is cut in half in the regions where the primary and rotated grids overlap.

Fortunately, in most foreseen applications, the gravity field will be queried in a near-sequential manner, such as a trajectory following a path in the 3-D domain. Hence, we can expect absolute evaluation times close to the minimum evaluation time values in Fig. 22. Typical low-altitude applications (those more likely to require high-fidelity gravity) will encounter cells with the greatest number of polynomial coefficients. The higher-order derivative routines are also tested, and it is found that including the Jacobian or the Jacobian and the Hessian of the acceleration costs an additional 20 and 30% compute times, respectively, when compared to the potential and acceleration-only case. On the contrary, in a simple SH experiment, it is noted the additional Jacobian or Jacobian and Hessian calculations lead to 50 and 200% compute time increases, respectively. Therefore, the speedups of the Fetch model are further amplified in the case of higher-order derivatives.

C. Path Comparison

To gauge the performance of Fetch in a realistic application, we query successive calls according to a trajectory or path in the global domain. Because different paths access different cells with a different number of coefficients, five representative spacecraft paths (see Table 6 and Fig. 24) are considered. Emphasis is put on selecting low- to medium-altitude paths where higher-order gravity perturbations are important. The timings of the Fetch and SH calls are evaluated for each of the four Fetch models and each of the five spacecraft trajectories. The SH field is not truncated as a function of radial distance for SH path computations, and this affects the speedup of the highly eccentric path computation (like path 5). It is assumed that the spacecraft trajectories are precomputed and the timing results only include the Fetch and SH function calls.

Figure 25 summarizes the speedup results. The effect of truncating the SH model [using Eq. (13)] during Fetch coefficient generation results in decreasing relative speedup (among various paths) with an increasing model degree. For medium- to high-order Fetch models, we achieve multiple orders of magnitude in speedup. Specifically for the complete 360×360 SH field, the final speedup varies from 770 to 800 times. Observe that the speedup scales greater than quadratic with the SH degree and order. This is explained from a combination of two effects. First, higher-order Fetch models require a fewer number of coefficients per cell (see Table 2). Second, the SH computation time scales quadratically with field size.

VI. Conclusions

The main objectives of the Fetch model is to provide a fast and accurate way for calculating high-order gravity fields while preserving the mathematical attractiveness (smoothness, nonsingularity) associated with the spherical harmonics formulation. The Fetch method trades an affordable memory investment for unprecedented speed gains. It uses the Junkins weighting function technique [20] to achieve continuity over the whole solution domain and allows for localized resolution via adaptive polynomial fitting. The geopotential is the only interpolated function with the acceleration and any higher-order derivatives calculated by explicitly differentiating the interpolant. The singularity present at the poles when conventionally dealing in spherical coordinates is tackled by implementing a two-level grid structure with an overlapping latitude band. Therefore, the four priorities (continuous/smooth, adaptive, nonsingular, and accurate) laid out in the Introduction (Sec. I) are achieved. The memory footprint is fixed at the cost of storing the potential only, and it scales almost linearly with the SH degree and order. In contrast, the speedup scales faster than quadratic, making a compelling case for applications requiring high-order SH.

Adaptivity for the Fetch model is achieved through both the radial direction and through the optimization of choosing the best out of 228 candidate interpolants for each node. Precomputed analytic solutions to the least-squares normal equations afford the extreme flexibility of evaluating all candidate interpolants for all nodes (up to eight million in this study). Furthermore, target residual tolerances for each node are calibrated based on the published accuracy of the GMM03C gravity model. These residual levels are used to guide the fitting process so as to minimize the number of coefficients needed for each node while maintaining a uniform global residual profile that is consistent with the physics of the problem and the accuracy of the SH function being fit. A master-worker-based parallel version of the algorithm is implemented (in a message-passing interface system) to efficiently generate the coefficients for the high-order SH fields. The parallel coefficient generation algorithm enables the fitting of high-degree SH fields. The highest resolution currently investigated (and native size of the GGM03C model) is 360×360, where the Fetch model achieves up to 800 times the speedup. The residual profile for the potential and acceleration are uniform according to the order of the tolerance specified. In its current form, the Fetch model does not satisfy the Laplacian equation. Future works will consider either constraining the least-squares problem or choosing basis functions that naturally satisfy the Laplacian equation.

The ease of implementation combined with speed and favorable continuity properties make the Fetch model attractive for high-fidelity orbit determination or trajectory optimization tasks The runtime codes and model coefficients are available online.[§]

Acknowledgments

This material presented in this paper is based, in part, upon work supported by U.S. Air Force Office of Scientific Research under contract no. FA9550-10-C-0061. The authors thank Kent Miller, Haijun Shen, Jack Van Wieren, Srinivas Bettadpur, and John Junkins for their interest, support, and fruitful discussions. The authors also acknowledge the Texas Advanced Computing Center at the University of Texas at Austin for providing (high-performance computing) resources that have contributed to the research results reported within this paper.

References

[1] Pines, S., "Uniform Representation of the Gravitational Potential and Its Derivatives," *AIAA Journal*, Vol. 11, No. 11, 1973, pp. 1508–1511. doi:10.2514/3.50619

[2] Casotto, S., and Fantino, E., "Evaluation of Methods for Spherical Harmonic Synthesis of the Gravitational Potential and Its Gradients," *Advances in Space Research*, Vol. 40, No. 1, 2007, pp. 69–75. doi:10.1016/j.asr.2007.01.021

[§]Data available online at http://russell.ae.utexas.edu/index_files/fetch.htm [retrieved 2015].

[3] Lundberg, J. B., and Schutz, B., "Recursion Formulas of Legendre Functions for Use with Nonsingular Geopotential Models," *Journal of Guidance, Control, and Dynamics*, Vol. 11, No. 1, 1988, pp. 32–38. doi:10.2514/3.20266

[4] Tapley, B., Ries, J., Bettadpur, S., Chambers, D., Cheng, M., Condi, F., and Poole, S., "The GGM03 Mean Earth Gravity Model from GRACE," *EOS Transactions of the AGU*, Vol. 88, No. 52, 2007, Fall Meeting Suppl., Abstract G42A-03.

[5] Hoffman, T., "GRAIL: Gravity Mapping the Moon," *2009 IEEE Aerospace Conference*, IEEE Publ., Piscataway, NJ, March 2009, pp. 1–8.

[6] Hoots, F. R., Schumacher, P. W., and Glover, R. A., "History of Analytical Orbit Modeling in the U.S. Space Surveillance System," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 2, 2004, pp. 174–185. doi:10.2514/1.9161

[7] Koch, K. R., "Simple Layer Model of the Geopotential in Satellite Geodesy," *The Use of Artificial Satellites for Geodesy*, Vol. 15, Geophysics Monograph Series, Wiley, New York, 1972, pp. 107–109. doi:10.1029/GM015

[8] Wong, L., Buechler, G., Downs, W., Sjogren, W., Muller, P., and Gottlieb, P., "A Surface Layer Representation of the Lunar Gravitational Field," *Journal of Geophysical Research*, Vol. 76, No. 26, 1971, pp. 6220–6236. doi:10.1029/JB076i026p06220

[9] Koch, K. R., and Witte, B. U., "Earth's Gravity Field Represented by a Simple Layer Potential from Doppler Tracking of Satellites," *Journal of Geophysical Research*, Vol. 76, No. 35, 1971, pp. 8471–8479. doi:10.1029/JB076i035p08471

[10] Koch, K. R., and Morrison, F., "A Simple Layer Model of the Geopotential from a Combination of Satellite and Gravity Data," *Journal of Geophysical Research*, Vol. 75, No. 8, 1970, pp. 1483–1492. doi:10.1029/JB075i008p01483

[11] Morrison, F., "Algorithms for Computing the Geopotential Using a Simple Density Layer," *Journal of Geophysical Research*, Vol. 81, No. 26, 1976, pp. 4933–4936. doi:10.1029/JB081i026p04933

[12] Woodburn, J., Szebehely, V., and Zare, K., "An Alternative Representation of the Geopotential," *AAS/AIAA Spaceflight Mechanics Meeting*, Vol. 89, Advances in the Astronautical Sciences, Spaceflight Mechanics, Univelt, Inc., San Diego, CA, 1995, pp. 1311–1330; also AAS Paper 95-191.

[13] Rayman, M. D., Fraschetti, T. C., Raymond, C. A., and Russell, C. T., "Dawn: A Mission in Development for Exploration of Main Belt Asteroids Vesta and Ceres," *Acta Astronautica*, Vol. 58, No. 11, 2006, pp. 605–616. doi:10.1016/j.actaastro.2006.01.014

[14] Werner, R., and Scheeres, D., "Exterior Gravitation of a Polyhedron Derived and Compared with Harmonic and Mascon Gravitation Representations of Asteroid 4769 Castalia," *Journal of Geophysical Research*, Vol. 81, No. 26, 1976, pp. 4933–4936. doi:10.1029/JB081i026p04933

[15] Park, R. S., and Scheeres, D. J., "Nonlinear Semi-Analytic Methods for Trajectory Estimation," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 6, 2007, pp. 1668–1676. doi:10.2514/1.29106

[16] Park, R. S., Werner, R. A., and Bhaskaran, S., "Estimating Small-Body Gravity Field from Shape Model and Navigation Data," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 1, 2010, pp. 212–221. doi:10.2514/1.41585

[17] Russell, R. P., and Arora, N., "Global Point Mascon Models for Simple, Accurate, and Parallel Geopotential Computation," *AAS/AIAA Space Flight Mechanics Meeting*, American Astronomical Soc. Paper 2011-158, Washington, D.C., 2011.

[18] Takahashi, Y., Scheeres, D. J., and Werner, R. A., "Surface Gravity Fields for Asteroids and Comets," *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 2, 2013, pp. 362–374. doi:10.2514/1.59144

[19] Herrera-Sucarrat, E., Palmer, P. L., and Roberts, R. M., "Modeling the Gravitational Potential of a Nonspherical Asteroid," *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 3, 2013, pp. 790–798. doi:10.2514/1.58140

[20] Junkins, J. L., "Investigation of Finite-Element Representations of the Geopotential," *AIAA Journal*, Vol. 14, No. 6, 1976, pp. 803–808. doi:10.2514/3.61420

[21] Junkins, J. L., Miller, G. W., and Jancaitis, J. R., "A Weighting Function Approach to Modeling of Irregular Surfaces," *Journal of Geophysical Research*, Vol. 78, No. 11, 1973, pp. 1794–1803. doi:10.1029/JB078i011p01794

[22] Junkins, J. L., and Engels, R. C., "Local Representation of the Geopotential by Weighted Orthonormal Polynomials," *Journal of Guidance, Control, and Dynamics*, Vol. 3, No. 1, 1980, pp. 55–61. doi:10.2514/3.55947

[23] Beylkin, G., and Cramer, R., "Toward Multiresolution Estimation and Efficient Representation of Gravitational Fields," *Celestial Mechanics and Dynamical Astronomy*, Vol. 84, No. 1, 2002, pp. 87–104. doi:10.1023/A:1019941111529

[24] Lekien, F., and Marsden, J., "Tricubic Interpolation in Three Dimensions," *International Journal for Numerical Methods in Engineering*, Vol. 63, No. 3, 2005, pp. 455–471. doi:10.1002/(ISSN)1097-0207

[25] Colombi, A., Hirani, A. H., 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

[26] Hujsak, R. S., "Gravity Acceleration Approximation Functions," Vol. 93, American Astronomical Soc. Paper 1996-123, Washington, D.C., 1996, pp. 335–349.

[27] Oltrogge, D. L., "AstroHD: Astrodynamics Modeling with a Distinctly Digital Flavor," *AAS/AIAA Astrodynamics Specialist Conference*, AIAA Paper 2008-7065, Aug. 2008.

[28] Jones, B. A., Born, G. H., and Beylkin, G., "Comparisons of the Cubed-Sphere Gravity Model with the Spherical Harmonics," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 2, 2010, pp. 415–425. doi:10.2514/1.45336

[29] Arora, N., and Russell, R. P., "Fast, Efficient and Adaptive Interpolation of the Geopotential," *AAS/AIAA Astrodynamics Specialist Conference*, American Astronomical Soc. Paper 2011-501, Washington, D.C., 2011.

[30] Stewart, G. W., "Lecture 20," *Afternotes on Numerical Analysis*, Soc. of Industrial and Applied Mathematics, Philadelphia, 1996, pp. 147–153, Chap. 20. doi:10.1137/1.9781611971491.ch20

[31] Acton, C., "Ancillary Data Services of NASA's Navigation and Ancillary Information Facility," *Planetary and Space Science*, Vol. 44, No. 1, 1996, pp. 65–70. doi:10.1016/0032-0633(95)00107-7

[32] Spier, G., "Design and Implementation of Models for the Double Precision Trajectory Program (DPTRAJ)," Jet Propulsion Lab., California Inst. of Technology TM-33-451, Pasadena, CA, 1971.