Geosci. Model Dev., 10, 425–431, 2017
www.geosci-model-dev.net/10/425/2017/
doi:10.5194/gmd-10-425-2017

Geoscientific Model Development
Open Access
EGU

Conservative interpolation between general spherical meshes

Evaggelos Kritsikis[1], **Matthias Aechtner**[2], **Yann Meurdesoif**[3], **and Thomas Dubos**[2]

[1]Laboratoire d'analyse, géométrie et applications, université Paris 13, 93430 Villetaneuse, France
[2]Laboratoire de météorologie dynamique, École polytechnique – IPSL, 91128 Palaiseau, France
[3]Laboratoire des sciences du climat et de l'environnement, CEA – IPSL, 91191 Gif-sur-Yvette, France

Correspondence to: Evaggelos Kritsikis (krits@math.univ-paris13.fr)

Received: 4 March 2015 – Published in Geosci. Model Dev. Discuss.: 30 June 2015
Revised: 20 April 2016 – Accepted: 13 July 2016 – Published: 30 January 2017

Abstract. An efficient, local, explicit, second-order, conservative interpolation algorithm between spherical meshes is presented. The cells composing the source and target meshes may be either spherical polygons or latitude–longitude quadrilaterals. Second-order accuracy is obtained by piece-wise linear finite-volume reconstruction over the source mesh. Global conservation is achieved through the introduction of a "supermesh", whose cells are all possible intersections of source and target cells. Areas and intersections are computed exactly to yield a geometrically exact method. The main efficiency bottleneck caused by the construction of the supermesh is overcome by adopting tree-based data structures and algorithms, from which the mesh connectivity can also be deduced efficiently.

The theoretical second-order accuracy is verified using a smooth test function and pairs of meshes commonly used for atmospheric modelling. Experiments confirm that the most expensive operations, especially the supermesh construction, have $O(N \log N)$ computational cost. The method presented is meant to be incorporated in pre- or post-processing atmospheric modelling pipelines, or directly into models for flexible input/output. It could also serve as a basis for conservative coupling between model components, e.g., atmosphere and ocean.

1 Introduction

Despite the simplicity and regularity of a spherical surface, there is no single ideal way to mesh it. Consequently, numerical methods formulated on the sphere, used for instance in weather forecasting and climate modelling, use a variety of meshes. For a long time, spectral and finite-difference schemes have been using latitude–longitude meshes. However, most recently developed methods use more flexible meshes like triangulations of the sphere and their Voronoi dual or quadrangular meshes like the cubed sphere. Such meshes avoid the polar singularity inherent to the latitude–longitude system (Williamson, 2007).

Different physical components like atmosphere, land, ice and ocean typically use distinct meshes. As they are coupled together, interpolation between the various meshes is required. Furthermore, the native model mesh may not be the most practical to perform post-processing and analysis of the simulations, and interpolating to a more convenient mesh can be desirable. Finally, interpolation is a crucial building block of dynamic mesh adaptation, which enables a simulation to dynamically focus resolution where it is important, potentially saving orders of magnitude in computational costs. Although dynamic adaptivity is not a current practice in ocean–atmosphere modelling, there is a growing body of research to this end, and dynamic adaptivity may mature in the future. Meanwhile, statically refined meshes are increasingly used, and there is a need to interpolate from/to such meshes.

In applications like climate modelling, it is often vital that some physical quantities be conserved, such as density, volume fractions or tracer concentrations. When interpolating fluxes between physical components coupled together, similar conservation constraints should be enforced. Failing to enforce these conservation properties may create spurious sources and sinks which, however small, may accumulate over time and overwhelm the physical trends. Therefore, even if one uses a conservative discretization method for the relevant equations, there is a need to ensure conservation in the interpolation step.

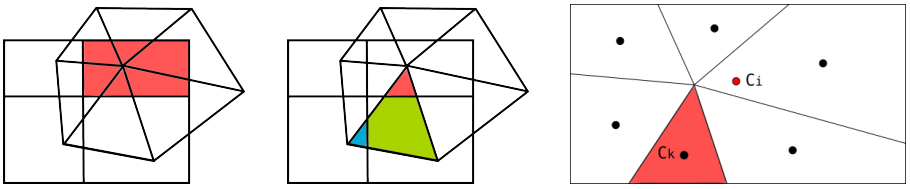

Figure 1. Common refinement of the source and target meshes. Here, the source mesh (S_i) is quadrangular and the target mesh (T_j) is triangular. The integral over quadrangle S_i (red, left panel) must be split among supermesh cells U_k as per Eq. (2); supermesh cell areas A_k (colored, center panel) serve as interpolation weights. To this end, linear reconstruction is done on each quadrangle. The quadrature points are the supermesh centroids C_k (right panel).

This paper describes a second-order conservative interpolation algorithm on the sphere. Our method improves on previously published work as follows:

- It is geometrically exact as defined and discussed in Ullrich et al. (2009), and unlike the method used by Jones (1999).
- It is local and explicit, unlike optimization-based approaches (Jiao and Heath, 2004; Farrell et al., 2009) which may require an iterative solver to approximate the function on the target mesh. We do not seek a "best" approximation here but only suppose a quadrature scheme in order to test the conservation requirement. Therefore, a small number of interpolation weights can be precomputed and parallelism is facilitated.
- It is not tied to a narrow class of meshes (e.g., Ullrich et al., 2009, which handles only cubed-sphere and lat–long meshes): our method handles lat–long meshes and arbitrary polygonal meshes, including the cubed-sphere, general triangulations and their Voronoi duals, which encompass the vast majority of currently used meshes.
- It does not assume the connectivity of the mesh is available, as do Alauzet and Mehrenberger (2009) and Farrell and Maddison (2011), but reconstructs it instead in quasilinear time.

Our method relies on constructing a common refinement of the source and target meshes, called a supermesh in Farrell et al. (2009). Assuming that the supermesh is known, formulae for second-order conservative interpolation are derived in Sect. 2. Algorithms used to construct the supermesh are described in Sect. 3. Numerical experiments are conducted in Sect. 4 to verify the accuracy of the method when used with various pairs of spherical meshes, as well as the theoretical algorithmic complexity. A summary is given in Sect. 5.

2 Second-order conservative interpolation

The source and target meshes are sets of spherical cells S_i and T_j, each cell being either a spherical polygon or a lat–long quadrilateral. The intersection $S_i \cap S_j$ (resp. $T_i \cap T_j$) for $i \neq j$ is either void, a shared vertex or a shared edge. The latter case defines neighboring cells. Both meshes are assumed to cover the whole sphere, i.e., $\bigcup S_i = \bigcup T_j$.

Scalar functions are assumed to be described via their integrals over mesh cells. Indeed, in most GCMs, many (if not all) fields are treated in a finite-volume manner. The problem we wish to solve is, given the integrals f_i of a smooth function f on the source mesh, to obtain accurate estimates f'_j of the integrals on the target mesh, so that the total integral is preserved:

$$\sum_i f_i = \sum_j f'_j. \tag{1}$$

Second-order accuracy will result from linear reconstructions on each S_i, assuming f has a bounded second derivative, in a similar manner to Alauzet and Mehrenberger (2009), although the spherical geometry introduces new conditions (see below). To achieve conservation (Eq. 1), one introduces the supermesh $U_k = (S_i \cap T_j)_{i,j}$. In the following, the i, j and k subscripts, respectively, denote the source mesh, target mesh and supermesh. The supermesh is a common refinement of the source and target meshes such that any cell of those is a union of cells of the supermesh. The problem comes down to finding approximations:

$$f''_k \approx \int_{U_k} f \quad \text{s. t.} \quad \sum_{U_k \subset S_i} f''_k = f_i. \tag{2}$$

We want the approximation to be exact for a constant function. For the cell areas A_i and A_k, this property implies

$$A_i = \sum_{U_k \subset S_i} A_k. \tag{3}$$

To satisfy Eq. (3), all spherical areas are computed exactly (see Sect. 3.4).

In the general case, a piece-wise linear reconstruction $f^{\text{app}} \in PC^1(S)$ of f over the source mesh is built and integrated by approximate quadrature over U_k, yielding f''_k. We define the reconstruction as

$$f_i^{\text{app}}(x) = \overline{f}_i + g_i \times (x - C_i) \quad \text{for any } x \in S_i, \tag{4}$$

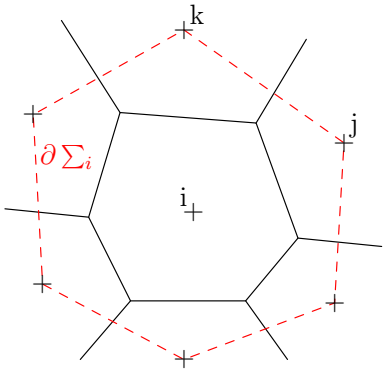

Figure 2. Gradient computation: Stokes formula is applied on the boundary $\partial\Sigma_i$ of the polygon surrounding cell i. The vertices of Σ_i are the barycenters of nearest-neighbor cells j, k, etc.

where $\overline{f}_i = f_i/A_i$ is the mean value of f over S_i, g_i is an approximation of the gradient of f on S_i and C_i is the centroid of S_i. The quadrature is defined as $f''_k = A_k\ f^{\text{app}}(C_k)$; in other words, the supermesh cell areas A_k are the interpolation weights (Fig. 1). It follows that

$$\forall i, \sum_{U_k \subset S_i} f''_k = \sum_{U_k \subset S_i} A_k \overline{f}_i + \sum_{U_k \subset S_i} A_k g_i \times (C_k - C_i) \tag{5}$$

$$= f_i + g_i \times \sum_{U_k \subset S_i} A_k C_k - A_i g_i \times C_i, \tag{6}$$

in view of Eq. (3), which gives two necessary orthogonality conditions for Eq. (2) to hold:

- $\forall i, g_i \times C_i = 0$,
- $\forall i, g_i \times \sum_{U_k \subset S_i} A_k C_k = 0$.

By computing first the barycenters C_k of the supermesh cells U_k, then obtaining the barycenters of the source cells from them as $C_i = N(\sum_{U_k \subset S_i} A_k C_k)$, where $N(C) = C/\sqrt{C \times C}$, the two above conditions become equivalent. To satisfy them, a first-order estimate $\widetilde{g}_i$ of the gradient is orthogonalized with respect to C_i, yielding g_i. Since the orthogonality condition is satisfied by the exact gradient, this orthogonalization entails no loss of accuracy.

The $\widetilde{g}_i$ are computed by the Gauss formula on a neighborhood of S_i that is the polygon Σ_i joining the centroids of neighboring elements (Tomita et al., 2001). Indeed as

$$\int_{V_i} \nabla f = \int_{\partial\Sigma_i} \left(f - \overline{f}_i\right) n \mathrm{d}s, \tag{7}$$

with $\partial\Sigma_i$ the boundary of Σ_i and n the outward normal to Σ_i, we set

$$\widetilde{g}_i = \frac{1}{A\left(\Sigma_k\right)} \sum_{\substack{S_i \cap S_j \cap S_k \neq \varnothing \\ i,j,k \text{ distinct}}} \left(\frac{\overline{f}_j + \overline{f}_k}{2} - \overline{f}_i\right) C_j \times C_k, \tag{8}$$

where each pair j, k of neighbors appears only once, so that the triangle $C_i C_j C_k$ is counterclockwise (Fig. 2). In Eq. (8), subtracting $\overline{f}_i$ guarantees that a constant field has a zero gradient.

3 Spherical supermesh

3.1 Intersection between a pair of cells

We describe here how, given two cells C and C', their intersection U is obtained. The algorithm starts from a vertex of C inside C' and looks along the edge of C for the next interior vertex or intersection with an edge of C' (all intersections on that edge are computed exactly as circle–circle intersections in 3-D Cartesian space). In the latter case, the intersecting edge of C' is then followed, with a possible direction reversal if a step was done towards the exterior of C (as checked by an orientation test with regard to the normal of the previously followed edge). When the initial vertex is reached again, a connected component of U has been determined (see Fig. 3).

Notice that U is allowed to have several connected components, in which case as many supermesh cells are created. The usual degenerate cases of zero area can of course happen. On the other hand, the problem of non-matching surfaces discussed in Jiao and Heath (2004) does not occur here, as all intersections are computed between bits of small and great circles on the sphere.

3.2 Fast search of potential intersectors

Constructing the supermesh requires in principle to compute the intersection between all S_i and T_j. Assuming both meshes have $O(N)$ cells, this brute-force approach has a quadratic algorithmic complexity $O(N^2)$. In fact, most intersections are empty. Cells of the source and destination meshes can be grouped hierarchically in sets with mostly empty mutual intersections. Exploiting this fact, as described below, yields fast search algorithms and is crucial to attain quasilinear algorithmic complexity. Another idea, used by Alauzet and Mehrenberger (2009) or Farrell and Maddison (2011), is to exploit the connectivity of the mesh. However, the connectivity may not be readily available, for instance, when reading data from NetCDF files following the NetCDF-CF convention (http://cfconventions.org/). Actually, our fast search algorithm can be used to reconstruct it.

The algorithm takes as input a mesh and a spherical circle. It yields a list of cells in the mesh that potentially lie partly or totally inside the circle. The algorithm guarantees that all cells of the mesh that actually lie partly or totally in the circle are on the list. Some of those cells may in fact lie outside the circle, although the algorithm is designed to keep their number to a minimum.

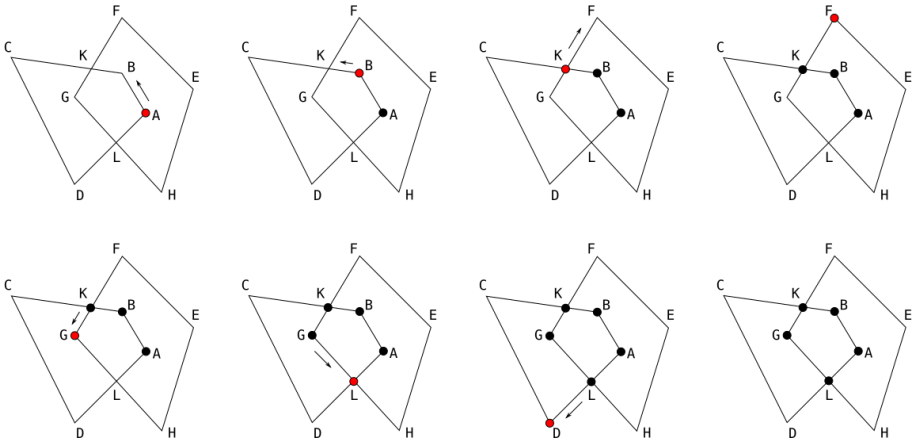

Figure 3. Steps in determining the intersection U of cells ABCD and EFGH. The starting point is a vertex interior to the other cell, here denoted by A. The first examined segment is AB, which contains no intersections (step 1) so it is included in U as is. Along edge BC, intersection K is encountered (step 2), so segment BK is included in U, and we branch to edge KF of the other cell (step 3). However, vertex F is checked to be outside cell ABCD (step 4), so the search direction is reversed towards G (step 5). Another intersection is then found along edge GH, namely L (step 6), and after another direction reversal due to stepping outside again (step 7), the loop is completed by returning to A (step 8).

In order to yield $O(N \log N)$ complexity, a bounding circle is computed for each cell and these circles are inserted sequentially into a similarity-search tree, or SS tree (White and Jain, 1996), which grows progressively starting from an empty tree with a single root node. During this process, each node of the tree has its own bounding circle which encloses the bounding circles of all of its children, and the mesh cells are at the leaves of the tree. To insert one circle, one traverses the tree top-down, choosing at each level the closest child node based on the distance between the centers of the bounding circles. The circle is then inserted at the lowermost level. Before the next circle is inserted, a tree-balancing step is performed. If the parent node of the newly inserted cell has more children than a predefined threshold (here set to $N_{\max} = 10$), it is split in two, hence increasing the child count of its own parent. If the threshold $N_{\max}$ is exceeded again, this node is split, and so on until the root node is reached. If the root node needs to be split, a parent node with two children is created and becomes the new root node, increasing the depth of the tree.

Every insertion is followed by a re-balancing step in order to avoid a large overlap between bounding circles, which would diminish the efficiency of the search algorithm. To this end, after a node (leaf or not) has been inserted, those of its siblings whose distance from the parent exceeds 80 % of its radius are removed from the tree and put into the list of nodes to be inserted later. Such nodes are marked so that they are not removed again from the tree.

To completely specify the tree construction algorithm, we now describe the method used to split a set of $N_{\max} + 1$ children into two sets. First, the child farthest from the center of the parent bounding circle is found. Then, the $N_{\max}/2$ nodes closest to that node are grouped together, while the remaining nodes form another group. Other splitting methods have been proposed and would be easy to implement (Fu et al., 2002).

Once all mesh cells have been inserted and the SS tree is ready, the list of potential intersectors is obtained by traversing the tree top-down, following the branches whose enclosing circle intersects the target circle. The detailed calculation of intersections is performed only with cells in this list.

3.3 Connectivity reconstruction

Although the SS tree is primarily built in order to speed up the construction of the supermesh, it also provides an essentially cost-free means of reconstructing the connectivity of the meshes. Indeed to reconstruct the connectivity of, say, the source mesh, it is sufficient to apply the previous algorithm to the source mesh and a source cell. This connectivity is actually required when computing the gradient $\widetilde{g}_i$. Therefore, as noted above, our method works in circumstances where mesh connectivity is not readily available.

3.4 Supermesh cell area and barycenter

Supermesh cell edges are an arbitrary mix of small and great circle segments. To compute their area, we represent them as a combination of spherical triangles and surfaces enclosed by a small circle segment and a great circle segment with the same endpoints, possibly counted negatively. A similar approach is used for barycenters.

An accurate treatment of small circle segments is crucial for accuracy on reduced latitude–longitude grids (Purser, 1998). Indeed, for such grids the cells close to the poles have strongly curved boundaries and approximations that conflate

a small arc and the great arc with the same endpoints fail to deliver second-order accuracy (not shown).

4 Results

In this section, we verify the accuracy and efficiency of the method, encompassing several types of meshes: latitude–longitude, triangular, polygonal dual and cubed sphere (see Fig. 4). Computations were done on an Intel P8700 processor at 2.53 GHz with 4 GB RAM.

4.1 Meshes

All meshes whose cell edges are an arbitrary mix of great and small spherical arcs are supported. This includes standard and skipped latitude–longitude meshes, cubed-sphere meshes, triangulations and general polygonal meshes. Fig. 4 shows meshes that we specifically use for the tests presented below:

- standard latitude–longitude meshes, where the zonal and meridional resolution are equal at the Equator and the pole is a vertex;
- their skipped variant, where the number of cells along a parallel varies, starting at four around the pole and doubling to keep the zonal cell size less than twice the meridional cell size (Purser, 1998);
- cubed-sphere meshes (Sadourny, 1972);
- triangular–icosahedral meshes and their hexagonal–pentagonal Voronoi duals (Sadourny et al., 1968);
- variable-resolution variants of the latter obtained by applying a Schmidt transform to each vertex (Guo and Drake, 2005).

4.2 Accuracy

Interpolation between various pairs of meshes is applied to the smooth field $2 + xy$. The input data are obtained by evaluating this function at source cell barycenters G_j. The global conservation property (Eq. 1) is satisfied within round-off error (not shown). Interpolation error is evaluated by evaluating the test function at destination cell barycenters and comparing to the interpolated value $\overline{f}_j = f_j / A_j$:

$$\begin{aligned} \varepsilon_p &= \left(\frac{1}{4\pi} \sum A_j \left\| \overline{f}_j - f\left(G_j\right) \right\|^p \right)^{1/p} \\ \varepsilon_\infty &= \max_j \left\| \overline{f}_j - f\left(G_j\right) \right\|. \end{aligned}$$

When using a piece-wise constant reconstruction on the source mesh, interpolation error is expected to be proportional to the local gradient of the test function and to the cell size (largest of source and target mesh sizes). When using a piece-wise linear reconstruction, interpolation error is expected to be proportional to the local second derivatives of the test function and to the squared cell size.

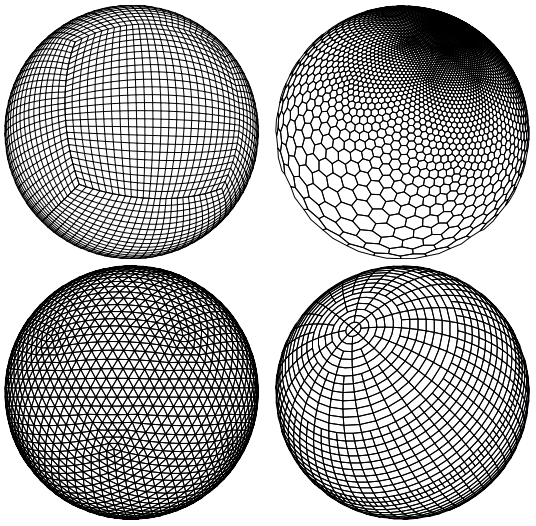

Figure 4. Different meshes are supported and have been tested: latitude–longitude, reduced latitude–longitude (bottom right), triangular (bottom left), cubed sphere (top left) and variable-resolution polygonal (top right).

We first consider remapping between pairs of uniform-resolution meshes of comparable resolution h ranging from 0.01 (a few hundred thousand cells) to 0.1 (a few thousand cells). Figure 5 shows the maximum (L^∞) and root mean square (L^2) interpolation error, as a function of a global characteristic cell size h defined as the average of the local cell sizes, themselves defined as the side length of a square with same area A ($h = \sqrt{(A)}$). Scaling of both errors confirms that the expected first-order (left) and second-order (right) accuracy is achieved.

An application to variable-resolution icosahedral–hexagonal meshes is shown in Fig. 6. The remapping is performed between two such meshes. The source mesh is everywhere about 25 % finer than the destination mesh, while the resolution of each single mesh spans about a decade. As expected, the local error is found to be bounded $O(h^2)$ with h the local mesh size defined here as the square root of the destination cell area.

4.3 Efficiency

Figure 7 shows the computation time of a second-order remapping from a uniform resolution icosahedral–hexagonal mesh to a regular latitude–longitude mesh vs. the number N of elements of the meshes. Total time is decomposed according to the different steps of the algorithms. Since the remapping is a linear operator, it can be expressed in terms of weights forming a sparse matrix. These weights are typically pre-computed for repeated later use. The cost of computing

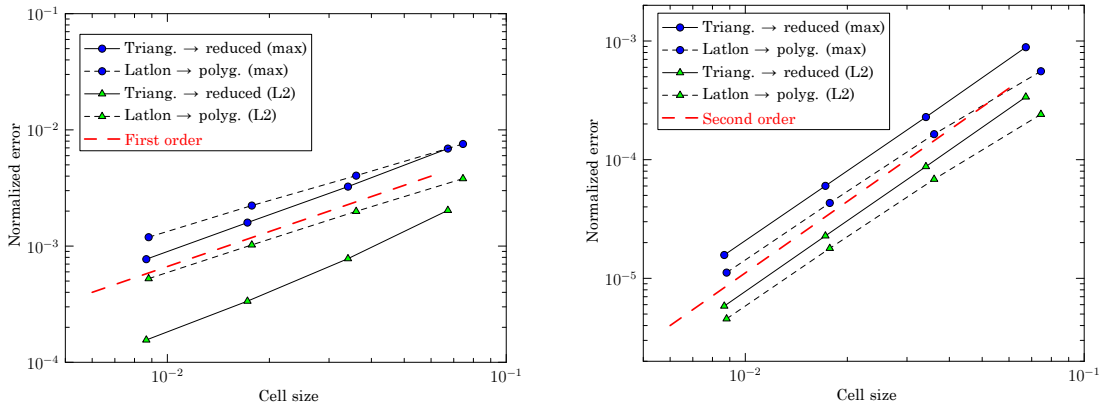

Figure 5. L^2 and L^∞ errors vs. characteristic mesh length h for the remapping of Y_2^2. Left: piece-wise constant reconstruction. Right: piece-wise linear reconstruction.

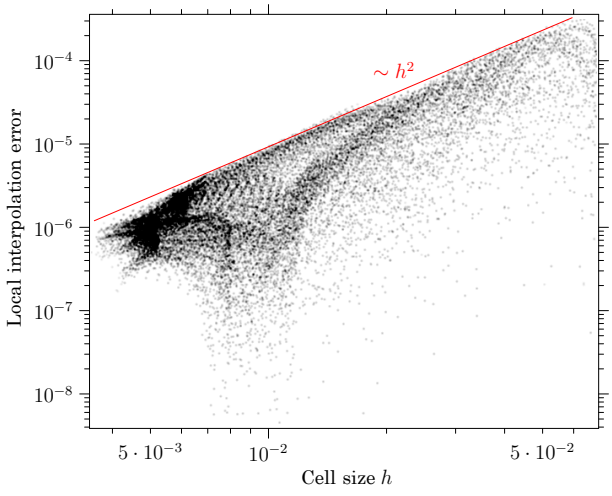

Figure 6. When mapping between non-uniform hexagonal meshes, the local error depends quadratically on the local resolution. Each dot represents a grid cell. The cell size h is computed as the square root of the cell area.

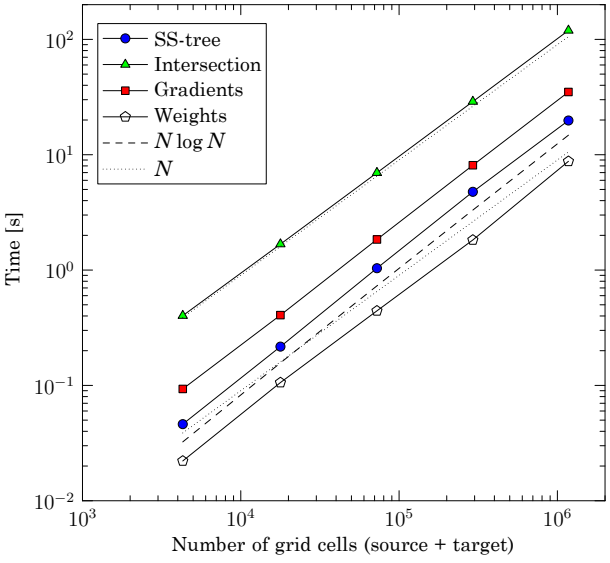

Figure 7. Timing of the various steps of second-order remapping from a uniform resolution icosahedral–hexagonal mesh to a regular latitude–longitude mesh. The SS tree construction shows the expected $O(N \log N)$ complexity.

intersections, gradients (only for second order) and weights is linear in the number of elements. Construction of the SS tree has the theoretical complexity of $O(N \log N)$ (dashed line).

The overall computational cost is dominated by the computation of intersections and therefore is close to linear. Extrapolating those curves suggests that for any imaginable problem size the SS tree will not require more computational resources than the computation of intersections, which has $O(N)$ complexity.

5 Conclusions

A local, explicit, second-order, conservative interpolation algorithm has been devised. The theoretical second-order accuracy has been verified using a smooth test function and pairs of meshes covering most meshes commonly used for atmospheric modelling. The main efficiency bottleneck caused by the construction of the supermesh has been overcome by adopting tree-based data structures and algorithms, from which the mesh connectivity can also be deduced efficiently. Experiments confirm a $O(N \log N)$ computational cost of the most expensive operations, especially the supermesh construction.

Cartesian curvilinear meshes are not covered by this work. Covering such meshes commonly used for ocean modelling requires essentially adapting the detailed computation of intersections. Higher-order interpolations, or vector interpola-

tions can also easily be incorporated. This is left for future work.

Although the present sequential method is fast enough to be included in pre- or post-processing pipelines, further efficiency gains can be obtained by parallelizing it. The least parallel part of the algorithm is the SS tree construction. Work is under way to parallelize this step, using tree approaches again to distribute and balance the workload, and will hopefully be presented separately.

Acknowledgements. E. Kritsikis and M. Aechtner acknowledge support by the ICOMEX project.

Edited by: S. Valcke
Reviewed by: two anonymous referees

References

Alauzet, F. and Mehrenberger, M.: P1-conservative solution interpolation on unstructured triangular meshes, Int. J. Numer. Meth. Engng., RR-6804, 1–48, 2009.

Farrell, P. and Maddison, J.: Conservative interpolation between volume meshes by local Galerkin projection, Comput. Meth. Appl. M., 200, 89–100, 2011.

Farrell, P., Piggott, M., Pain, C., Gorman, G., and Wilson, C.: Conservative interpolation between unstructured meshes via supermesh construction, Comput. Meth. Appl. M., 198, 2632–2642, 2009.

Fu, Y., Teng, J.-C., and Subramanya, S.: Node splitting algorithms in tree-structured high-dimensional indexes for similarity search, Proc. of SAC '02, 766–770, 2002.

Guo, D. X. and Drake, J. B.: A global semi-Lagrangian spectral model of the shallow water equations with variable resolution, J. Comput. Phys., 206, 559–577, 2005.

Jiao, X. and Heath, M.: Common-refinement-based data transfer between non-matching meshes in multiphysics simulations, Int. J. Numer. Meth. Eng., 61, 2402–2427, 2004.

Jones, P.: First- and second-order conservative remapping schemes for grids in spherical coordinates, Mon. Weather Rev., 127, 2204–2210, 1999.

Purser, R.: Non-standard grids, Proc. Seminar on Recent Developments in Numerical Methods for Atmospheric Modelling, Reading, UK, ECMWF, 44–72, 1998.

Sadourny, R.: Conservative Finite-Difference Approximations of the Primitive Equations on Quasi-Uniform Spherical Grids, Mon. Weather Rev., 100, 136–144, 1972.

Sadourny, R., Arakawa, A. K. I. O., and Mintz, Y. A. L. E.: Integration of the nondivergent barotropic vorticity equation with an icosahedral-hexagonal grid for the sphere1, Mon. Weather Rev., 96, 351–356, 1968.

Tomita, H., Tsugawa, M., Satoh, M., and Goto, K.: Shallow Water Model on a Modified Icosahedral Geodesic Grid by Using Spring Dynamics, J. Comput. Phys., 174, 579–613, 2001.

Ullrich, P. A., Lauritzen, P. H., and Jablonowski, C.: Geometrically Exact Conservative Remapping (GECoRe): Regular Latitude–Longitude and Cubed-Sphere Grids, Mon. Weather Rev., 137, 1721–1741, 2009.

White, D. and Jain, R.: Similarity indexing with the SS-tree, Proc. ICDE '96, 516–523, 1996.

Williamson, D. L.: The Evolution of Dynamical Cores for Global Atmospheric Models, J. Meteorol. Soc. Jpn, 85B, 241–269, 2007.