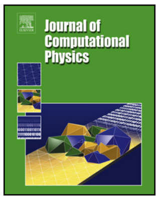
Journal of Computational Physics 302 (2015) 21–40

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp



Conservative interpolation of edge and face data on *n* dimensional structured grids using differential forms

Alexander Pletzer, David Fillmore

Tech-X Corp., 5621 Arapahoe Ave., Boulder, CO 80303, USA

A R T I C L E   I N F O

Article history:
Received 13 January 2015
Received in revised form 30 May 2015
Accepted 22 August 2015
Available online 5 September 2015

Keywords:
Interpolation
Regridding
Whitney forms
Structured grid
Conservation laws
Stokes' theorem
Mimetic
Differential form

A B S T R A C T

Interpolation methods for edge and face centered data are described, which preserve line and area integrals under regridding. These interpolation methods complement the multilinear nodal and conservative interpolation methods, which are widely used in climate data processing and other areas. The presented interpolation schemes ensure that curl-free and divergence-free fields remain so after regridding. These edge and face conservative interpolation methods are suitable for general curvilinear structured grids, including those with singular points (poles). Support for masked (invalid) regions is implicitly provided by attaching (partial) line/surface integral field values to cell edges/faces.

1. Introduction

Interpolation is an ubiquitous technique for inferring the values of a discretized field at user specified target locations. Many visualization packages, postprocessing tools, and partial differential equation solvers rely on interpolation. For instance, streamline computations require fields to be evaluated along trajectories [1,2] and finite volume discretizations must estimate fluxes at cell boundaries [3]. Therefore, it is not surprising that most postprocessing tools [4–6], infrastructure libraries [7,8], and scripting languages [9–11] support interpolation in one form or another. The Ultrascale Visualization-Climate Data Analysis (UV-CDAT) tools project [12], with which the authors are familiar, is an example of such a postprocessing tool. A recent survey of UV-CDAT users identified data interpolation from one grid to another as the second most sought after feature [personal communication from C. Doutriaux, June 1st 2012], ahead of visualization, statistics, parallel computing, and other postprocessing capabilities. The need for regridding arises in particular when comparing the output from different climate models [13], each with its own grid and resolution. Interestingly, the ability to compute accurate averages was the most important feature to users; we will see that interpolation and data averaging are in fact closely related.

Here, we are concerned with conservative interpolation [14,15], which is applicable to density fields for which the target element is not a point but a line, a surface, or a volume, depending on the type of field. By conservative interpolation, we mean that quantities associated with geometric objects are preserved under regridding if the original and target grids occupy the same *n*-dimensional space. Conserving fluids, materials, energy, voltage, and fluxes, is critical to many applications. In the case of fluxes (magnetic, material, or otherwise), the geometric object is a surface; in the case of voltage it is a line.

E-mail address: alexander@gokliya.net (A. Pletzer).

http://dx.doi.org/10.1016/j.jcp.2015.08.029
0021-9991/© 2015 Elsevier Inc. All rights reserved.

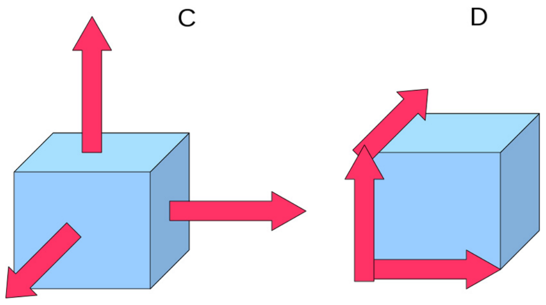


Fig. 1. The Arakawa C and D grids used by some climate modeling codes place wind velocity on faces (left) or on edges (right). The finite difference time domain method (FDTD) places the electric field on cell edges and the magnetic field on cell faces.

However, the idea is the same; integrating a quantify over a volume, surface, or line should give a consistent answer that does not depend on how the volume, surface, or line is discretized.

Conservative interpolation has gained considerable traction in some application domains, notably in climate science. Consider that climate change is the result of a tiny $< 0.1\%$ offset in the energy balance of the earth's atmosphere [16]. Assessing the effect of climate change would be very difficult, or impossible, without the ability to capture such a small discrepancy. It is therefore not surprising that, to our own estimate, 90% of the interpolation requests in UV-CDAT are for conservative interpolation, despite the fact that conservative interpolation is more numerically demanding than point interpolation.

In this article we argue that there are other types of interpolation schemes, which complement multilinear interpolation for point data and conservative interpolation for centered data. The interpolation methods we present are applicable to discretized vector fields where components are placed onto cell edges or faces. Such staggerings arise in finite volume [17] and mixed finite element discretization schemes [18–20]. In electromagnetism, the finite difference time domain (FDTD) method [21] dictates that the electric field be edge centered and the magnetic field face centered. Face and edge centered field arrangements are well known in the climate model community where they are referred to as Arakawa C/D grids (see Fig. 1). In fluid dynamics, the fluid velocity is often placed on cell faces [22]. Generally speaking, different field arrangements can lead to different numerical stability properties, with the consequence that misplacing fields can yield numerical pollution in spectral codes [23] or the checkerboard instability in time advancing codes [24].

It is important to realize that the interpolation method, whether it is conserving volume, surface, line integrals, or is a point interpolation method, is *not* a user choice but is entirely determined by the field properties. In other words, the underlying physics dictates the choice of the interpolation method. Such a viewpoint is shared by a growing number of people [25–27] who believe in mimetic, or structure preserving methods when solving partial differential equations (PDEs). Our approach is similar in spirit to these efforts except that we focus on interpolation and regridding instead of solving PDEs.

It has been known since the work of Whitney in 1957 how to recursively construct nodal, edge, face, and cell centered interpolating basis functions on tetrahedral simplices [28]. These Whitney basis functions have been re-discovered several times in the context of the finite element method [19,20]. However, it is only in the past 15 years that volume conservative interpolation methods have become widely available [15] in the climate simulation and some other communities. There appears to be no mention of surface and line conserving interpolation methods in climate and/or electromagnetic modeling. Therefore, one aim of this paper is to raise awareness of surface and line conserving interpolation methods in these communities.

To address the interpolation needs of these application areas, we have developed the Node, Edge, Face, And Cell Element (NEFACE) library, which implements exterior derivative calculus in spaces of arbitrary number of dimensions. Exterior calculus is an extension of vector calculus with a single derivative operator replacing the gradient, curl, and divergence operators. Exterior calculus relies on differential forms to represent fields and these are associated with integrals along an infinitesimal path, area, or volume element. There is a one-to-one correspondence between the staggering of a field and its integral representation. By expressing fields in terms of forms, one ensures that the gradient, Stokes', and the divergence theorems are satisfied. If in addition, line, surface, and volume integrals can be computed "exactly", as is the case for basis functions expressed as polynomials, then the basic elements are in place for interpolation schemes that conserve integrals to near machine accuracy. This addresses the need for accurate averaging procedures in addition to interpolation.

This paper is organized as follows. In Section 2 we give a brief introduction to the theory of differential forms, emphasizing the key aspects of the theory for interpolation. In Section 3 we present nodal, edge, and face interpolating forms for cuboid cells, which are suitable for general structured fields in n dimensions. In Section 4 we show how to project edge, face, and cell centered data onto line segments, triangles, and tetrahedra. In Section 5 we discuss the property of conservation. In Section 6 we apply NEFACE to line and surface integrations of discretized fields on singular, curvilinear grids in two to four dimensions. In Section 7 we summarize our results.

Table 1
Correspondence between the order of a differential form, the discretized field staggering, and vector calculus notation in 3D.

Order	Staggering	Differential form	Vector calculus	Example
0	Node	$\int f\ (=f)$	f	Temperature
1	Edge	$\int \alpha$	$\int \boldsymbol{\alpha}\cdot d\mathbf{x}$	Electric field **E**
2	Face	$\int \beta$	$\int \boldsymbol{\beta}\cdot d\mathbf{S}$	Magnetic field **B**
3	Cell	$\int \gamma$	$\int \gamma dV$	Mass density

2. Differential forms and the staggering of fields

The natural setting for conservative interpolation is the theory of differential forms and exterior calculus. Differential forms have recently attracted considerable attention as a way to numerically capture physical phenomena to high accuracy [29,30,25–27]. Compared to vector calculus, the language of differential forms has the advantage of extending to arbitrarily high dimensionality, of unifying Stokes' and Gauss' theorems, and of preserving differential calculus identities such as $\nabla \times \nabla = 0$.

2.1. What are p-forms?

Let $(x_1, x_2, \cdots x_n)$ denote the coordinates in an n dimensional space. A p-form

$$\omega = \sum \omega_{i_1,i_2,\cdots i_p} dx_{i_1} \wedge dx_{i_2} \wedge \cdots dx_{i_p} \tag{1}$$

is defined as a linear combination of $dx_{i_1} \wedge dx_{i_2} \wedge \cdots dx_{i_p}$ forms, each being a "wedge" product of 1-forms dx_i. The term $dx_{i_1} \wedge dx_{i_2} \wedge \cdots dx_{i_p}$ is analogous to an p-dimensional oriented hypervolume in n dimensions. Specifically, dx_i is a line element, $dx_i \wedge dx_j$ a surface element, $dx_i \wedge dx_j \wedge dx_k$ a volume element, etc. Like the cross product, the wedge product of two dx forms is anticommutative, $dx_i \wedge dx_j = -dx_j \wedge dx_i$. The anticommutative property implies (and takes automatically into account) the orientation of edges and faces. Unlike the cross product in 3D, the wedge product can be defined in any number of dimensions.

A zero form (0-form) is just a function of space and a 1-form, $\sum_i \omega_i dx_i$, corresponds to a vector field. Table 1 shows the correspondence between differential form and vector calculus notation. Note that differential forms carry the differential elements (e.g. dx_i or $dx_i \wedge dx_j$) over which to integrate. Zero forms have no differential element attached and therefore the integral applied to a 0-form has no effect ($\int f = f$).

In n dimensions, 1-forms are attached to the cells' oriented edges. In 3D, 2-forms are attached to oriented faces and 3-forms to oriented volumes. The correspondence between p-forms and staggering comes out naturally by letting dx_i represent the variation of coordinate x_i along the edge of a cell. Hence, edge fields are line integrals of vector fields along cell edges, face fields are area integrals of vector fields on cell faces, and cell centered fields are discrete, cell volume integrals of densities. There is a one-to-one correspondence between the order of a form and its staggering (see Table 1).

From the definition of the gradient, $\int \nabla f \cdot d\mathbf{x} = \int df$, we see that ∇f maps to df in differential form notation. Hence, $\sum_i \omega_i dx_i \rightarrow \sum_i \omega_i \nabla x_i$ becomes an expansion in contravariant basis vectors ∇x_i. In 3D, a 2-form $\sum_i \sum_{j>i} \omega_{ij} dx_i \wedge dx_j$ represents an expansion in $\nabla x_i \times \nabla x_j$ basis vectors.

For many coordinates systems, expressions for ∇x_i are known. In the case of a cylindrical coordinate system, $(x_1, x_2, x_2) = (\rho, \theta, z)$ for instance, we have $\nabla \rho = \hat{\boldsymbol{\rho}}$, $\nabla \theta = \hat{\boldsymbol{\theta}}/\rho$, and $\nabla z = \hat{\mathbf{z}}$. For arbitrary coordinate systems on the other hand, the contravariant basis vectors might need to be estimated numerically. Below we show how the ∇x_i's can be expressed in terms of the covariant basis vectors $\partial \mathbf{x}/\partial x_i$, which approximate the cell edge lengths.

From

$$dx_i = \nabla x_i \cdot d\mathbf{x} = \sum_{\ell} \nabla x_i \cdot \frac{\partial \mathbf{x}}{\partial x_\ell} dx_\ell, \tag{2}$$

one gets the orthogonality relation

$$\nabla x_i \cdot \frac{\partial \mathbf{x}}{\partial x_\ell} = \delta_{i,\ell} \tag{3}$$

between contravariant (∇x_i) and covariant ($\frac{\partial \mathbf{x}}{\partial x_\ell}$) basis vectors. Therefore,

$$\frac{\partial \mathbf{x}}{\partial x_i} = J \nabla x_j \times \nabla x_k \tag{4}$$

must be orthogonal to the contravariant ∇x_j and ∇x_k, with i, j, and k in cyclic order and

$$J^{-1} \equiv (\nabla x_i \times \nabla x_j) \cdot \nabla x_k \tag{5}$$

Table 2
Number of independent p-form terms ($p = 0$, 1, 2, 3, and 4) in 1D, 2D, 3D, and 4D. Regardless of the dimensionality, a 0-form has a single component, its proxy is a scalar field. A 1-form has n components in n dimensions, its proxy is a vector field. It is only in 3D that a 2-form has n components.

Dimensions	Order 0	Order 1	Order 2	Order 3	Order 4
1D	1	1			
2D	1	2	1		
3D	1	3	3	1	
4D	1	4	6	4	1

being the inverse Jacobian. Hence, the vector calculus proxy of $\sum_i \sum_{j>i} \omega_{ij} dx_i \wedge dx_j$ is $\sum_k \omega_{ij} J^{-1} \frac{\partial \mathbf{x}}{\partial x_k}$. We emphasize that (3) is valid for *any* coordinate systems, including non-orthogonal ones — contravariant and covariant vectors are always mutually orthogonal.

Vector fields associated with 2-forms in 3D are sometimes called pseudo-vector or axial vector fields, to distinguish them from “true” or “polar” vector fields (1-forms). Pseudo-vector fields do not obey the rules of physics under reflection. An example is provided by a loop carrying a current, which generates a magnetic field. The reflection of the loop as seen through a mirror would show a magnetic field that is opposite to the physically correct direction.

In more than three dimensions, it is no longer possible to associate 2-forms with vector fields. Because the wedge product of a form with itself is zero, the number of possible terms in the expansion of a p-form is given by the binomial coefficient $\frac{n!}{(n-p)!p!}$, see Pascal's triangle (Table 2). We observe that in 4D a 2-form has six independent terms.

2.2. The exterior derivative

In place of the ∇, $\nabla\times$, and $\nabla\cdot$ operators, the language of differential forms has only a single derivative operator, d, which can be regarded as the “usual” differential operator applying to 0-forms but now extended to p-forms. The exterior derivative d is linear, $d(\omega_1 + \omega_2) = d\omega_1 + d\omega_2$, obeys Leibniz's rule, $d(\omega_1 \wedge \omega_2) = d\omega_1 \wedge \omega_2 + \omega_1 \wedge d\omega_2$, and applying twice d, $d(d\alpha) = 0$, yields a property analogous to $\nabla \times \nabla = 0$ and $\nabla \cdot \nabla\times = 0$. Operator d is called the exterior derivative because it increases the order of the form, as in

$$d(\omega dx_j) = d\omega \wedge dx_j = \sum_i \frac{\partial \omega}{\partial x_i} dx_i \wedge dx_j. \tag{6}$$

In 3D, the familiar expressions for the gradient, curl, and divergence can be recovered. Applying d to a 0-form f yields the gradient operator

$$df = \frac{\partial f}{\partial x_1} dx_1 + \frac{\partial f}{\partial x_2} dx_2 + \frac{\partial f}{\partial x_3} dx_3. \tag{7}$$

The exterior derivative applied to a 1-form $\alpha = \alpha_1 dx_1 + \alpha_2 dx_2 + \alpha_3 dx_3$ amounts to taking the curl,

$$\begin{aligned} d\alpha = &\left(\frac{\partial \alpha_2}{\partial x_3} - \frac{\partial \alpha_3}{\partial x_2}\right) dx_2 \wedge dx_3 \\ &+ \left(\frac{\partial \alpha_3}{\partial x_1} - \frac{\partial \alpha_1}{\partial x_3}\right) dx_3 \wedge dx_1 + \left(\frac{\partial \alpha_1}{\partial x_2} - \frac{\partial \alpha_2}{\partial x_1}\right) dx_1 \wedge dx_2. \end{aligned} \tag{8}$$

And applying d to a 2-form $\beta = \beta_1 dx_2 \wedge dx_3 + \beta_2 dx_3 \wedge dx_1 + \beta_3 dx_1 \wedge dx_2$ gives the divergence,

$$d\beta = \left(\frac{\partial \beta_1}{\partial x_1} + \frac{\partial \beta_2}{\partial x_2} + \frac{\partial \beta_3}{\partial x_3}\right) dx_1 \wedge dx_2 \wedge dx_3. \tag{9}$$

2.3. Remarks

Some remarks are in order. First, expressions (7)–(9) naturally extend to n dimensions. Second, the forms for the gradient, curl, and divergence operator do not change with the coordinate system. Differential forms therefore enforce a clear separation between topology and geometry, making it possible to work in different coordinate systems without the need to modify the underlying equations. This invariance derives from the fact, mentioned previously, that differential forms carry the (line, area, volume) integration elements with them so that forms in some sense represent line, area, and volume integrations. Because of this invariance, one is free to choose the coordinate system supporting the Whitney forms.

In Sections 6.1–6.3, we will see that working with forms, or their numerical approximation, brings additional benefits. Unlike their proxy scalar and vector fields, forms are well behaved in the vicinity of coordinate singularities.

In the context of discretized fields attached to a mesh, the term $dx_{i1} \wedge \cdots x_{ip}$ represents a basis function, which is different for nodal, edge-centered, face-centered, and cell-centered fields. Expressions for each type of basis function will be provided in Section 3.

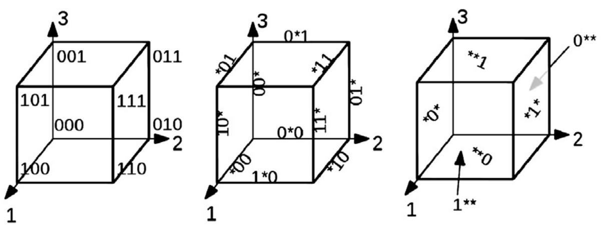


Fig. 2. Each grid element (node, edge, face, and cell) is uniquely identified with array $\boldsymbol{\sigma}$ taking one of three values. Indexing of nodes (left), edges (middle), and faces (right). A cell has $\boldsymbol{\sigma} = [***]$.

Finally, it is also noteworthy to mention that Stokes' and the divergence theorems take a particularly elegant form,

$$\int_{\Omega} d\omega = \oint_{\partial\Omega} \omega. \tag{10}$$

In the above, we have erased the distinction between the gradient, curl, and divergence operators, all these operators behave in similar manner. The generalized Stokes' theorem expresses the fact that the (hyper-)volume integral of an exterior derivative applied to a form can be obtained by integrating the form over the boundary of the volume. In the next section we will show how to construct basis functions that satisfy (10).

3. Interpolating basis functions on cuboids

In this section, we show how to generate basis functions for edge and face data starting from the well known multilinear nodal basis function formulas. However, in order to accomplish this task, we must first devise a way to uniquely identify all the nodes, edges, faces, which contribute to a cuboid in n dimensions.

Let $\boldsymbol{\sigma} = [\sigma_1, \cdots \sigma_n]$ be a list where each σ_i takes one of three values, $\sigma_i \in \{0, 1, *\}$. We will use the value 0 to indicate that an element is located on the low side of the cell, 1 if the element is on the high side of the cell, and $*$ if the element varies along the axis. For instance, in 1D, the lower end vertex has $\boldsymbol{\sigma} = [0]$, the upper end vertex $\boldsymbol{\sigma} = [1]$, and the edge $\boldsymbol{\sigma} = [*]$. Fig. 2 shows how a unique $\boldsymbol{\sigma}$ can be associated with each node (left), edge (middle), and face (right) of a 3D cell. This numbering system generalizes to n dimensions.

The number of nodes, edges, faces, etc. depends on the number of combinations of 0's, 1's, and $*$'s. For a certain type of element k where $k = 0$ for nodes, 1 for edges, 2 for faces, etc., there are k $*$'s in the $\boldsymbol{\sigma}$ list and the number of elements is given by the formula

$$\frac{2^{n-k} n!}{(n-k)!k!}. \tag{11}$$

We can readily check that this formula yields the correct number of nodes in 3D: 8 nodes, 12 edges, and 6 faces. In 4D, there are 16 nodes, 32 edges, 24 faces, and 8 hyper-faces.

Given this numbering system in place, we can write interpolation as the sum

$$\bar{f} = \sum_{\boldsymbol{\sigma}} w_{\boldsymbol{\sigma}} f_{\boldsymbol{\sigma}} \tag{12}$$

where $w_{\boldsymbol{\sigma}}$ are the interpolation weights and $f_{\boldsymbol{\sigma}}$ the field values associated with cell element $\boldsymbol{\sigma}$.

Two remarks are in order. First, it is important to realize that the $f_{\boldsymbol{\sigma}}$'s are scalars regardless of whether the field is a vector field or not. For edge centered fields, $f_{\boldsymbol{\sigma}}$ represents a line integral along edge $\boldsymbol{\sigma}$. In the case of face-centered data, $f_{\boldsymbol{\sigma}}$ is an area integral. For edge and face data, the interpolation weights will be expressed in terms of possibly nested integrals (more about this in Section 4). Second, the target element of the interpolation varies according to the staggering. The target is a point for nodal fields, a line for edge fields, an area for face fields, etc.

Finally, we will find it convenient to express the basis functions in terms of their parametric coordinate representation $\boldsymbol{\xi}$, which is local to the cell (Fig. 3). Such a local coordinate system is widely used in the finite element community to compute the stiffness matrix. Here, it will help us break the interpolation problem into (i) finding the cell that contains the target element and (ii) computing the interpolation weights for each of the cell elements. The problem of locating the cell containing a given target element will not be further discussed here since it is the same for any interpolation scheme.

We will start with nodal interpolation since most readers are familiar with this type of interpolation, moving to edge and face interpolation in Sections 3.2 and 3.3. Cell interpolation can be covered in the same way as edge and face interpolation although we will not dwelve on this since ample references exist on this subject, e.g. [15,31,32].

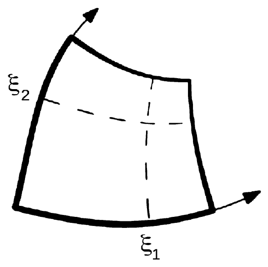


Fig. 3. The parametric coordinates $0 \le \xi_i \le 1$ for a quadrilateral. For structured grids, the parametric coordinates are the differences between the index position and the cell index set.

3.1. Nodal basis functions

For nodal interpolation, the weights are just the basis functions $\phi_{\boldsymbol{\sigma}}$ evaluated at the target position $\hat{\mathbf{x}}$, i.e. $w = \phi_{\boldsymbol{\sigma}}(\hat{\mathbf{x}})$. In Appendix A.1 we list the multilinear basis functions for the 3D case. These basis functions can be seen to generalize to n dimensions with the formula

$$\phi_{\boldsymbol{\sigma}} = \prod_{i=1}^{n} \left[(1-\sigma_i)(1-\xi_i) + \sigma_i \xi_i\right]. \tag{13}$$

It can be seen that the $w_{\boldsymbol{\sigma}}$'s have the property to be one on one node and zero on all other cell nodes,

$$\phi_{\boldsymbol{\sigma}}(\mathbf{x}_{\boldsymbol{\sigma}'}) = \delta_{\boldsymbol{\sigma},\boldsymbol{\sigma}'} \tag{14}$$

where $\mathbf{x}_{\boldsymbol{\sigma}'}$ are the coordinates of vertex $\boldsymbol{\sigma}'$. In addition, $\sum_{\boldsymbol{\sigma}} w_{\boldsymbol{\sigma}} = 1$.

3.2. Edge basis functions

Edge elements have the property that $\boldsymbol{\sigma}$ contains a single $*$ with all other σ_i being either 0 or 1. The index j for which $\sigma_j = *$ is the direction of the edge. Edge basis functions $\phi^{(1)}_{\boldsymbol{\sigma}}$ are sought which satisfy a property similar to (14)

$$\int\limits_{\boldsymbol{\sigma}'} \phi_{\boldsymbol{\sigma}} = \delta_{\boldsymbol{\sigma},\boldsymbol{\sigma}'} \tag{15}$$

except that the above is expressed in terms of line integrals along edges. Equation (15) states that the line integral of the basis function attached to edge $\boldsymbol{\sigma}$ is zero on all edges $\boldsymbol{\sigma}'$ except on edge $\boldsymbol{\sigma}$, where the integral is one.

Edge basis functions satisfying (15) can be constructed from (13) by replacing the jth term in the product by $d\xi_j$,

$$\begin{aligned}\phi^{(1)}_{\boldsymbol{\sigma}} = &\left[(1-\sigma_1)(1-\xi_1)+\sigma_1\xi_1\right]\cdots\left[(1-\sigma_{j-1})(1-\xi_{j-1})+\sigma_{j-1}\xi_{j-1}\right]\\ &\left[(1-\sigma_{j+1})(1-\xi_{j+1})+\sigma_{j+1}\xi_{j+1}\right]\cdots\left[(1-\sigma_n)(1-\xi_n)+\sigma_n\xi_n\right]\\ &d\xi_j.\end{aligned} \tag{16}$$

Note that in doing so, $\phi^{(1)}_{\boldsymbol{\sigma}}$ is no longer scalar but a 1-form (or equivalently a vector basis function after substituting $d\xi_j$ with $\nabla\xi_j$). Appendix A.2 lists the edge vector basis functions in 3D.

3.3. Face and other staggered basis functions

Face elements have two directions, j_1 and j_2, along which the element $\boldsymbol{\sigma} = [\cdots, *, \cdots, *, \cdots]$ varies. Face basis functions can be obtained by replacing the two terms j_1 and j_2 by $d\xi_{j_1} \wedge d\xi_{j_2}$,

$$\begin{aligned}\phi^{(2)}_{\boldsymbol{\sigma}} = &\left[(1-\sigma_1)(1-\xi_1)+\sigma_1\xi_1\right]\cdots\\ &\left[(1-\sigma_{j_1-1})(1-\xi_{j_1-1})+\sigma_{j_1-1}\xi_{j_1-1}\right]\\ &\left[(1-\sigma_{j_1+1})(1-\xi_{j_1+1})+\sigma_{j_1+1}\xi_{j_1+1}\right]\cdots\\ &\left[(1-\sigma_{j_2-1})(1-\xi_{j_2-1})+\sigma_{j_2-1}\xi_{j_2-1}\right]\\ &\left[(1-\sigma_{j_2+1})(1-\xi_{j_2+1})+\sigma_{j_2+1}\xi_{j_2+1}\right]\cdots\\ &\left[(1-\sigma_n)(1-\xi_n)+\sigma_n\xi_n\right]d\xi_{j_1}\wedge d\xi_{j_2}.\end{aligned} \tag{17}$$

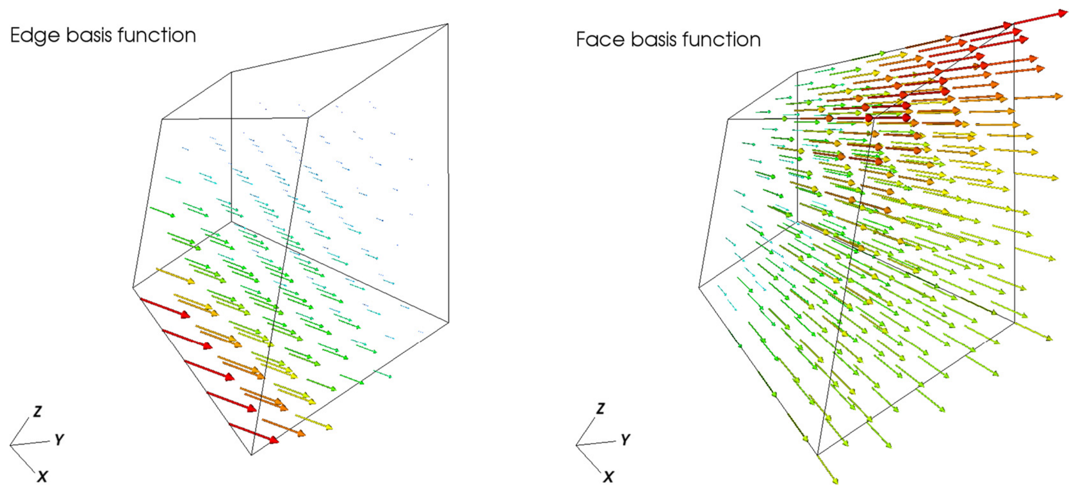


Fig. 4. Edge and face vector basis functions for a general hexahedral cell. Left: the edge basis function for the lower left edge is either zero or perpendicular to all other edges. Right: the face basis function is zero or tangential to all other faces.

As in Section 2, the vector equivalent to (17) can be obtained by replacing $d\xi_{j_1} \wedge d\xi_{j_2}$ by $\nabla\xi_{j_1} \times \nabla\xi_{j_2}$ in 3D (see Appendix A.3). The so obtained basis functions are compatible with the Van Welij [20] finite elements. Orthogonality condition (15) applies to the $\phi^{(2)}$ bases after interpreting the integral as extending over cell faces.

Clearly, the process of constructing staggered basis functions can be pursued for arbitrary k-elements $(0 \le k \le n)$ by replacing each $*$ in $\boldsymbol{\sigma}$ by $d\xi$ and wedging the result. Cell basis functions can in particular be obtained this way. In 3D, there is a single basis function $\phi^{(3)} = d\xi_1 \wedge d\xi_2 \wedge d\xi_3$, or $\nabla\xi_1 \times \nabla\xi_2 \cdot \nabla\xi_3$ in vector notation, a quantity which is inversely proportional to the volume of the cell in $\boldsymbol{\xi}$ space.

The general orthogonality property becomes integral

$$\int\limits_{\boldsymbol{\sigma}'} \phi^{(k)}_{\boldsymbol{\sigma}} = \delta_{\boldsymbol{\sigma},\boldsymbol{\sigma}'} \tag{18}$$

where the integral is taken over element $\boldsymbol{\sigma}'$. Similar orthogonality properties exist for the Nédélec family of finite element on triangles and tetrahedra [19].

Property (18) remains valid even in the case of highly distorted cells, i.e. cells for which edges intersect with non-orthogonal angles. Fig. 4 shows that the requirement of vanishing integral on neighboring edges (faces) forces the field to be either perpendicular to neighboring edges in the case of edge centering, or tangential to neighboring faces for face centered basis functions. The vector basis function associated with an edge will generally not be tangent to the edge, and the vector basis attached to a face will not typically be perpendicular to the face, except in the case of cells with orthogonal axes.

The accuracy of linear face interpolation in 3D is similar to linear interpolation of a nodal field in 1D. Let F represent the flux between two opposite faces then the maximum error $F \le \frac{1}{8}h^2 F''_{max}$ with F'' being the second order derivative of the flux in the direction of variation and h the cell size in this direction. For edge interpolation, the error is comparable to that of bilinear interpolation since in this case there are two directions where the line integral varies in 3D.

It is known that the error of nodal, edge, face, and cell centered fields can significantly degrade for cells that are not affine transformations of the n-cube [33–35]. This is not an issue here as the edge and face interpolation target objects are line and area integrals, which are invariant with respect to orientation preserving transformations, as discussed in Section 2.3. Thus, any logically rectangular space will give the same integral and we are free to choose the space in which we compute the integrals to our convenience (selecting for instance the index space where all the cells are cubes).

4. Analytic computation of integrals

In Section 3, the edge and face basis functions were derived for cuboids in n dimensions. Although these basis functions can be employed to determine the flux density or other quantities at given points, it is often more important, from a physical viewpoint, to project the basis forms onto geometric objects of dimensionality compatible with the order of the form. For instance, one might want to estimate the total mass contained within a certain volume, or compute the total flux across a given surface. In this section, we show how to *analytically* compute integrals of forms projected onto geometric objects (1-forms onto lines, 2-forms onto areas, etc.). Specifically, the integration will not involve any quadrature scheme. The target object will be a simplex that is fully contained within a grid cell.

Analytic integration of a p-form onto a p dimensional simplex can be performed if each component of the p-form is a polynomial. Let $\omega(\boldsymbol{\xi})d\xi_{i1} \wedge \cdots d\xi_{ip}$ be one such component of the p-form (each component contributing linearly to the total p-form projection), and

$$S(\boldsymbol{\lambda}) \equiv \boldsymbol{\xi}_{s0} + \sum_{j=1}^{p} \lambda_j (\boldsymbol{\xi}_{sj} - \boldsymbol{\xi}_{s0}) \tag{19}$$

be the parametric representation of the simplex in the local $\boldsymbol{\xi}$-space of the cell with $0 \le \lambda_1 < 1$, $0 \le \lambda_2 < 1 - \lambda_1, \cdots, 0 \le \lambda_p \le 1 - \sum_{j=1}^{p-1} \lambda_j$, then

$$\int \omega(\boldsymbol{\xi}) = V_s \int_0^1 \int_0^{1-\lambda_1} \cdots \int_0^{1-\sum_{j=1}^{p-1}\lambda_j} \omega \circ S(\boldsymbol{\lambda}) \tag{20}$$

is a nested sequence of integrals, with $\circ$ denoting function composition, i.e. $\omega \circ S(\boldsymbol{\lambda})$ is ω evaluated on simplex $S(\boldsymbol{\lambda})$. In the above, V_s is equal to $p!$ times the "volume" of the p dimensional simplex, this quantity can be computed from the determinant

$$V_s = \det\left(\boldsymbol{\xi}_{s1} - \boldsymbol{\xi}_{s0}, \quad \boldsymbol{\xi}_{s2} - \boldsymbol{\xi}_{s0}, \quad \cdots \quad \boldsymbol{\xi}_{sp} - \boldsymbol{\xi}_{s0}\right). \tag{21}$$

For readers who are unfamiliar with differential forms and prefer the more traditional vector calculus approach, Appendix A reproduces the above formula for nodal, edge, and face centered fields in 3D.

To see how (20) works, consider first the simple case of a 0-form projected onto a point in 2D and let ω represent one of the nodal basis functions, e.g. $\xi_1(1-\xi_2)$, with the target simplex $S(\boldsymbol{\lambda}) = \boldsymbol{\xi}_{s0}$ being just a point. Integral (20) then reduces simply to evaluating ω at the location $\boldsymbol{\xi}_{s0}$. Thus, (20) collapses to what is generally known as the interpolation of a nodal field.

Next, consider the 1-form $\omega = \phi^{(1)}_{1*} = \xi_1 d\xi_2$ projected onto line $S(\boldsymbol{\lambda}) = \boldsymbol{\xi}_{s0} + \lambda_1(\boldsymbol{\xi}_{s1} - \boldsymbol{\xi}_{s0})$. In this case (20) becomes

$$\int_0^1 \left[\xi_{s0,1} + \lambda_1(\xi_{s1,1} - \xi_{s0,1})\right]\left(\xi_{s1,2} - \xi_{s0,2}\right) d\lambda_1 = (\xi_{s1,2} - \xi_{s0,2})\frac{1}{2}(\xi_{s0,1} + \xi_{s1,1}) \tag{22}$$

where $\xi_{s0,1}$ is the first component of $\boldsymbol{\xi}_{s0}$, $\xi_{s1,2}$ is the second component of $\boldsymbol{\xi}_{s1}$, etc. Thus, we see that the integral represents the projection of the vertical basis function field evaluated at the mid horizontal position of the segment. Integrals for other basis functions, e.g. $\phi^{(1)}_{*0}$, can be obtained by switching indices and replacing ξ by $1-\xi$.

It should be observed that (20) can still be applied when the target is of dimensionality incompatible with the order of the form, however the result will be zero. Projecting a 1-form onto a point, for instance, will be zero because there is no λ to integrate over (or alternatively the λ range to integrate over is zero). Likewise projecting a 2-form such as $d\xi_1 \wedge d\xi_2$ onto a line also gives zero, this time because $d\xi_1$ and $d\xi_2$ are $\propto d\lambda_1$ and hence $\int d\xi_1 \wedge d\xi_2 \propto \int d\lambda_1 \wedge d\lambda_1 = 0$.

It should be apparent that projecting a p-form onto simplex follows a recursive pattern once a generic integration procedure is written. Let the target be a p dimensional simplex, then the integrand will be a polynomial function of λ_1, $\lambda_2 \cdots \lambda_p$, for which the primitive can easily be found. Each definite integration will thus remove a variable λ_j, a process which can be continued until no more variables to integrate over remain.

Equation (20) requires the target simplex to be fully inside a grid cell. For targets that extend over many cells, the target must first be tessellated so that each tetrahedron entirely fits into a cell. Tessellation in n dimensions can be performed using standard algorithms, for instance the Quickhull method of Ref. [36]. The additional face flipping step of Delaunay tetrahedralization software can be dispensed with since we are not concerned with the quality of the generated tetrahedra, our only requirement is that each tetrahedron be contained within a grid cell.

In order to tessellate, all the intersection points between the simplex and the grid cells must first be obtained. In 3D and for a tetrahedron target, these points include the tetrahedron vertices, the intersections of the tetrahedron edges with cell faces, the intersection of the tetrahedron faces with the cell edges, and the grid nodes contained within the tetrahedron. We note that each simplex element (node, edge, face, ...) will intersect a cell element with complementary dimensionality, which we term the co-cell element. The sum of the simplex element dimensionality and the co-cell element dimensionality must add to the number of parametric grid dimensions n. By doing so we ensure that the system of equations to solve for the intersection point is always $n \times n$ (see Algorithm 1). However, the system of equations can be singular when a target simplex edge runs tangentially to the cell face for instance. To avoid such situations, we recommend to perturb the vertices of the simplex by some small amount. The overall effect on interpolation of the vertex perturbations will remain negligible provided the edge lengths and face areas are not significantly changed.

Each intersection point found by Algorithm 1 must in addition lie within the simplex element *and* the co-cell element and we can check this by testing that the values of the simplex element parameters $\lambda_1 \cdots \lambda_k$ satisfy the tetrahedron conditions $0 \le \lambda_1 < 1$, $0 \le \lambda_2 < 1-\lambda_1, \cdots, 0 \le \lambda_k < 1-\sum_{i=1}^{k-1} \lambda_{k-1}$, and the co-cell element conditions $0 \le \mu_j < 1$ $(j = 1, \cdots n-k)$.

Algorithm 1 Computing the intersection point between a simplex element and a co-cell element.

```
1: function computeIntersectionPoint($se$, $ce$)
2:                                                     ▷ $se$ is a simplex element (node, edge, etc.)
3:                                                     ▷ $ce$ is a co-cell element (cell, face, etc.)
4:     $k = \mathrm{getSimplexElementOrder}(se)$          ▷ 0 for nodes, 1 for edges, etc.
5:     $n = \mathrm{getCoCellElementOrder}(ce) + k$
6:     $\left[\xi_s^{(0)}, \cdots \xi_s^{(k)}\right] = \mathrm{getSimplexElementVertices}(se)$
7:     $\left[\xi_c^{(0)}, \cdots \xi_c^{(n-k)}\right] = \mathrm{getCoCellElementVertices}(ce)$
8:     build system $\left[\xi_s^{(1)} - \xi_s^{(0)}, \cdots \xi_s^{(k)} - \xi_s^{(0)}, \xi_c^{(0)} - \xi_c^{(1)}, \cdots \xi_c^{(0)} - \xi_c^{(n-k)}\right] \cdot [\lambda_1, \cdots \lambda_k, \mu_1 \cdots \mu_{n-k}]^T = \left[\xi_c^{(0)} - \xi_s^{(0)}\right]$
9:     solve system
10:    return $[\lambda_1, \cdots \lambda_k, \mu_1 \cdots \mu_{n-k}]$
11: end function
```

Algorithm 2 Finding all the intersection points between a simplex and a grid.

```
1: function findAllIntersectionPoints($simplex$, $grid$)
2:     $points = [\ ]$                                  ▷ empty list
3:     $n = \mathrm{getNumberOfParametricDimensions}(grid)$
4:     $order = \mathrm{getOrder}(simplex)$
5:     for $gridCell$ in $grid$ do
6:         for $k$ in $0 \cdots order$ do
7:             $simplexElements = \mathrm{getSimplexElements}(simplex, k)$
8:             $coCellElements = \mathrm{getCoCellElements}(gridCell, n-k)$
9:             for $se$ in $simplexElements$ do
10:                for $ce$ in $coCellElements$ do
11:                    $point = \mathrm{computeIntersectionPoint}(se, ce)$
12:                    if $point$ in $se$ and $point$ in $ce$ then
13:                        append $point$ to $points$
14:                    end if
15:                end for
16:            end for
17:        end for
18:    end for
19:    return $points$
20: end function
```

Note that finite floating point accuracy could lead to double counting or misses for simplices that are co-linear to faces. To reduce the sensitivity to floating point comparisons, we also recommend in this case to perturb the above conditions by a number comparable to machine epsilon. With these caveats in mind, the intersection points can be collected using Algorithm 2.

5. Conservative property of the edge and face interpolation

Expressions for the basis functions and formulas for projecting edge and face fields to a simplex were derived in Sections 3 and 4. In this section we will apply these formulas to demonstrate that the edge and face interpolations are conservative.

An interpolation is conservative if it preserves the physical properties of the field. Specifically, an edge interpolation method applied to a gradient is conservative if the projection of the field's gradient (df) along an arbitrary path S

$$\int_S df \approx \sum_{\boldsymbol{\sigma}} f_{\boldsymbol{\sigma}} \int_S d\phi_{\boldsymbol{\sigma}} \tag{23}$$

depends only on the field values at the integration endpoints [see (10)],

$$\sum_{\boldsymbol{\sigma}} f_{\boldsymbol{\sigma}} \int_S d\phi_{\boldsymbol{\sigma}} = \sum_{\boldsymbol{\sigma}} f_{\boldsymbol{\sigma}} \oint_{\partial S} \phi_{\boldsymbol{\sigma}} = \bar{f}(end) - \bar{f}(start) \tag{24}$$

where $\boldsymbol{\sigma}$ runs across all the cell nodes and $\bar{f}$ represents the nodal interpolated value of the zero-form f at the start and end positions of S. Note that the above should also be equal to

$$\sum_{\boldsymbol{\sigma}} f_{\boldsymbol{\sigma}} \oint_{\partial S} \phi_{\boldsymbol{\sigma}} = \sum_{\boldsymbol{\sigma}'} F_{\boldsymbol{\sigma}'} \int_S \phi^{(1)}_{\boldsymbol{\sigma}'} \tag{25}$$

where $\boldsymbol{\sigma}'$ runs across all the cell edges and $F_{\boldsymbol{\sigma}'}$ are the line integrals of df along each edge $\boldsymbol{\sigma}'$.

To show that (25) is satisfied, we need only consider the case where S is a straight line segment $\xi_i = \xi_{si} + \lambda(\xi_{ei} - \xi_{si})$ since it is always possible to approximate an arbitrary path as a collection of possibly infinitesimal straight segments. Moreover, we will concentrate on the 2D case where $F_{*0} = f_{10} - f_{00}$, $F_{*1} = f_{11} - f_{01}$, etc. — the calculation readily generalizes to higher dimensions although the algebra becomes cumbersome.

Using (22) we get

$$\begin{aligned}\sum_{\sigma'} F_{\sigma'} \int_S \phi^{(1)}_{\sigma'} = {} & (f_{01} - f_{00}) \left[1 - \frac{1}{2}(\xi_{s1} + \xi_{e1})\right] (\xi_{e2} - \xi_{s2}) \\ & + (f_{11} - f_{10}) \frac{1}{2}(\xi_{s1} + \xi_{e1})(\xi_{e2} - \xi_{s2}) \\ & + (f_{10} - f_{00}) \left[1 - \frac{1}{2}(\xi_{s2} + \xi_{e2})\right] (\xi_{e1} - \xi_{s1}) \\ & + (f_{11} - f_{01}) \frac{1}{2}(\xi_{s2} + \xi_{e2})(\xi_{e1} - \xi_{s1}),\end{aligned}$$

which, after some manipulations, reduces to

$$\begin{aligned}\sum_{\sigma'} F_{\sigma'} \int_S \phi^{(1)}_{\sigma'} = {} & f_{00} \left[(1 - \xi_{e1})(1 - \xi_{e2}) - (1 - \xi_{s1})(1 - \xi_{s2})\right] \\ & + f_{01} \left[\xi_{e1}(1 - \xi_{e2}) - \xi_{s1}(1 - \xi_{s2})\right] \\ & + f_{11} \left[\xi_{e1}\xi_{e2} - \xi_{s1}\xi_{s2}\right] + f_{10} \left[\xi_{e1}(1 - \xi_{e2}) - \xi_{s1}(1 - \xi_{s2})\right],\end{aligned}$$

an expression that is clearly equivalent to (24).

To show that face interpolation is conservative, a similar approach is taken except that we now require

$$\sum_{\sigma} f_{\sigma} \int_S d\phi^{(1)}_{\sigma} = \sum_{\sigma'} F_{\sigma'} \int_S \phi^{(2)}_{\sigma'} \tag{26}$$

with S being a triangle, $\xi_i = \xi_{si} + \lambda(\xi_{ui} - \xi_{si}) + \mu(\xi_{vi} - \xi_{si})$, and $\phi^{(1)}$ ($\phi^{(2)}$) being the edge (face) basis function. In 2D, we have $d\phi^{(1)}_{*0} = -d\phi^{(1)}_{*1} = -d\phi^{(1)}_{0*} = d\phi^{(1)}_{1*} = d\xi_1 \wedge d\xi_2$ and thus (26) yields $(f_{0*} + f_{1*} - f_{*1} - f_{0*}) \int_S d\xi_1 \wedge d\xi_2 = F_{**} V_s/2$ with $V_s = (\xi_{u1} - \xi_{s1})(\xi_{v2} - \xi_{s2}) - (\xi_{v1} - \xi_{s1})(\xi_{u2} - \xi_{s2})$ given by (21). We observe that F_{**} is just the circulation of the edge field f over the face, thus proving that the 1-form bases satisfy a conservation law. Similar formulas can be derived for 3D and higher dimensions.

6. Applications

In the following, we provide examples of line and surface integrals in two, three, and four dimensions. In Sections 6.1–6.2 we focus on edge-centered fields in cylindrical and spherical geometry. In Section 6.3 we demonstrate face field interpolation and conservation of the electrostatic flux. In Section 6.4 we show how to handle face-centered fields with partially valid cells and in Section 6.5 we apply our conservative face interpolation to a four dimensional, space–time problem.

6.1. Line integral of an edge field on a polar grid

In our first test case, we integrate the magnetic 1-form

$$\int H = \int \frac{I}{2\pi} d\theta \tag{27}$$

produced by a current filament I along some arbitrary path in polar (ρ, θ). Here, ρ is the radial distance from the filament and θ the poloidal angle along the H field.

If we choose our grid to be aligned to the (ρ, θ) coordinates, then line integral (27) starting at (ρ_i, θ_i) and finishing at (ρ_f, θ_f) simply reduces to evaluating $\frac{I}{2\pi}(\theta_f - \theta_i)$. In other words, the line integral, which does not depend on ρ_i or ρ_f, can be computed exactly in trivial manner.

Contrast this to using a point interpolation method, which requires segmenting the path and approximating the integral using, e.g. the trapezoidal rule. Along each mid-segment location, we would evaluate the field $\mathbf{H} = \frac{I}{2\pi\rho}\hat{\theta}$. Note that this field is singular as ρ approaches zero, making the integral ill defined in this limit.

Fig. 5 shows the average error of the point interpolation method versus the radial number of grid cells along a radial path starting at $\rho = 0.01$ and finishing at $\rho = 1$. The first node ($\rho = 0.01$) was chosen to be a small distance away from the singularity. Large errors are found at coarse resolution — it is only when the resolution is finer than $\rho = 0.01$ that the method exhibits quadratic convergence.

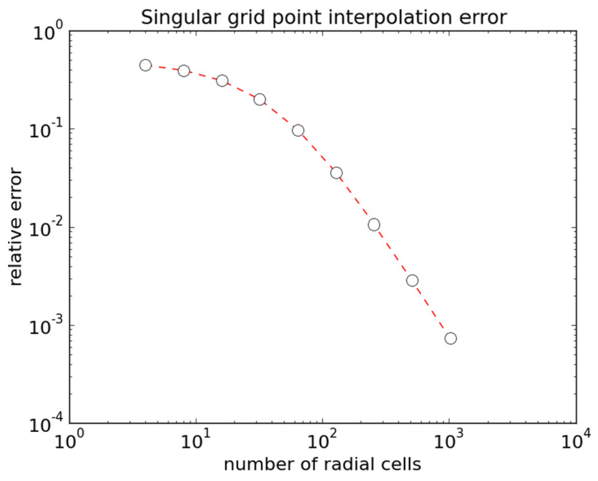


Fig. 5. Relative, L_2-norm error of the point interpolation method for the current filament problem. Conservative interpolation is nearly exact (data not shown).

This test demonstrates that conservative interpolation eliminates many of the problems associated with singular fields. Although field values can grow infinitely large in the $\rho \to 0$ limit, line integrals on the other hand are well behaved and do not suffer from infinities. Thus, much higher accuracy and numerical stability can be expected when working with forms instead of fields.

6.2. Line integral of an incompressible flow

An incompressible flow

$$v = d\psi \wedge dr \tag{28}$$

is prescribed on the surface of a sphere of radius r. Here, $\psi = \psi(\lambda, \theta)$ is the stream function expressed in longitude $0 \le \lambda \le 2\pi$ and latitude $-\pi/2 \le \theta \le \pi/2$ coordinate angles ($\theta = \pi/2$ at the north pole). The transformation from spherical (λ, θ, r) to Cartesian coordinates (x, y, z) obeys

$$x = r\cos\theta\cos\lambda$$
$$y = r\cos\theta\sin\lambda$$
$$z = r\sin\theta$$

with

$$\begin{aligned} dx &= \frac{xdr}{r} - z\cos\lambda d\theta - yd\lambda \\ dy &= \frac{ydr}{r} - z\sin\lambda d\theta + xd\lambda \\ dz &= \frac{zdr}{r} + r\cos\theta d\theta, \end{aligned} \tag{29}$$

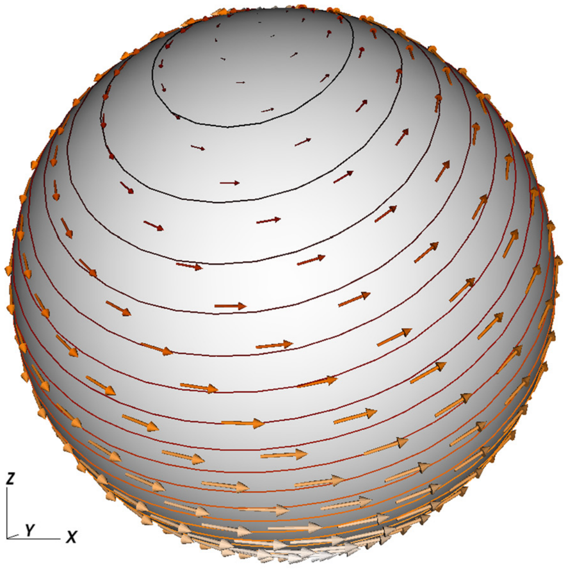
$$\begin{aligned} d\lambda &= -\frac{y}{\rho^2}dx + \frac{x}{\rho^2}dy \\ d\theta &= -\frac{xz}{\rho r^2}dx - \frac{yz}{\rho r^2}dy + \frac{\rho}{r^2}dz \\ dr &= \frac{x}{r}dx + \frac{y}{r}dy + \frac{z}{r}dz, \end{aligned} \tag{30}$$

$\rho^2 \equiv x^2 + y^2$, and $r^2 \equiv \rho^2 + z^2$.

Correspondence between (28) and a vector field is obtained by substituting d with ∇ and $\wedge$ with the cross product $\times$, yielding

$$\boldsymbol{v} = \nabla\psi \times \hat{\boldsymbol{r}} = \frac{1}{r}\left(\frac{\partial\psi}{\partial\theta}\hat{\boldsymbol{\lambda}} - \frac{1}{\cos\theta}\frac{\partial\psi}{\partial\lambda}\hat{\boldsymbol{\theta}}\right) \tag{31}$$

where $\nabla\lambda = \hat{\boldsymbol{\lambda}}/r\cos\theta$ and $\nabla\theta = \hat{\boldsymbol{\theta}}/r$.



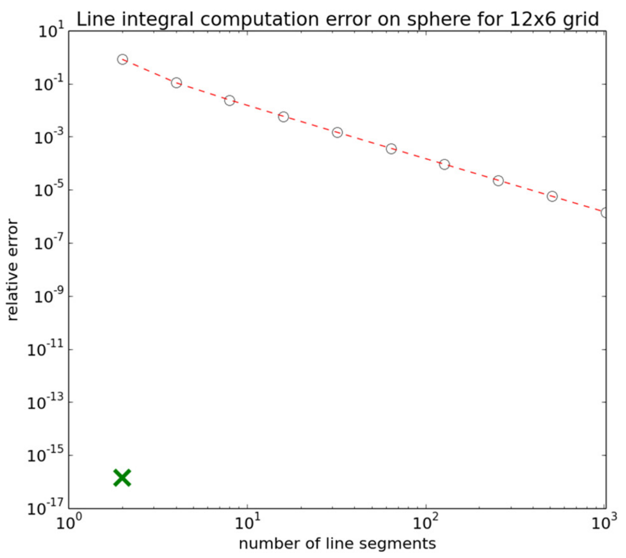


Fig. 6. Left: stream function $\psi = \theta - \cos\theta \sin\lambda$. Right: line integration error obtained by evaluating $d\psi/d\mu$ along the path $\lambda = 2\pi\mu$ and $\theta = \pi(\mu - \frac{1}{2})$ using the trapezoidal rule (dashed line). The cross shows the integration error of the conservative edge interpolation method.

From (28), it can readily be seen that the velocity field is divergence-free, i.e. $dv = 0$. By virtue of Stokes' theorem, $\int dv = \oint v = 0$, the flow across the surface of an arbitrarily bounded volume must therefore also vanish. Since $dr \neq 0$ on all spherical surfaces, one has $\oint v = \oint d\psi \wedge dr = (\oint d\psi) \wedge dr = 0$, so that $\oint d\psi$ must be zero along any closed contour and for any ψ.

The flow across an open line, $\int_a^b v = (\int_a^b d\psi) \wedge dr = (\psi(b) - \psi(a)) \wedge dr$ is just the difference of the stream function evaluated at the endpoints of the line. In other words, the flow integral can be computed to arbitrarily high precision if ψ is known analytically.

To test the accuracy of line integrals, we choose as path the line $\lambda = 2\pi\mu$ and $\theta = -\pi/2 + \pi\mu$ with μ the parametric coordinate varying from $\mu = 0$ at the starting point to $\mu = 1$ at the end point. The stream function is chosen to be $\psi = \theta - \cos\theta \sin\lambda$. It can be verified that the line integral should give π in this case. Two approaches are compared: (i) $\int d\psi$ is approximated by the sum $\frac{1}{N_s}\sum_{i=1}^{N_s}(d\psi/d\mu)$ with $d\psi/d\mu$ the exact stream function derivative evaluated at the center of each segment (trapezoidal quadrature rule) and (ii) by representing $d\psi$ as an edge field and integrating $d\psi$ over the segment using the method described in Section 4. Given that ψ is edge centered, the integration along a segmented line must amount to taking the difference between the end and start values.

Fig. 6 shows the error of the line integral obtained using both approaches. The non-conservative, trapezoidal error decreases proportionally to the square of the inverse of the number of segments. The conservative interpolation method on the other hand computes the integral to near machine accuracy using only two segments (no error was detected when using a single segment). Approximately 10 million segments would be required for the non-conservative method to achieve the same accuracy.

6.3. Flux integral of the electric displacement field

The electric displacement field D (a 2-form) satisfies Gauss' law

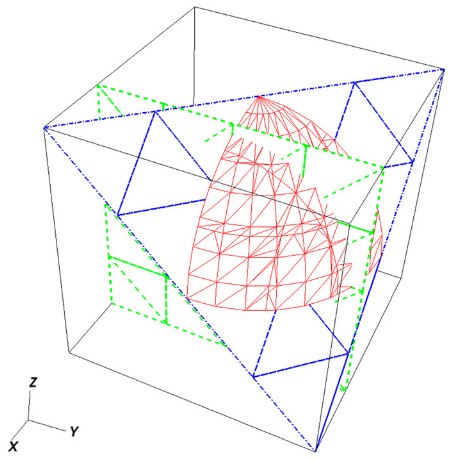
$$dD = \rho \tag{32}$$

where ρ is the charge density (a 3-form). For a single point charge located at position $(0, 0, 0)$, the displacement field can be written as

$$D = \frac{1}{4\pi r}(xdy \wedge dz + ydz \wedge dx + zdx \wedge dy) \tag{33}$$

in Cartesian coordinates (x, y, z), where $r^2 = x^2 + y^2 + z^2$. In spherical coordinates (λ, θ, r), the displacement field can be rewritten as

$$D = \frac{1}{4\pi}\cos\theta d\lambda \wedge d\theta. \tag{34}$$



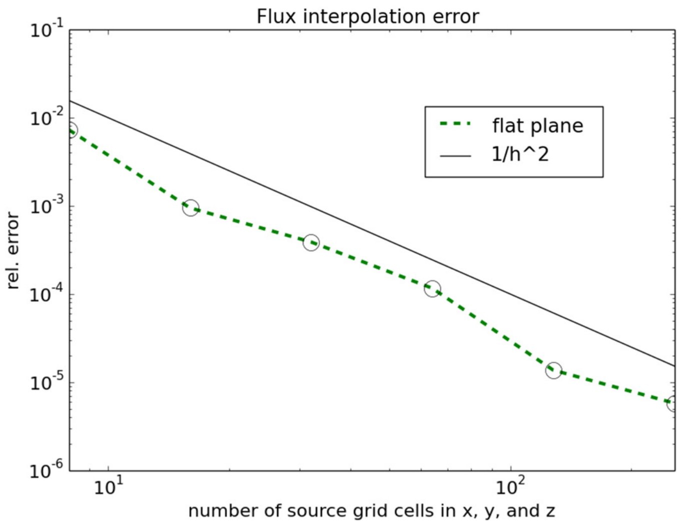


Fig. 7. Convergence of flux integrals as the 3D grid resolution increases for the three surfaces shown in the left pane (flat plane at $x = \pi/8$, slanted plane, and a spherical surface approximated with eight polar and azimuthal facets). A coarse $4 \times 4 \times 4$ grid was used to show the tessellation resulting from intersecting the surfaces with the uniform Cartesian grid. The convergence of the flux interpolation onto a flat surface follows the h^{-2} law for linear interpolation. The slanted and spherical surface interpolations have very small error, comparable to machine accuracy.

It is straightforward to check that the integral over the surface of the sphere centered around point $(0, 0, 0)$, $\int_{sphere} D = \frac{1}{4\pi} \int_0^{2\pi} d\lambda \int_{-\pi/2}^{\pi/2} d\theta \cos\theta = 1$.

For this numerical experiment, we choose a uniform rectilinear grid and set the field to be the integrated flux of (33) over each face of grid cells. The integrals of the field over cell faces can be obtained analytically, e.g.

$$\int_{x_i, y_j, z_k}^{x_i, y_{j+1}, z_{k+1}} x dy \wedge dz = \frac{1}{4\pi} \left[\arctan\left(\frac{y_{j+1} z_{k+1}}{x_i \sqrt{x_i^2 + y_{j+1}^2 + z_{k+1}^2}} \right) - \arctan\left(\frac{y_j z_{k+1}}{x_i \sqrt{x_i^2 + y_j^2 + z_{k+1}^2}} \right) - \arctan\left(\frac{y_{j+1} z_k}{x_i \sqrt{x_i^2 + y_{j+1}^2 + z_k^2}} \right) + \arctan\left(\frac{y_j z_k}{x_i \sqrt{x_i^2 + y_j^2 + z_k^2}} \right) \right] \tag{35}$$

where $y_{j+1} \equiv y_j + h$, $z_{k+1} = z_k + h$ and h is the cell size (the same in each direction). Other terms in (33) can be obtained by cyclic rotation of x, y, and z.

In Fig. 7 we show the conservative interpolation error as the resolution of the 3D grid increases for three target surfaces: a plane in the (y, z) direction, a slanted plane, and a partial spherical surface approximated by planar facets.

Given that the fluxes attached to each cell face are exact, there is no numerical interpolation error for fields projected onto surfaces that are aligned to and in contact with cell faces. All the projection errors are the result of interpolating fluxes between opposing faces. The error in this case is the same as that of a nodal 1D field, namely $\sim \frac{1}{8} f''_{max} h^2$, where f''_{max} is the maximum second derivative of the flux in the direction perpendicular to the faces.

The dependence of the interpolation error ($\sim h^2$) on the grid resolution can be clearly observed for the case where the target surface is the plane $x = \pi/8$ (dashed line). The small oscillations in the convergence are due to fluctuations in the distance separating the plane from the nearest cell faces, the choice of location $\pi/8$ ensuring that the plane never touches these cell faces.

For the slanted target surface, we have $x = 1 - 2v$, $y = 2(u + v) - 1$, and $z = 1 - 2u$, $0 \le u, v \le 1$, with the exact flux integral amounting to $\frac{1}{4}$. This flux value can be determined by observing that there are four such slanted planes, which, when properly arranged, completely contain the source point $(0, 0, 0)$. It is a property of conservative interpolation to

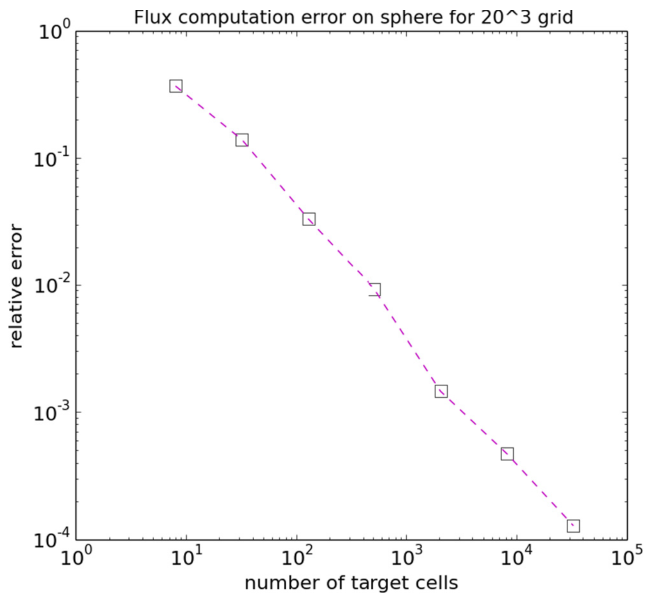


Fig. 8. Flux integration error obtained using a non-conservative interpolation method as the *target* grid resolution increases. The total flux is computed by evaluating the exact field at face centers. Conservative interpolation on the other hand yields errors $< O(10^{-13})$.

preserve symmetry. Since the total flux is one by virtue of symmetry, each slanted plane must contribute to $\frac{1}{4}$ (we observed an error of $< O(10^{-14})$). The same argument holds for the partial sphere approximated by 8×8 facets and in this case we also find near exact interpolation.

In Fig. 8 we show the accuracy of the flux computation assuming an "exact" nodal representation of the field with conservative interpolation as the number of *target* cells increases, keeping the original $20 \times 20 \times 20$ grid resolution constant. Conservative interpolation is nearly exact in this case.

6.4. Fluxes with partial cells: Poiseuille flow in a pipe

Next we consider a uniform, laminar flow inside a pipe of radius one. The velocity 2-form

$$v = \frac{2}{\pi}\left(1 - x^2 - y^2\right) dx \wedge dy, \tag{36}$$

is purely in the direction of the flow (z) with

$$\int_0^1 v = 1. \tag{37}$$

The geometry of the pipe is embedded in a uniform grid with cell size $h_x \times h_y \times h_z$. Velocity data are attached to cell faces. For cells that are fully contained inside the pipe ($r \le 1$), the z-flow in cell (i, j, k) is

$$\int_{ijk} v = \frac{2}{\pi}\left[h_x h_y - \frac{h_y}{3}(x_{i+1}^3 - x_i^3) - \frac{h_x}{3}(y_{j+1}^3 - y_j^3)\right]. \tag{38}$$

Care must be taken for cells that are intersected by the boundary surface $r = 1$. Fig. 9 shows that such partially valid cells consist of subcell bounded by a circle, and one or two rectangular subcells. The flow in the rectangular subcells can be computed using (38). To set the flow across a subcell bounded by a circle, we used the formula

$$\int_{ijk} v = \frac{2}{\pi}\left[-\frac{1}{3}\left(x_i\sqrt{1 - x_i^2} + (x_i/4 - x_i^3/2)\sqrt{1 - x_i^2}\right.\right.$$
$$\left. - 3x_i y_j + x_i^3 y_j + x_i y_j^3 + \frac{3}{4}\arcsin(x_i)\right)$$

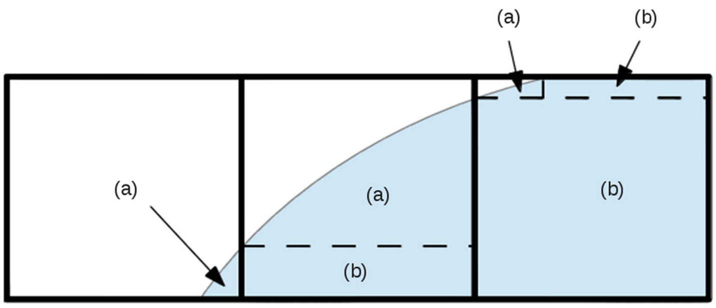


Fig. 9. Examples of partially valid cells (shaded areas). The flow integral can be computed analytically in the Poiseuille flow test case by splitting the shaded area into a subcell bounded by an arc circle (a) and one or two rectangular subcells (b).

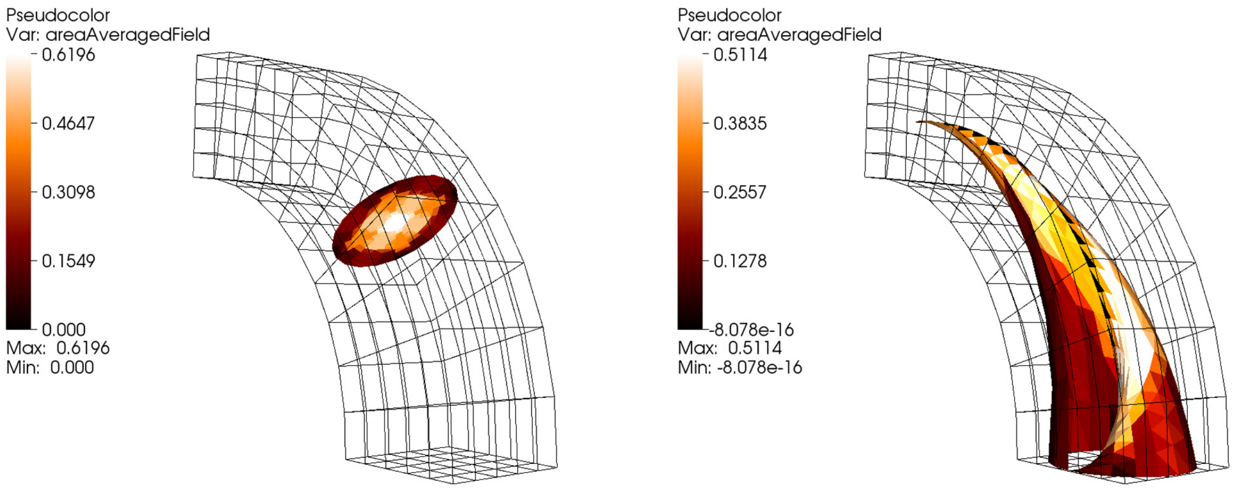


Fig. 10. Face-centered flow across two distinct surfaces, a disk and a disk with a helical perturbation. Both target surfaces share the same projection onto the plane perpendicular to the flow. The total integrated flow is found to be the same in both cases to near machine precision despite vastly different target surfaces.

$$+\frac{1}{3}\Bigg(x_{i+1}\sqrt{1-x_{i+1}^2}+(x_{i+1}/4-x_{i+1}^3/2)\sqrt{1-x_{i+1}^2}$$
$$-3x_{i+1}y_j+x_{i+1}^3y_j+x_{i+1}y_j^3+\frac{3}{4}\arcsin(x_{i+1})\Bigg)\Bigg]. \tag{39}$$

The flow across cut cells intersected on the south side can be obtained from this expression by substituting x_i, x_{i+1} and y_j for x_{i+1}, x_i, and $-y_j$. Expressions for cells intersected on the east/west side can be obtained by similar coordinate transformations. Note that (39) already incorporates the effect of partial faces, no masking array was used to turn off the field on the outside of the pipe.

In Fig. 10 we show the Poiseuille flow solution in a curvilinear grid geometry. The left panel shows the velocity flow across a disk approximated with 20 poloidal sectors. The coarse discretization is noticeable on the projected disk. The total flux across the disk (0.9491) is somewhat lower than the total flux across the pipe due the finite resolution of both the 3D and the projected surface resolutions. On the right, the flow is projected onto a surface with the same cross section but with a helical distortion added in the direction of the flow. Despite the differences in target surface geometries, the two fluxes were found to be the same to within a small error $O(10^{-15})$, thus confirming conservation.

6.5. Fluxes in four dimensions

The formalism for interpolating differential forms can be applied to spaces with more than three dimensions. Higher dimensional spaces arise in many settings, e.g. when solving Maxwell's equations in space–time. It is well known that Faraday's law takes a particularly elegant and simple form in space–time [37]:

$$dF=0 \tag{40}$$

where $F\equiv E\wedge dt+B$ is a 2-form that combines the electric 1-form E and the magnetic 2-form B.

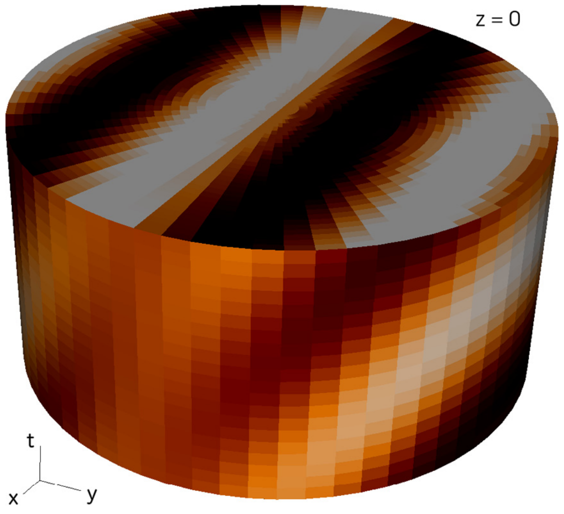


Fig. 11. Projection of the space–time electromagnetic 2-form F onto a cylinder with unit length and radius for a plane wave solution. The vertical axis represents time and the horizontal axes x and y, respectively. Each cell is colored according to the average value of the 2-flux in that cell (white for 1, black for -1). The top and bottom of the cylinder represent the magnetic flux while the side of the cylinder captures the contribution from $E \wedge dt$.

The accuracy of an electromagnetic solution can be tested by computing the flux integral of F over an enclosed surface,

$$\int_{\Omega} dF = \oint_{\partial\Omega} F = 0 \tag{41}$$

and checking that it is zero. For this test case, we applied a plane wave solution

$$E = \cos(k_y y + k_z z - \omega t)dx \tag{42}$$

and

$$B = -\frac{k_y}{\omega}\cos(k_y y + k_z z - \omega t)dx \wedge dy + \frac{k_z}{\omega}\cos(k_y y + k_z z - \omega t)dz \wedge dx \tag{43}$$

to the cell faces of a 4D structured grid of size $32 \times 32 \times 1 \times 32$. The projection of F onto a cylinder in (x, y, t) ($k_y = 2\pi/0.9$, $k_z = 2\pi/1.5$, $\omega = 2\pi/1.2$) is shown in Fig. 11 for $z = 0$. Conservation is achieved when the sum of the fluxes across the top, side, and bottom of the cylinder is zero, we found this to be the case within roundoff errors $O(10^{-15})$.

7. Summary

We presented a method for applying linear interpolation to staggered fields. Particular emphasis was put on edge and face staggerings although the presented formalism also covers point and cell-centered conservative interpolation as special cases.

To our knowledge, it is the first time that conservative interpolation is presented for n dimensional, structured meshes. As a proof of principle, we have written conservative interpolation software that operates in any number of dimensions. We believe that this could open up new post-processing methodologies — particularly for systems like Maxwell's equations, which can be written in conservative form. We showed in particular how the electromagnetic 2-form flux integrated over a space–time surface should give zero, thus providing a measure of the accuracy of a numerical solution. Using simple examples based on analytic fields, we demonstrated how the presented interpolation method conserves line integrals (in the case of edge-centered data) and area integrals (for face centered data). These conservation properties are closely related to other fundamental properties. For instance, line integrals of vector fields deriving from a potential were shown to depend only on the end values along the integration path. Likewise, integrals of curl-free fields only depend on the circulation of the field over the boundary of the surface.

Most cell-centered conservative interpolation schemes require users to provide a source field and both source and target cell volumes. In place of field densities, we recommend working with cell-volume, face-area, and edge-length integrated values. This has several advantages. First, integrals automatically take into account fractional and vanishing volume, area, and length elements so that there is no need to introduce masking arrays. Second, the integrals remain well defined as one approaches a pole singularity. Third, inconsistencies regarding how and to what accuracy (volume, area, length) elements are computed are avoided since the effect of the volumes, areas, and lengths is included in the integral representation. This is a particular important issue in geophysics because of the curvature of the earth; using slightly different values for the earth's radius between the source and target elements can lead to a lack of conservation when working with fields and cell areas separately.

It is our hope that the ideas presented in this paper will be applied to the regridding of staggered fields arising in climate modeling and other fields where conservation of mass flow, energy, and other quantities are paramount. Given that some numerical schemes, e.g. the finite volume method, put great emphasis on flux conservation, we argue that the same level of attention should be paid to postprocessing and visualization tasks. As a first step, we have extended the open-source visualization tool VisIt [38] by adding support for edge and face centered data in the VizSchema plugin [39]. The result is a more accurate streamline visualization, which removes many visualization artifacts observed when treating face centered field as either nodal or cell centered.

With our focus on structured fields, we only scraped the surface of the possibilities of applying conservative edge and face interpolation. The same ideas carry over to unstructured grids with the main difference that the interpolation basis functions are different [28]. Naturally, we expect the computation of grid intersections to be numerically more demanding due to the lack of an index space for unstructured grids. This can be remedied by using an octree algorithm [7]. Nevertheless, these challenges *cannot* be regarded as being inherent to edge and face interpolation as they arise for any type of unstructured field interpolation. Moreover, this numerical cost can be amortized for time varying data if the grid is static by reusing the interpolation weights.

Beyond the presented linear interpolation method for edge and face data, there has been some work [40] aimed at deriving hierarchical interpolation bases with varying polynomial orders, which are conservative. This promises a way for more efficient interpolation methods, capable of capturing small details where needed while at the same time reducing the number of degrees of freedom in regions that exhibit small variations.

Acknowledgement

This work was supported by US Department of Energy – Advanced Scientific Computing Research Grant SBIR II DE-SC0006241.

Appendix A. Interpolation of staggered fields on 3D cuboids

Interpolation formulas are provided in terms of local, parametric coordinates $\boldsymbol{\xi}$ in 3D. The volume of the cell is $J = \frac{\partial \mathbf{x}}{\partial \xi_1} \times \frac{\partial \mathbf{x}}{\partial \xi_2} \cdot \frac{\partial \mathbf{x}}{\partial \xi_3}$ and the reader should refer to Fig. 2 for the location of each basis function within the cuboid. Expressions for the edge and face basis function are given both in terms of contravariant $\nabla \xi_i$ and covariant $\frac{\partial \mathbf{x}}{\partial \xi_i}$ basis vectors, the latter can be approximated with finite differences.

We start with the well known basis functions for nodal fields, the reader is encouraged to compare these expressions with those for edge and face centered fields.

A.1. Interpolation of a nodal field

Interpolation formula:

$$\bar{f} = \sum_{\sigma} f_{\sigma} w_{\sigma} \tag{A.1}$$

Field values (field at cell vertices):

$$f_{\sigma} = f(\boldsymbol{\xi}_{\sigma}) \tag{A.2}$$

Interpolation weights for target point $\hat{\boldsymbol{\xi}}$:

$$w_{\sigma} = \phi_{\sigma}(\hat{\boldsymbol{\xi}}) \tag{A.3}$$

Basis functions:

$$\phi_{000} = (1-\xi_1)(1-\xi_2)(1-\xi_3)$$

$$\phi_{001} = (1-\xi_1)(1-\xi_2)\xi_3$$

$$\phi_{010} = (1-\xi_1)\xi_2(1-\xi_3)$$

$$\phi_{011} = (1-\xi_1)\xi_2\xi_3$$

$$\phi_{100} = \xi_1(1-\xi_2)(1-\xi_3)$$

$$\phi_{101} = \xi_1(1-\xi_2)\xi_3$$

$$\phi_{110} = \xi_1\xi_2(1-\xi_3)$$

$$\phi_{111} = \xi_1\xi_2\xi_3$$

A.2. Interpolation of an edge centered field

Interpolation formula:

$$\bar{f} = \sum_{\sigma} f_{\sigma} w_{\sigma} \tag{A.4}$$

Field values (line integral of field along cell edges):

$$f_{\sigma} = \int_{\sigma} \mathbf{f} \cdot d\mathbf{x} \tag{A.5}$$

Interpolation weights for target line $\hat{\boldsymbol{\xi}}_0 + \lambda(\hat{\boldsymbol{\xi}}_1 - \hat{\boldsymbol{\xi}}_0)$:

$$w_{\sigma} = \int_0^1 d\lambda \phi_{\sigma} \left(\hat{\boldsymbol{\xi}}_0 + \lambda(\hat{\boldsymbol{\xi}}_1 - \hat{\boldsymbol{\xi}}_0) \right) \tag{A.6}$$

Basis functions:

$$\begin{aligned}
\boldsymbol{\phi}^{(1)}_{*00} &= (1-\xi_2)(1-\xi_3)\nabla\xi_1 = J^{-1}(1-\xi_2)(1-\xi_3)\frac{\partial\mathbf{x}}{\partial\xi_2} \times \frac{\partial\mathbf{x}}{\partial\xi_3} \\
\boldsymbol{\phi}^{(1)}_{*01} &= (1-\xi_2)\xi_3\nabla\xi_1 = J^{-1}(1-\xi_2)\xi_3\frac{\partial\mathbf{x}}{\partial\xi_2} \times \frac{\partial\mathbf{x}}{\partial\xi_3} \\
\boldsymbol{\phi}^{(1)}_{*10} &= \xi_2(1-\xi_3)\nabla\xi_1 = J^{-1}\xi_2(1-\xi_3)\frac{\partial\mathbf{x}}{\partial\xi_2} \times \frac{\partial\mathbf{x}}{\partial\xi_3} \\
\boldsymbol{\phi}^{(1)}_{*11} &= \xi_2\xi_3\nabla\xi_1 = J^{-1}\xi_2\xi_3\frac{\partial\mathbf{x}}{\partial\xi_2} \times \frac{\partial\mathbf{x}}{\partial\xi_3} \\
\boldsymbol{\phi}^{(1)}_{0*0} &= (1-\xi_1)(1-\xi_3)\nabla\xi_2 = J^{-1}(1-\xi_1)(1-\xi_3)\frac{\partial\mathbf{x}}{\partial\xi_3} \times \frac{\partial\mathbf{x}}{\partial\xi_1} \\
\boldsymbol{\phi}^{(1)}_{0*1} &= (1-\xi_1)\xi_3\nabla\xi_2 = J^{-1}(1-\xi_1)\xi_3\frac{\partial\mathbf{x}}{\partial\xi_3} \times \frac{\partial\mathbf{x}}{\partial\xi_1} \\
\boldsymbol{\phi}^{(1)}_{1*0} &= \xi_1(1-\xi_3)\nabla\xi_2 = J^{-1}\xi_1(1-\xi_3)\frac{\partial\mathbf{x}}{\partial\xi_3} \times \frac{\partial\mathbf{x}}{\partial\xi_1} \\
\boldsymbol{\phi}^{(1)}_{1*1} &= \xi_1\xi_3\nabla\xi_2 = J^{-1}\xi_1\xi_3\frac{\partial\mathbf{x}}{\partial\xi_3} \times \frac{\partial\mathbf{x}}{\partial\xi_1} \\
\boldsymbol{\phi}^{(1)}_{00*} &= (1-\xi_1)(1-\xi_2)\nabla\xi_3 = J^{-1}(1-\xi_1)(1-\xi_2)\frac{\partial\mathbf{x}}{\partial\xi_1} \times \frac{\partial\mathbf{x}}{\partial\xi_2} \\
\boldsymbol{\phi}^{(1)}_{01*} &= (1-\xi_1)\xi_2\nabla\xi_3 = J^{-1}(1-\xi_1)\xi_2\frac{\partial\mathbf{x}}{\partial\xi_1} \times \frac{\partial\mathbf{x}}{\partial\xi_2} \\
\boldsymbol{\phi}^{(1)}_{10*} &= \xi_1(1-\xi_2)\nabla\xi_3 = J^{-1}\xi_1(1-\xi_2)\frac{\partial\mathbf{x}}{\partial\xi_1} \times \frac{\partial\mathbf{x}}{\partial\xi_2} \\
\boldsymbol{\phi}^{(1)}_{11*} &= \xi_1\xi_2\nabla\xi_3 = J^{-1}\xi_1\xi_2\frac{\partial\mathbf{x}}{\partial\xi_1} \times \frac{\partial\mathbf{x}}{\partial\xi_2}
\end{aligned}$$

A.3. Interpolation of a face centered field

Interpolation formula:

$$\bar{f} = \sum_{\sigma} f_{\sigma} w_{\sigma} \tag{A.7}$$

Field values (area integrals of field on cell faces):

$$f_{\sigma} = \int_{\sigma} \mathbf{f} \cdot d\mathbf{S} \tag{A.8}$$

Interpolation weights for target triangle $\boldsymbol{\xi}_0 + \lambda_1(\hat{\boldsymbol{\xi}}_1 - \hat{\boldsymbol{\xi}}_0) + \lambda_2(\hat{\boldsymbol{\xi}}_2 - \hat{\boldsymbol{\xi}}_0)$:

$$w_{\sigma} = \int_0^1 d\lambda_1 \int_0^{1-\lambda_1} d\lambda_2 \boldsymbol{\phi}_{\sigma} \left(\hat{\boldsymbol{\xi}}_0 + \lambda_1(\hat{\boldsymbol{\xi}}_1 - \hat{\boldsymbol{\xi}}_0) + \lambda_2(\hat{\boldsymbol{\xi}}_2 - \hat{\boldsymbol{\xi}}_0) \right) \tag{A.9}$$

Basis functions:

$$\boldsymbol{\phi}^{(2)}_{**0} = (1-\xi_3)\nabla\xi_1 \times \nabla\xi_2 = J^{-1}(1-\xi_3)\frac{\partial \mathbf{x}}{\partial \xi_3}$$

$$\boldsymbol{\phi}^{(2)}_{**1} = \xi_3\nabla\xi_1 \times \nabla\xi_2 = J^{-1}\xi_3\frac{\partial \mathbf{x}}{\partial \xi_3}$$

$$\boldsymbol{\phi}^{(2)}_{*0*} = (1-\xi_2)\nabla\xi_1 \times \nabla\xi_3 = -J^{-1}(1-\xi_2)\frac{\partial \mathbf{x}}{\partial \xi_2}$$

$$\boldsymbol{\phi}^{(2)}_{*1*} = \xi_2\nabla\xi_1 \times \nabla\xi_3 = -J^{-1}\xi_2\frac{\partial \mathbf{x}}{\partial \xi_2}$$

$$\boldsymbol{\phi}^{(2)}_{0**} = (1-\xi_1)\nabla\xi_2 \times \nabla\xi_3 = J^{-1}(1-\xi_1)\frac{\partial \mathbf{x}}{\partial \xi_1}$$

$$\boldsymbol{\phi}^{(2)}_{1**} = \xi_1\nabla\xi_2 \times \nabla\xi_3 = J^{-1}\xi_1\frac{\partial \mathbf{x}}{\partial \xi_1}$$

References

[1] P.K. Kundu, I.M. Cohen, Fluid Mechanics, Elsevier Academic Press, 2004, pp. 52–54.

[2] L.S. Avila, S. Barré, B. Geveci, A. Henderson, W.A. Hoffmann, B. King, C.C. Law, K.M. Martin, W.J. Schroeder, The VTK User's Guide, Kitware, Inc., 2003, pp. 79–81.

[3] A. Pletzer, B. Jamroz, R. Crockett, S. Sides, Compact cell-centered discretization stencils at fine-coarse block structured grid interfaces, J. Comput. Phys. 260 (2014) 25–36.

[4] E. Santos, J. Poco, Y. Wei, S. Liu, B. Cook, D. Williams, C. Silva, UV-CDAT: analyzing climate datasets from a user's perspective, Comput. Sci. Eng. 15 (1) (2013) 94–103.

[5] S. Hankin, D.E. Harrison, J. Osborne, J. Davidson, K. O'Brien, A strategy and a tool, Ferret, for closely integrated visualization and analysis, J. Vis. Comput. Animat. 7 (3) (1996) 149–157.

[6] The NCAR Command Language (version 6.3.0) [Software], Tech. Rep., UCAR/NCAR/CISL/TDD, Boulder, CO, 2015.

[7] C. Hill, C. DeLuca, V. Balaji, M. Suarez, A. da Silva, The architecture of the Earth system modeling framework, Comput. Sci. Eng. 6 (Jan. 2004) 18–28.

[8] K. Taylor, EZGET: a library for Fortran subrountines to facilitate data retrieval, PCMDI Report, 1996.

[9] MATLAB, version 7.10.0 (R2010a), The MathWorks Inc., Natick, MA, 2015.

[10] K.P. Bowman, An Introduction to Programming with IDL: Interactive Data Language, Academic Press, Oct. 2005.

[11] I. Wolfram Research, Mathematica, version 10.1 ed., Wolfram Research, Inc., 2015.

[12] D. Williams, T. Bremer, C. Doutriaux, J. Patchett, S. Williams, G. Shipman, R. Miller, D. Pugmire, B. Smith, C. Steed, E. Bethel, H. Childs, H. Krishnan, P. Prabhat, M. Wehner, C. Silva, E. Santos, D. Koop, T. Ellqvist, J. Poco, B. Geveci, A. Chaudhary, A. Bauer, A. Pletzer, D. Kindig, G. Potter, T. Maxwell, Ultrascale visualization of climate data, Computer 46 (September 2013) 68–76.

[13] T.J. Phillips, K. Achutarao, D. Bader, C. Covey, C.M. Doutriaux, M. Fiorino, P.J. Gleckler, K.R. Sperber, K.E. Taylor, Coupled climate model appraisal: a benchmark for future studies, Eos 87 (19) (2006) 185.

[14] J.D. Ramshaw, Conservative rezoning algorithm for generalized two-dimensional meshes, J. Comput. Phys. 59 (2) (1985) 193–199.

[15] P.W. Jones, First- and second-order conservative remapping schemes for grids in spherical coordinates, Mon. Weather Rev. 127 (Sept. 1999) 2204–2210.

[16] Climate and Earth's Energy Budget, http://earthobservatory.nasa.gov/Features/EnergyBalance/.

[17] S.J. Lin, R.B. Rood, Multidimensional flux-form semi-Lagrangian transport schemes, Mon. Weather Rev. 124 (1996) 2046–2070.

[18] P. Raviart, J. Thomas, A mixed finite element method for 2-nd order elliptic problems, in: Mathematical Aspects of Finite Element Methods, vol. 606, 1977, pp. 292–315.

[19] J.C. Nédélec, Mixed elements in R^3, Numer. Math. 35 (1980) 315–340.

[20] J.S.V. Welij, Basis functions matching tangential components on element edges, IEEE Trans. Magn. 21 (1985) 537–542.

[21] K.S. Yee, Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media, IEEE Trans. Antennas Propag. (1966) 302–307.

[22] F.H. Harlow, J.E. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, Phys. Fluids (1958–1988) 8 (12) (1965) 2182–2189.

[23] X. Llobet, K. Appert, A. Bondeson, J. Vaclavik, On spectral pollution, Comput. Phys. Commun. 59 (2) (1990) 199–216.

[24] J.H. Ferziger, M. Perić, Computational Methods for Fluid Dynamic, Springer-Verlag, Berlin, Heidelberg, 2002, pp. 164–167.

[25] J.B. Perot, C.J. Zusi, Differential forms for scientists and engineers, J. Comput. Phys. 257 (Part B) (2014) 1373–1393, Physics-compatible numerical methods.

[26] R. Hiemstra, D. Toshniwal, R. Huijsmans, M. Gerritsma, High order geometric methods with exact conservation properties, J. Comput. Phys. 257 (Part B) (2014) 1444–1471, Physics-compatible numerical methods.

[27] C. Cotter, J. Thuburn, A finite element exterior calculus framework for the rotating shallow-water equations, J. Comput. Phys. 257 (Part B) (2014) 1506–1526, Physics-compatible numerical methods.

[28] H. Whitney, Geometric Integration Theory, Dover Books on Mathematics, 1957.

[29] R. Hiptmair, Finite elements in computational electromagnetism, Acta Numer. 11 (2002) 237–339.

[30] A. Bossavit, Discretization of Electromagnetic Problems: The "Generalized Finite Differences" Approach, Elsevier, Amsterdam, 2005, pp. 105–197.

[31] R. Garimella, M. Kucharik, M. Shashkov, An efficient linearity and bound preserving conservative interpolation (remapping) on polyhedral meshes, Comput. Fluids 36 (2) (2007) 224–237.

[32] F. Alauzet, M. Mehrenberger, P1-conservative solution interpolation on unstructured triangular meshes, Int. J. Numer. Methods Eng. 84 (13) (2010) 1552–1588.

[33] D.N. Arnold, D. Boffi, F. Bonizzoni, Finite element differential forms on curvilinear cubic meshes and their approximation properties, Numer. Math. 129 (1) (2015) 1–20.
[34] D.N. Arnold, D. Boffi, R.S. Falk, Approximation by quadrilateral finite elements, Math. Comput. 71 (239) (2002) 909–922.
[35] G. Matthies, Mapped finite elements on hexahedra: necessary and sufficient conditions for optimal interpolation errors, Numer. Algorithms 27 (4) (2001) 317–327.
[36] C.B. Barber, D.P. Dobkin, H. Huhdanpaa, The quickhull algorithm for convex hulls, ACM Trans. Math. Softw. 22 (Dec. 1996) 469–483.
[37] K. Warnick, P. Russer, Two, three and four-dimensional electromagnetics using differential forms, Turk. J. Electr. Eng. Comput. Sci. 14 (1) (2006) 153–172.
[38] A contract-based system for large data visualization, http://www.idav.ucdavis.edu/publications/print_pub?pub_id=890/.
[39] S. Shasharina, D. Alexander, J. Cary, M. Durant, S. Kruger, et al., VizSchema – a unified visualization of computational accelerator physics data, in: Conf. Proc., vol. C100523, 2010, TUPEC069.
[40] R. Hiptmair, Higher order Whitney forms, in: Geometrical Methods in Computational Electromagnetics, 2000, pp. 271–299.