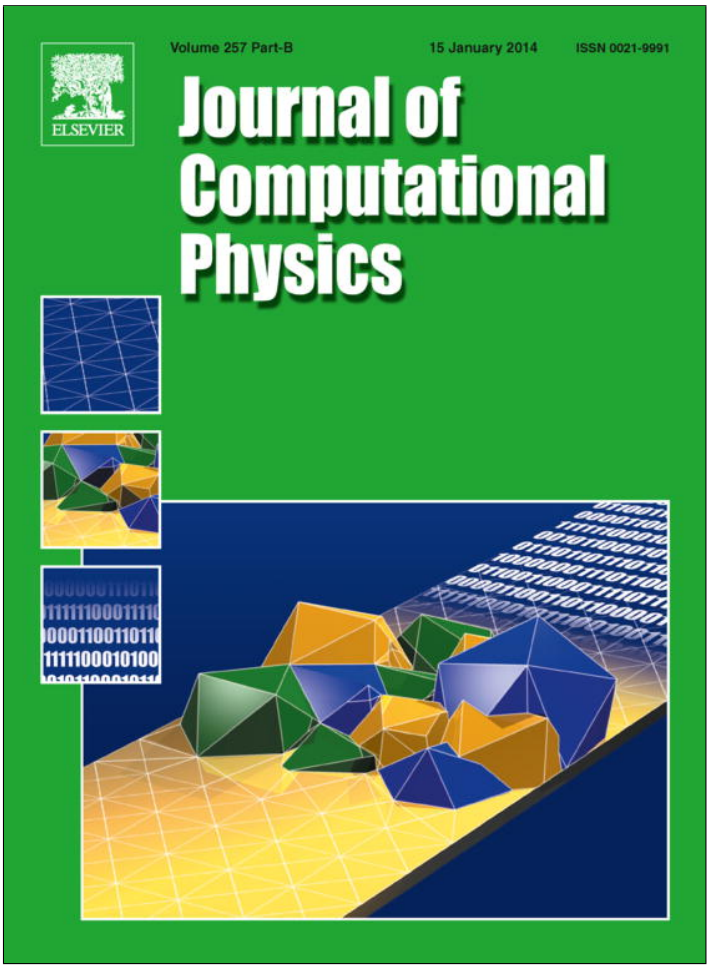

Journal of Computational Physics 257 (2014) 1373–1393

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Differential forms for scientists and engineers

J. Blair Perot *, Christopher J. Zusi

Theoretical and Computational Fluid Dynamics Laboratory, University of Massachusetts, Amherst, MA 01003, USA

ARTICLE INFO

Article history:
Received 13 August 2012
Received in revised form 17 May 2013
Accepted 4 August 2013
Available online 14 August 2013

Keywords:
Mimetic
Differential forms
Exterior calculus
Wedge product
Lie derivative
Numerical methods
Algebraic topology
Mimetic

ABSTRACT

This paper is a review of a number of mathematical concepts from differential geometry and exterior calculus that are finding increasing application in the numerical solution of partial differential equations. The objective of the paper is to introduce the scientist/engineer to some of these ideas via a number of concrete examples in 2, 3, and 4 dimensions. The goal is not to explain these ideas with mathematical precision but to present concrete examples and enable a physical intuition of these concepts for those who are not mathematicians. The objective of this paper is to provide enough context so that scientist/engineers can interpret, implement, and understand other works which use these elegant mathematical concepts.

1. Introduction

Recently, numerical methods for the solution of partial differential equations (PDEs) have begun to focus on creating methods that are more precise at capturing the critical physical and mathematical properties of the PDE. One interesting observation is that these types of numerical methods (referred to as mimetic, compatible, symmetry preserving) are often described and analyzed mathematically using the language of differential forms. Differential forms are an alternative, and potentially very powerful, notation for describing physical systems. However, differential forms are not a topic that is typically taught to scientists and engineers whereas multivariable calculus is a topic that is familiar. This work will attempt to show the connection between differential forms and multivariable calculus from an applied perspective.

Differential forms provide an extremely clean framework from which to analyze PDEs. They have the advantage of working in arbitrarily high dimensions and being coordinate system independent. For the scientist or engineer high dimensionality is not the appeal of differential forms. Scientists and engineers are most interested in solving PDEs such as the Navier–Stokes equations, Maxwell's equations, the Kortveg de Vries equation, Schrödinger equation, etc, where three or four dimensions are all that are required to describe the system. The driving force for using differential forms is the clarity that differential forms (in particular, discrete or mesh extensions of differential forms) and their associated notation can impart on the discretization process when describing the numerical solution of PDEs (particularly mimetic or compatible discretizations). Exterior calculus and differential geometry are a formalism that emphasizes the important *relationships* between differential operators and their operands. This makes it easier to develop numerical methods that respect those relationships and this, in turn, makes it possible to capture the essential physics and mathematics of the PDE.

* Corresponding author.
E-mail address: perot@ecs.umass.edu (J. Blair Perot).

0021-9991/$ – see front matter
http://dx.doi.org/10.1016/j.jcp.2013.08.007

The classic numerical analysis concepts of order of accuracy, stability, consistency and convergence, are not related to the dimension of the system being studied or directly to the physics underlying the PDE. But in order to properly account for the physics of PDE systems, the dimensionality (or topology) of the PDE system is actually important. For example, incompressibility never appears in 1D PDE systems. 2D vorticity and all its moments are conserved by the incompressible inviscid 2D Navier–Stokes equations, but in 3D only the vorticity (or more properly the circulation) is conserved. In 2D, the curl differential operation does not add new information, as it essentially mimics the divergence operation (curl is the divergence of the 90 degrees rotated vector field). However, in 3D the curl operation is unique, and in 4D an additional unique differential operation arises.

One of the import mathematical relationships that differential forms enable numerical methods to respect discretely are the orthogonality relations; $\nabla \times \nabla s = 0$ for any scalar s and $\nabla \cdot \nabla \times \mathbf{v} = 0$ for any vector $\mathbf{v}$. Enforcing these properties on the discrete operators was indirectly the impetus for the development of staggered mesh methods (Harlow and Welch, 1965) [1] and was directly the motivation for the 'face' elements of Raviart and Thomas (1977) [2] and their 'edge' counterparts (Nedelec 1980) [3]. As a result of these operator orthogonality properties, these types of numerical methods have remarkable physical attributes (see reference [4] for examples). These methods were not developed with differential forms, but subsequent analysis of these methods using differential forms has helped to understand why these methods are so effective at capturing the physics of PDE systems [5–7].

The dimensional generality of differential forms is both their greatest advantage (analysis simplification) as well as potentially their greatest weakness due to the obfuscation that can result for a newcomer from the high level of abstraction. A similar loss of clarity happens with Newton's equations of motion, $\mathbf{F} = m\mathbf{a}$. This equation describes all of classical physics (simplification) but is also so abstract that it is essentially useless on a practical level (one has to specify the forces to actually solve anything). In order to numerically solve a PDE with a numerical method that is described using differential forms it is helpful to be able to translate differential forms and exterior calculus, back to multivariable calculus. The primary purpose of this paper is to enable scientists/engineers to perform this sort of translation. The goal is to 'spell everything out'. This is contrary to the real purpose of differential forms, which are designed to make everything simple and clear. But for the newcomer, the prosaic translation in this paper may be a useful way to be introduced to some fairly new concepts. As a result of this intent, to reach scientists/engineers, this paper will take some (mathematically improper) liberties with the formal mathematical notation and also with the explanations so as to keep things as simple as possible. The paper will also focus on differential forms in the context of PDEs and the numerical solution of PDEs.

This paper is not an exhaustive discussion of differential forms. It is not even a proper introduction to differential forms. It focuses on those concepts most necessary to understand how differential forms can be applied to the discretization of PDEs. The discussion will progress through various mathematical operations with one operation per section. For each mathematical operation there is one (simple) expression using the notation of differential forms (which is dimension independent) and many translations into differential calculus (at least one translation for each dimension). This paper will consider 3D to be the natural and basic dimension and will use terminology relevant to 3D. Where possible, 3D pictures will be used. But 2D pictures are used when they are less confusing than 3D. 2D and 4D will be discussed as the somewhat unusual cases. Almost everything translates trivially to 1D and so this case is not discussed explicitly. The 4D cases show the process necessary to go to even higher dimensions if that might ever be desired.

This paper does not directly present a numerical method based on discrete forms, but an example of how differential forms are used to develop mimetic numerical methods is provided in Section 10. More detailed examples are referenced extensively throughout the text, and are also found elsewhere in this special journal issue. The goal of this paper is to make those texts available to a broad audience.

2. Forms

In the first sentence of a classic text on the subject entitled "Differential Forms" [8] Flanders describes differential forms as "the things which occur under integral signs." His example is a line integral $\int_{n_1}^{n_2} \mathbf{v} \cdot d\mathbf{l}$ which leads to the 1-form $v^1 = \mathbf{v} \cdot d\mathbf{l}$. The superscript is not a power, but an indicator of the type of the form. There are also 2-forms $w^2 = \mathbf{w} \cdot d\mathbf{A}$ which appear in surface integrals, and 3-forms $a^3 = a\,dV$ which appear in volume integrals and 0-forms b^0 which are regular scalar functions (a 0-integral is trivial). In general, a differential form does not require a differential (like $d\mathbf{l}$) to define, nor a Cartesian vector representation (also used above) [9]. But scientists and engineers tend to prefer concrete examples to mathematically precise definitions so the formal definition of a differential form is given in Appendix A of this paper. This paper will also assume a Cartesian representation for vectors (a simple list of 3 numbers in 3D). Vector representations in general coordinate systems are a complexity that is not necessary for this introduction.

Discrete forms are not a universally recognized mathematical object. Nevertheless, there has been some prior work [10] and we define them here because they are particularly useful for solving and analyzing numerical methods that approximate PDEs. A discrete form can be thought of as a quantity integrated over a point (0-form), line (1-form), surface (2-form), or volume (3-form). Discrete forms are not "the things which occur under integral signs" but the entire integral quantity. As with continuous differential forms the dimension is usually explicitly noted as a superscript on the form. For example, when solving PDEs, a discrete 1-form is $v^1 = \int_{n_1}^{n_2} \mathbf{v} \cdot d\mathbf{l}$. This is usually calculated along the edges of a mesh. Similarly a discrete 3-form is a volume integral $a^3 = \int_{\Omega_i} a\,dV$ on a cell volume, Ω_i. Integration over a point is trivial, it just returns the value at

that point. So a discrete 0-forms acts just like a scalar fields, except that values are only known at certain locations (usually mesh vertices/nodes). We use the same notation for discrete and differential forms in this paper and attempt to make it clear in the text if some differentiation between the two is necessary.

The fact that forms combine the field variable along with a geometric component is what makes them useful and elegant. Note that a continuous differential form is defined at every point in the domain but a discrete form is defined on a mesh. The discrete 1-forms are on the mesh edges, discrete 2-forms on the faces, discrete 3-forms in the volumes/elements, and discrete 0-forms at the mesh vertices/nodes. In the limit as the mesh is infinitely refined, the discrete forms are defined at every spatial location and become equal in some ways to the continuous differential forms. However, differential forms can be, and usually are, defined without this type of limiting procedure (see Appendix A).

The advantage of using integral quantities (or discrete forms) as the primary unknowns for a numerical method cannot be understated. It allows one to *exactly* transform any continuous PDE system into a discrete algebraic system (see Ref. [11] for examples). Exact transformation means that all errors/approximations happen at the (simpler) algebraic level and NOT in the approximation of the differential operators of the PDE. In essence, we can deal with the multivariable calculus exactly. Somewhat surprisingly, differential operators (div, grad, curl) are NOT the crux of the issue of numerically solving PDEs. Exact discretization (which does not imply exact solution) is the primary reason that forms are so useful for developing numerical methods that can mimic the physical and mathematical properties of the continuous PDE system.

2.1. In 3D

A discrete 0-form is integrated over a point, so it is just a point value. This is precisely what many numerical methods deal with, a finite set of discrete 0-form unknowns (point values). 0-form unknowns for each node (or vertex) in the mesh is quite common. Linear finite elements typically use these unknowns. Cell centered unknowns, such as for some finite volume methods, are located at the vertices of a dual mesh and are also discrete 0-forms. However, it is difficult to generate physics capturing numerical methods with *only* the use of 0-form unknowns. Other forms and their use are therefore considered below.

A discrete 1-form is a line integral of a vector quantity. Remember, the superscript on $v^1 = \int_{n_1}^{n_2} \mathbf{v} \cdot d\mathbf{l}$ represents the dimension (or topology) of the form. The integral is along any line (not necessarily a straight line) between the two points n_1 and n_2. On a mesh, the discrete points (mesh vertices) in that mesh will be referred to as nodes (hence the notation using n). Any edge of the mesh has a discrete 1-form associated with it. One critical aspect of a discrete form is that every discrete form has just a single value associated with it. This makes representing vectors with forms somewhat interesting. For a discrete system it is usual to associate each 1-form with a vector value (the vector component) along an edge of the mesh.

In 3D, vector quantities can also be naturally represented as discrete 2-forms using $w^2 = \int \mathbf{w} \cdot \mathbf{n} \, dA$ where the integral is now over a bounded surface (think a small triangle or quad). The surface does not need to be planar. The discrete 2-form is typically associated with the faces of a mesh (and fluxes). 2-forms really are different from 1-forms, despite the fact that (in 3D) both are good representations for vector quantities. Physically, the distinction between 1-forms and 2-forms often manifests itself directly, especially in a reduction of the vectors to 2D. For example, the vorticity vector behaves differently (becomes a single perpendicular component) in 2D, whereas the velocity vector becomes a 2D vector (in the 2D plane). Physically (and therefore probably numerically as well) these two vectors should be discretized/represented differently. The distinction between which form is physically appropriate is often answered by what boundary conditions are used for a vector variable, or what jump conditions are found at material interfaces (which is a type of boundary condition). Tonti [12] addresses this issue in considerable detail.

Finally we have discrete 3-forms, which in 3D are volume integrals, $s^3 = \int_{\Omega_i} s \, dV$ over some bounded volume, Ω_i. In 3D, the 3-forms are naturally suitable for scalar quantities (just like the 0-forms). The letters we use to represent the forms are arbitrary at this point, but the superscripts (the 3 in this case) are important. Forms force us to note the distinction between things like mass which are inherently volume integral quantities (3-forms), and things like density which are inherently point values (0-forms).

It is easy to think of the continuous differential forms as the infinitesimal counterparts. Especial given our crude description as "the things inside integrals". For example we might represent a continuous 3-form as $s^3 = s\,dV$ where s is a scalar function. Everything about the form notation suggests this interpretation. For example formally the 3-form is actually given by $s^3 = s\,dx \wedge dy \wedge dz$. The wedge symbol is defined next (in Section 3), but this certainly suggests a small volume (like $s^3 = s\,dV$). But formally, as we mentioned earlier, smallness is not actually required by the notation. As shown in Appendix A, mathematicians will treat the dx and dy in this expression much as if they were *unit* basis vectors.

The key idea from this section is that every 0-form and 3-form has a scalar function associated with it, and every 1-form and 2-form (in 3D) has a vector associated with it. In Cartesian coordinates a 1-form and its associated vector are essentially the same thing. In arbitrary coordinate systems the 1-form looks essentially like the co-variant representation of a contra-variant vector (when using a Euclidean metric). But the point of this paper is not to be an introduction to forms, so we leave any further exploration of forms to Appendix A and references. Our primary concern is simply how to translate expressions that use forms, back into old-fashioned vector notation.

2.2. Vector discretization problem

In 3D there are two types of natural scalar forms: 0-forms, which correspond to point unknowns (Finite Difference and many Finite Element methods use these), and 3-forms which correspond to volume averaged discrete unknowns (Finite Volume methods, and some Discontinuous Galerkin (DG) methods use these). However, it is important to note that forms suggest that vector quantities (such as velocity and electric field) are ***not*** well represented this way (by 0-forms or 3-forms). Forms suggest that line average or face average discrete unknowns are the more natural way to set up the discrete system to obtain the correct physics for vector PDEs. This observation is reinforced by the evidence of existing numerical methods that capture physics well. Staggered mesh methods use face average normal velocities (2-forms) as the primary unknowns [1,4], and Nedelec (or Whitney) elements use line average (1-forms) as the primary unknowns for the electric and magnetic field vectors [3,5].

2.3. In 2D

The discrete 0-form (scalar point value on nodes) and discrete 1-form (vector line integral on edges) remain essentially the same. The number of components in the 1-form's vector proxy is now 2 instead of 3, but this is fairly trivial because the discrete 1-form itself remains just a single number. Volumes no longer exist in 2D. Mesh cells are represented by areas, not volumes. The discrete 2-form is therefore a cell average integral of a scalar field. It behaves very much like the 3-form does in 3D. This is true in general; the N-form in N-space represents a scalar field well.

Note that the behavior of the 2-form in 2D space is actually consistent with the 3D definition of the 2-form. Assume a 2D (planar) mesh is embedded into 3D space (the PDE is solved on a plane, but in 3D). In this case, the 3D normal for every face points exactly the same direction (let's say it is the 3-direction, out of the plane). Then the 3D definition of the 2-form only ever integrates (or cares about) the 3-component (or plane normal component) of the constructing 3D vector field **w**. In this sense, the 2-form only captures/represents a scalar field (the 3-component of the vector, **w**).

2.4. In 4D

As the dimension changes, it is the forms in the middle of the sequence that change the most. Discrete 0-forms (on one end) always represent point values of scalars, and discrete N-forms (on the other end of the sequence) always represent cell averages of scalar values (where a cell always sits in N, the highest dimension).

Let time be the 4th dimension, so we have a clear way of describing the extra dimension. Then a discrete 0-form is a scalar value at a certain location and at a certain time, and a discrete 4-form is a space–time integral of a scalar. For simplicity think of a mesh where time is discretized in slabs (Fig. 1) that go from one time level to the next. Then the mesh can move during that time interval, and the volumes (areas in the picture) can change with time (from the bottom to the top of each slab). So the discrete 4-form is $s^4 = \int_{t^n}^{t^{n+1}} dt \int_{V(t)} s\, dV$.

Note that this unknown does not have a time level, since it is integrated over a time interval. Pressure in the incompressible Navier–Stokes equations is frequently best discretized as a time integral. The desire to associate the pressure unknowns with a particular time level is ill-advised as it causes significant confusion/complication in many numerical implementations (such as fractional step methods).

A discrete 1-form in 4D takes a 4-vector and dots it with a 4-vector that represents the line along which it is integrated. A straight line in 4D is just a space–time vector. Assume you have a mesh point that moves at a constant speed over the time interval Δt, then the space–time displacement of the mesh point is $\mathbf{d} = (\Delta x, \Delta y, \Delta z, \Delta t)$ and this is a 4D time-edge. These time-edges (thin lines in Fig. 1) complement the regular mesh edges (solid lines in Fig. 1) that have no displacement along the time axis. For a fixed-in-time (stationary) mesh, all regular (spatial or 3D) edges of the mesh have no time component (lie in the t^n or t^{n+1} planes in Fig. 1), and the other (4D) edges only have a time component (these are the thin lines in Fig. 1), so a 1-form is either a regular edge integral of a 3D vector (at the beginning or ending time level of the slab), or an integral of a scalar value over the time interval. In the incompressible Navier–Stokes system this means that a possible discrete 4D (space–time) 1-form is the velocity vector integrated on mesh edges at a fixed time level **AND** the pressure (at mesh nodes) integrated over the time interval.

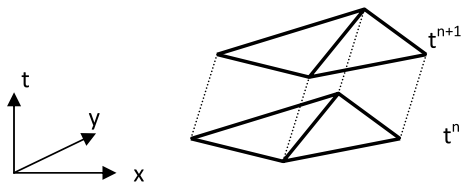

Fig. 1. A small, moving in time, 2D mesh (two triangles). The same happens for a 3D mesh moving in time but a 4D picture is too complex to draw. For a fixed (not moving) mesh the time-edges (thin dotted lines) are straight along the time axis and perpendicular to the spatial dimensions.

In 4D, a discrete 3-form is a volume integral of a scalar value at one of the time levels in a slab mesh (just like a 3D discrete 3-form), or a face-normal component of a vector (in 3D this would be a discrete 2-form) integrated over the time interval. This later possibility is just a time integrated face flux. Note that in 4D, a discrete 3-form is the primary unknown of a Finite Volume method on a (possibly moving) mesh. The exact discretization of the left-hand side of the Reynolds Transport theorem

$$\frac{d}{dt}\int\limits_{V(t)} s\,dV + \int\limits_{V(t)} \nabla\cdot(\mathbf{u}s)\,dV \tag{1}$$

is

$$\int\limits_{V(t)} s\,dV|^{n+1} - \int\limits_{V(t)} s\,dV|^{n+1} + \int\limits_{t^n}^{t^{n+1}} \left\{ \int\limits_{S(t)} \mathbf{n}\cdot(\mathbf{u}s)\,dS \right\} dt \tag{2}$$

which uses exactly the discrete 3-form components (when viewed in 4D). The 4D vector field from which these discrete 3-forms were constructed is $(s, s\mathbf{u})$. The negative sign for the t^n volume integral just accounts for the fact that this 3D face of the 4D volume points into (rather than out of, $+$) the 4D space–time volume. Keeping track of inwards and outwards orientations is important in topology and exterior calculus. But orientation tracking boils down to accounting in practice, and so it is not emphasized in this work.

In 4D, the 2-form is fairly novel. It is constructed from a super-vector of 6-components (essentially two 3D vectors, $\mathbf{f_a}$ and $\mathbf{f_b}$). For a stationary mesh the discrete 2-form is either the face-normal averages of the first 3D vector field ($\mathbf{f_a}$) at a fixed time level (just like the discrete 2-forms in the 3D case), or it is a time average of the line integral of the other 3D vector ($\mathbf{f_b}$) which is like a time average of the discrete 3D 1-forms. In space–time, the motion of an edge in time sweeps out a 2D surface. See Fig. 2.

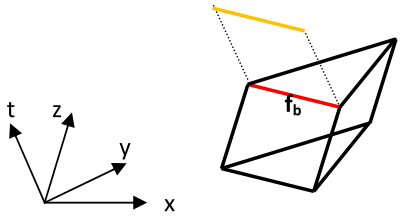

Fig. 2. One edge of a 3D mesh cell moving in time. Initially (at time t^n) the edge is red (lower line marked $\mathbf{f_b}$) and at the final time (t^{n+1}) the edge is yellow (upper line at the top of the figure). The moving edge makes a 2D surface in the 4D space.

Note that in 3D, forms suggests that there are two fundamentally different types of vectors (see discussion above). In 4D, forms suggest that there are 3 different types of vectors, or ways of correctly representing velocity or electric field. The 2-forms are a pairing of two related vectors (like electric and magnetic field or velocity and vorticity).

2.5. Summary

Table 1 shows that Pascal's triangle is involved in the number of natural inputs required for each type of form in each dimension. In 4D, for forms 0 through 4, the required inputs are (1, 4, 6, 4, 1). The fact that the 2-form in 4D takes in two 3D vectors is not accidental, but directly related to how Pascal's triangle (and forms) are constructed. It is also not accidental that the 4D 1-form and the 4D 3-form require a scalar plus a 3D-vector as their input. From Table 1 it is clear that in 5D a 2-form requires (or represents) 10 'inputs' (a 4-vector and a super-vector, or a scalar and three 3-vectors).

Table 1
Type of input (scalar, vector, etc.) required to generate each type of form (up to 4-forms) in each dimension (up to 4D).

	0-form (point)	1-form (line)	2-form (area)	3-form (volume)	4-form (volume $*$ interval)
1D	scalar	scalar			
2D	scalar	2-vector	scalar		
3D	scalar	3-vector	3-vector	scalar	
4D	scalar	4-vector	super-vector	4-vector	scalar

The dimension of the form refers to the dimension over which the quantity is integrated (point, line, area, volume, space–time), not the dimension of the space in which it sits. In a 3D numerical discretization, a discrete 1-form is represented by a single number for each mesh edge. All three components of the original vector that generated that discrete 1-form cannot be recovered without some sort of approximation and interpolation. However, for a *differential* 1-form the edges of the mesh are so small that you *can* essentially recover the 3 components of the generating vector at each location

(since nearby mesh edges are close enough in the infinitesimal limit). Hence *differential* 1-forms behave like (but are not identical to) a vector in many ways. However, discrete 1-forms are always distributed vector component values. Recovering vector fields from distributed discrete components is the subject of Refs. [13,14].

3. Multiplication

When dealing with differential forms the exterior product or wedge product, $\wedge$, means multiplication. For two different forms (of possibly different dimension) multiplication results in an addition of their dimensions. You can multiply different forms by each other as long as the resulting dimension of the product is not larger than the dimension of the space you are sitting in. So formally,

$$a^j \wedge b^k = c^{j+k} \quad \text{for } j+k \leqslant N \tag{3a}$$

where N is the dimension of the space (most commonly $N=3$ for our examples). Note that switching the order of multiplication changes the sign sometimes. So

$$a^j \wedge b^k = (-1)^{jk} b^k \wedge a^j \tag{3b}$$

Multiplication is also associative so

$$\left(a^j \wedge b^k\right) \wedge c^l = a^j \wedge \left(b^k \wedge c^l\right) \quad \text{for } j+k+l \leqslant N \tag{3c}$$

Formally, these three statements and the fact that the wedge product is bilinear, fully define the multiplication operation on forms (or the wedge product). This is the beauty of exterior calculus (maximal simplicity). But this description of multiplication of forms is also very abstract. One still does not know what the wedge product (or multiplication) operation is functionally (for example, how to perform it in a computer program).

For this reason, the wedge product (multiply), is spelled out in practical operational terms below, for each dimension and for each type of differential form. Note that in this section, we are really showing a translation, not equivalence. The cross product or dot product is on the vector proxy (the vector that would produce that form). A formal notation for transforming vectors to 1 forms and back again is discussed in Section 9 almost at the end of the paper. We do not use it here, because it only adds unnecessary complexity.

3.1. In 3D

The wedge product is essentially either a cross-product $\times$, or a dot-product $\cdot$, or a regular multiply ( ) depending on the dimension of the two forms involved in the multiply. The combinations are summarized in Table 2.

Table 2
Manifestations of the wedge product in 3D, depending on its two operands. Blank spaces indicate that the wedge product is not defined for these combinations of operands. This table shows what the wedge product on two forms means for the vector proxies associated with those two forms.

	b^0	b^1	b^2	b^3
a^0	( )	( )	( )	( )
a^1	( )	$\times$	$\cdot$	
a^2	( )	$\cdot$		
a^3	( )			

If either operand (in $a^i \wedge b^j$) is a scalar 0-form (top row or first column) then the regular multiply is called for. Remember that 1-forms and 2-forms behave much like vectors, but a 0-form behaves like a scalar and it is fairly clear how one ought to multiply a scalar times a vector or a scalar times a scalar (0-form times a 3-form, for example). Note that of all the multiply operations, only the cross-product is anti-symmetric in the two operands and it only occurs between two 1-forms (producing a 2-form). 2-forms, although behaving much like vectors, cannot participate in a cross-product (directly) because the resulting dimension would be 4, which is too high for the 3D space. Similarly, the common identity $(\mathbf{a}\times\mathbf{b})\cdot\mathbf{c} = \mathbf{a}\cdot(\mathbf{b}\times\mathbf{c})$ only applies to three 1-forms (in 3D) if one were to identify a form with each of the vectors, so we can also write $(a^1 \wedge b^1)\wedge c^1 = a^1 \wedge (b^1 \wedge c^1)$, which is Eq. (3c). Note that the wedge of two 1-forms is cross product on the vector proxies that results in a 2-form, and the wedge of that 2-form with a 1-form is (from Table 2) a dot product of the two vector proxies.

Table 2 shows that, in 3D, the dot product is only between the two different types of vectors (a 1-form and a 2-form). While multivariable calculus makes little distinction between the two types of scalars (0-forms and 3-forms) and two types of vectors (1-forms and 2-forms), the wedge product implies that more care should be observed. Differential forms help

to enforce the proper level of rigor by having superscripts that force us to remember what type of vector or scalar we are operating with.

3.2. In 2D

In this case the sum of the two operand dimensions must be less than or equal to 2 so the only interesting operation is between two 1-forms, $a^1 \wedge b^1$. If the form a^1 is associated with the vector $\mathbf{a}$, and the form b^1 is associated with the vector $\mathbf{b}$ then $a^1 \wedge b^1$ can be translated as equivalent to $\mathbf{a} \times_{2D} \mathbf{b} = a_x b_y - a_y b_x$. This is a 3D curl that is deprecated to 2D (3rd component of the result is extracted to a scalar result). If you rotate one of the vector arguments by 90 degrees this operation is also a 2D dot product. The 2D cross product is anti-symmetric in the arguments (as it should be). The wedge product in 2D is summarized as

	b^0	b^1	b^2
a^0	()	()	()
a^1	()	$\times_{(2D)}$	
a^2	()		

Note that the 2D dot product does not appear in this table.

3.3. In 4D

The multiply operation (wedge product) is summarized below for 4D and then explained.

	b^0	b^1	b^2	b^3	b^4
a^0	()	()	()	()	()
a^1	()	$\times_{(4D)}$	$\circ$	$\otimes$	
a^2	()	$\circ$	$\cdot_{(4D)}$		
a^3	()	$\otimes$			
a^4	()				

The 4D cross product $\times_{(4D)}$ and the new 4D multiply operation $\otimes$ are anti-symmetric operations.

The table above shows that the multiplication of two 4D 1-forms $a^1 \wedge b^1$ is given by the 4D cross product. Using 3D vectors $\mathbf{v}$ and $\mathbf{w}$ this is,

$$a^1 \wedge b^1 \Rightarrow \mathbf{a} \times_{(4D)} \mathbf{b} = \begin{pmatrix} \mathbf{v} \\ q \end{pmatrix} \times_{4D} \begin{pmatrix} \mathbf{w} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{v} \times \mathbf{w} \\ \mathbf{w}q - p\mathbf{v} \end{pmatrix} = \begin{pmatrix} \mathbf{f}_a \\ \mathbf{f}_b \end{pmatrix} \tag{4a}$$

The arrow means 'translates into'. The result is a super-vector (two 3D vectors, $\mathbf{f}_a$ and $\mathbf{f}_b$). The first part of the resulting super-vector is the regular 3D curl, and the second part is an anti-symmetric vector scalar product with the fourth dimension components (p and q).

The 4D version of the dot product operation takes a 1-form (4D vector) and a 2-form (super-vector) as its arguments and produces a 3-form (a 4D vector) as its result. Using 3D vectors this is.

$$a^1 \wedge b^2 \Rightarrow \mathbf{a} \circ \mathbf{f} = \begin{pmatrix} \mathbf{v} \\ q \end{pmatrix} \circ \begin{pmatrix} \mathbf{f}_a \\ \mathbf{f}_b \end{pmatrix} = \begin{pmatrix} \mathbf{v} \cdot \mathbf{f}_a \\ \mathbf{f}_a q - \mathbf{v} \times \mathbf{f}_b \end{pmatrix} \tag{4b}$$

This has the classic dot product on the top (resulting in a scalar value) and the lower dimensional multiplies for the lower (vector) part of the result. The larger symbol for the dot indicates that this is a 4D dot product. This operator is symmetric (it does not matter if the 1-form or the 2-form argument is first or second).

The result of multiplying a 1-form (associated with a 4-vector) and a 3-form (also associated with a 4-vector) should produce a 4-form (a scalar). A 4D version of the classic dot-product (sum of multiplied components) would produce the correct dimensional result, but does not have the correct anti-symmetry property. The correct multiply is a new operation that is unique to 4D and looks almost like a dot product.

$$a^1 \wedge b^3 = \mathbf{a} \otimes \mathbf{b} = \begin{pmatrix} \mathbf{v} \\ q \end{pmatrix} \otimes \begin{pmatrix} p \\ \mathbf{u} \end{pmatrix} = pq - \mathbf{v} \cdot \mathbf{u} \tag{4c}$$

Note that this is anti-symmetric. It (and the other 4D operations) uses a cross pattern similar to the determinant. If the order is reversed the cross-pattern remains and the sign switches, so this operator is anti-symmetric,

$$b^3 \wedge a^1 = \mathbf{b} \otimes \mathbf{a} = \begin{pmatrix} p \\ \mathbf{u} \end{pmatrix} \otimes \begin{pmatrix} \mathbf{v} \\ q \end{pmatrix} = \mathbf{v} \cdot \mathbf{u} - pq$$

Finally, the multiply of two 2-forms is now possible in 4D. It generates a scalar result and looks almost like the classic dot product operation but with the cross pattern found in all the 4D operators.

$$\begin{pmatrix} \mathbf{g}_a \\ \mathbf{g}_b \end{pmatrix} \cdot_{4D} \begin{pmatrix} \mathbf{f}_a \\ \mathbf{f}_b \end{pmatrix} = \mathbf{f}_b \cdot \mathbf{g}_a + \mathbf{f}_a \cdot \mathbf{g}_b \tag{4d}$$

The 4D dot-product is also symmetric and takes two super-vectors as arguments.

3.4. Finite forms

In practice, when dealing with finite forms, it has already been observed that vector (and super-vector) quantities do not really exist at a single location. So the wedge product is difficult to generalize to the finite case unless some mechanism to 'reconstruct' single-location vectors (and super-vectors) from finite forms is provided. See Refs. [13,14] for a detailed discussion of mimetic reconstruction.

Fortunately, the wedge product (or higher dimensional multiplication with anything other than a scalar) is rarely needed in the formulation of PDEs. For example, Maxwell's equations are linear and no multiplication appears in the PDE (except with scalar material constants). The Navier–Stokes equations are linear in all but the advection term. The advection term is discussed in detail in Section 7.

Despite the wedge product's seeming lack of relevance for PDEs, it has been introduced in this paper to motivate and support the next section which involves the discussion of differential operations (that do appear all over PDEs). The classic notion for the gradient $\nabla()$, curl $\nabla\times$, and divergence $\nabla\cdot$, involves nabla and the multiply (or wedge) operation because of this very close correspondence between multiplication and differentiation.

4. Differentiation

The exterior derivative operator: $\mathbf{d}$ behaves much like multi-dimensional differentiation (or in 3D: $\nabla, \nabla\times, \nabla\cdot$). This operation (like multiply) takes a form to a different dimension. Differentiation takes forms to the next higher dimensional form. In summary,

$$\mathbf{d}a^i = b^{i+1} \tag{5}$$

4.1. In 3D

The operators $\nabla, \nabla\times, \nabla\cdot$ are quite familiar. The gradient of a scalar (0-form), $\nabla a = \mathbf{b}$ is equivalent to $\mathbf{d}a^0 = b^1$, which shows that the resulting vector is best represented as a 1-form. The curl of a (1-form) vector, $\nabla \times \mathbf{a} = \mathbf{b}$ is equivalent to $\mathbf{d}a^1 = b^2$, showing that the resulting vector is actually a 2-form. For example if velocity is a 1-form then vorticity (the curl) is a 2-form. Finally the divergence $\nabla \cdot \mathbf{a} = b$ of a (2-form) vector is equivalent to $\mathbf{d}a^2 = b^3$ and the result is scalar (3-form). The same symbol is used for the exterior derivative on forms to highlight the similarity between the divergence, the gradient, and the curl. Forms suggest that it is the operands, and not the operator which are fundamentally different in each case. And this viewpoint is very useful when constructing numerical methods that properly mimic those differential operators.

In 3D it is not possible to take the exterior derivative of a 3-form. This is true in general; the highest form in any space cannot be differentiated. If one really wants to differentiate a 3-form then the 3-form must first be converted to another form (usually a 0-form). The conversion of forms is the essence of Section 6. Conversion adds error to a numerical method and is therefore should be invoked judiciously in a numerical method.

Of great importance to PDE solution is the fact that these differentiation ideas extend directly to discrete differentiation. Consider a discrete 1-form whose underlying vector generator $\mathbf{b}$ is the vector field produced by taking the gradient of a scalar, a. Using the fundamental theorem of calculus the integral of a gradient along a line segment is

$$b^1 = \int_{n_1}^{n_2} \mathbf{b} \cdot d\mathbf{l} = \int_{n_1}^{n_2} \nabla a \cdot d\mathbf{l} = a|_{n_2} - a|_{n_1} = \mathbf{G}a|_{n_i} = \mathbf{G}a^0 \tag{6a}$$

The discrete 1-form of the gradient (located on an edge) can be ***exactly*** computed from the 0-form values (located on the edge ends, or at the mesh nodes). Note that this is a simple subtraction of the two discrete 0-form values. With the + and – determined by the orientation of the edge (which is chosen arbitrarily). The matrix $\mathbf{G}$ is the discrete gradient matrix, it is the discrete equivalent of $\mathbf{d}$ that acts on discrete 0-forms.

Similarly

$$b^2 = \int_S \mathbf{b} \cdot \mathbf{n}\, dS = \int_S (\nabla \times \mathbf{a}) \cdot \mathbf{n}\, dS = \int_{\partial S} \mathbf{a} \cdot d\mathbf{l} = \mathbf{C} \int_{e_i} \mathbf{a} \cdot d\mathbf{l} = \mathbf{C}a^1 \tag{6b}$$

Using Stokes Theorem the discrete 2-form can be exactly obtained from the discrete 1-form values on the edges that bound the 2-form's surface. The surface need not be planar and the edges need not be lines, but the edges must surround the surface. Note that this is again a simple summation operation of the 1-forms but with a + or − to account for the edge and

surface orientations. An arbitrary orientation is chosen for the surface and the edges. Then if the edge and surface normals obey a right-hand rule (+) is used, and otherwise the (−) is used in the summation. The matrix **C** is the discrete curl matrix, which is the discrete equivalent of **d** that acts on discrete 1-forms.

Finally, we also have

$$b^3 = \int_V b\,dV = \int_V \nabla\cdot\mathbf{a}\,dV = \int_{\partial V} \mathbf{a}\cdot\mathbf{n}\,dS = \mathbf{D}\int_{f_i} \mathbf{a}\cdot\mathbf{n}\,dS = \mathbf{D}a^2 \tag{6c}$$

which uses Gauss' Divergence Theorem. It produces an exact discrete 3-form from the cell surface discrete 2-form values. Again simple summation is used, with a (+) if the surface is oriented out of the cell, and (−) if the surface points into the cell. How the orientation of a surface is chosen/defined is arbitrary (and irrelevant to the results) as long as it has a specific orientation. The matrix **D** is the discrete divergence matrix and is the discrete equivalent of **d** that acts on discrete 2-forms.

Each discrete difference operation (or exterior derivative, **d,** operation) is just an oriented summation of the forms on the one lower dimension that bounds the original form; hence the name exterior calculus. For example, a divergence (for a 3-form or cell) is exactly calculated by summing single values (2-forms) from that cell's faces (i.e. the cell's exterior). Similar relations with respect to topology and dimension hold for the curl and the gradient. At the infinitesimal level, the topological and dimensional information is not important and so multivariable calculus abandons it. However, clearly topological relationships exist in the finite realm (such as for Stokes' and Gauss' theorems), and since PDEs are actually solved on computers in the finite realm, this topological aspect of the problem should not be neglected when one intends to solve PDEs with finite numbers of unknowns.

Note the similarity with multiplication. Similar operations (and order) are found in the second row of Table 2 (for 3D multiplication). This is the row that shifts the dimension of the result by 1 (like differentiation does).

4.2. 2D

This actually has some complexity to it because of the tendency to think in 3D. In two dimensions only two primary differentiation operations should exist. The analogy with multiplication suggests that gradient, ∇, and 2D curl, $\nabla\times_{2D}$, should be used and that the divergence does not add informative content in 2D.

In 2D, the gradient looks almost the same. It is the difference between two (discrete 0-form) node values. The curl on edges that lie in a plane always results in the calculation of the vector component that is normal to the plane. This is really just a single component of information. The 2D curl $\nabla\times_{2D}\mathbf{a} = \frac{\partial a_y}{\partial x_x} - \frac{\partial a_x}{\partial x_y}$ therefore produces a scalar result. This is correct because the 2D curl takes the edges (1-forms) as input and produces a 2-form result. In 2D the 2-form is associated with a scalar.

If one considers a 2D cell and its boundary (the cell faces), such as in Fig. 3, then the 2D curl contains almost the same fundamental information as a 2D divergence. The discrete curl is the identical operation to a rotation of every vector in the field by 90 degrees clockwise and then taking a 2D divergence. In 3D this equivalence is not present because an obvious axis of rotation is not present (as it is in 2D).

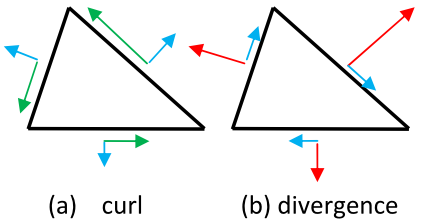

Fig. 3. The 2D curl is shown on the left. It is a sum of the green (tangential) vector components along the edges. On the right is the vector field rotated 90 degrees clockwise. The same summation of the same components (now red) is now a divergence, and now normal. Blue vector components do not participate in either operation.

Alternatively, sometimes the set of curl and divergence is used in 2D (with no gradient operation). This is less intuitive, but is possible by viewing the 2D mesh in a 3D sense (where the divergence really exists). To do this, the 2D mesh is extruded in the 3rd direction. See Fig. 4. The values on the two planes (top and bottom) are assumed to be the same since one is just an extrusion of the other. The curl then results in a simple difference of two values (at nodes in the 2D mesh) because of cancellation. This curl now looks like a gradient but the result is perpendicular to the 2D mesh edges, not along the 2D mesh edges (like a regular gradient). Similarly, the divergence ends up looking like the curl of 90 degrees rotated 2D vector field (as in Fig. 3 above).

4.3. 4D

In this case there is another dimension (time). There is therefore another fundamental differential operation. The 4D differentials will be discussed in terms of the classic 3D terminology.

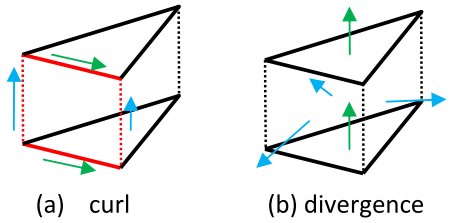

Fig. 4. The 2D triangle is shown extruded upwards (along the dotted lines) into 3D. The curl on the red (front) face involves 4 edges, but the two green (upper and lower) components are identical (because they are the same 2D values extruded into 3D) and their signs make them cancel, so the curl really functions as a gradient of the two 2D node values. Similarly, for the 5 faces of the 3D divergence shown in (b), the two green contributions (top and bottom faces) cancel, giving the situation shown in Fig. 3 (which is really a curl).

The 4D gradient is still simple,

$$\nabla_{4D} s = \left(\nabla s, \frac{\partial s}{\partial t} \right) \tag{7a}$$

It takes a scalar point value and produces the 3D gradient and the time derivative (4 things, or a 4-vector). The discrete gradient lies on edges in space–time. For a fixed mesh in 4D we have the usual mesh edges as well as edges that are created by nodes values moving over the time interval. The 4D discrete gradient is therefore either located on mesh edges and is the difference between two node values (as in 3D), or it lies at node locations and is the difference of the node value at two different times (discrete time difference of the scalar s at the nodes).

As with multiplication in 4D, differentiation of a 1-form (4 vector) produces a 2-form or super-vector (two 3D vectors) proxy. The curl in 4D is given by

$$\nabla \times_{4D} \mathbf{a} = \nabla \times_{4D} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} \nabla \times \mathbf{u} \\ \frac{\partial \mathbf{u}}{\partial t} - \nabla p \end{pmatrix} \tag{7b}$$

The first half of the resulting super-vector is just the standard 3D curl, and the second half is the time derivative of the vector and the gradient (one lower dimensional differentiation operation) of the extra scalar. On a stationary mesh, the first half of the resulting (2-form) super-vector is located on the mesh faces (as in 3D), and the second half of the super-vector result is on edges integrated over the time interval. Note that the curl of the gradient is still zero in 4D,

$$\nabla \times_{4D} \nabla_{4D} s = \begin{pmatrix} \nabla \times \nabla s \\ \frac{\partial \nabla s}{\partial t} - \nabla \frac{\partial s}{\partial t} \end{pmatrix} = 0 \tag{8a}$$

The 4D divergence (in analogy to multiplication) should take the super-vector (2-form) and reduce it back to a regular 4-vector (3 form). It is given by

$$\nabla \circ \mathbf{f} = \nabla \circ \begin{pmatrix} \mathbf{f}_a \\ \mathbf{f}_b \end{pmatrix} = \begin{pmatrix} \nabla \cdot \mathbf{f}_a \\ \frac{\partial \mathbf{f}_a}{\partial t} - \nabla \times \mathbf{f}_b \end{pmatrix} \tag{7c}$$

Again, the first part is the classic 3D divergence (a scalar) and the last part involves the one lower dimensional differentiation operation (the curl). For the discrete version (using integration) the first part of the result is integrated on the mesh cells at a fixed time, and the second part is integrated on the mesh faces and integrated over the time interval. Note that the divergence of a curl is still zero in 4D (and any dimension).

$$\nabla \circ \nabla \times_{4D} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} \nabla \cdot \nabla \times \mathbf{u} \\ \frac{\partial \nabla \times \mathbf{u}}{\partial t} - \nabla \times \left(\frac{\partial \mathbf{u}}{\partial t} - \nabla p \right) \end{pmatrix} = 0 \tag{8b}$$

The final 4D differentiation operation is new. It takes a 4-vector (3-form) and produces a scalar (4-form).

$$\nabla \otimes \mathbf{b} = \nabla \otimes \begin{pmatrix} p \\ \mathbf{u} \end{pmatrix} = \frac{\partial p}{\partial t} - \nabla \cdot \mathbf{u} \tag{7d}$$

The second part again uses the one lower dimension differential operation (3D divergence). For the discrete version this produces a scalar result integrated on the 3D mesh volumes and integrated over the time interval. Many scalar transport systems (including the Reynolds transport theorem) have this basic form (with $\mathbf{u}$ as the flux). Not too surprisingly, this new 4D operation when operating on the divergence is always zero, since

$$\nabla \otimes \nabla \circ \begin{pmatrix} \mathbf{f}_a \\ \mathbf{f}_b \end{pmatrix} = \frac{\partial \nabla \cdot \mathbf{f}_a}{\partial t} - \nabla \cdot \left(\frac{\partial \mathbf{f}_a}{\partial t} - \nabla \times \mathbf{f}_b \right) = 0 \tag{8c}$$

4.4. Repeated differentiation (sequence property)

Repeated differentiation is written using exterior calculus as $\mathbf{dd} = 0$. Remember the functional form of $\mathbf{d}$ depends a great deal on what it is operating on. In the above expression each $\mathbf{d}$ operator is different because it must operate on a different type of form. This expressions actually says that each differential operators represented by $\mathbf{d}$ will zero the one-lower dimensional $\mathbf{d}$ operator. In 3D this repeated differentiation is similar to the expressions, $\nabla \times \nabla = 0$ and $\nabla \cdot \nabla \times = 0$. In 4D the expressions are $\nabla \times_{4D} \nabla_{4D} = 0$, $\nabla \circ \nabla \times_{4D} = 0$ and $\nabla \otimes \nabla \circ = 0$ (Eqs. (8a)–(8c)).

This relationship has very important mathematical and physical implications for numerical methods. Methods that respect these relationships between the discrete operators have advantageous conservation properties, better accuracy than other methods of the same order, and lack unphysical 'modes'. See Refs. [15–22] for examples. Note that obtaining matrices which respect these properties is not difficult. It was already performed in the Section 4.1 above. The discrete derivative matrices always obey the properties $\mathbf{CG} = 0$ corresponding to $\nabla \times \nabla = 0$ and $\mathbf{DC} = 0$ corresponding to $\nabla \cdot \nabla \times = 0$. One can see this most easily just using geometric (topological) reasoning. See Fig. 5.

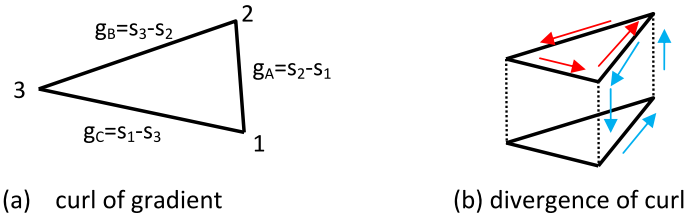

Fig. 5. (a) The discrete gradient **G**s on each edge (A,B, or C) is the difference between two node values. The minus sign determines the edge orientation. In this case the edges are all oriented counterclockwise. But the orientation choice is arbitrary for each edge. A curl **C** is the oriented sum of these edges (in this case all positive, because all counterclockwise). Every node value therefore appears twice and always with opposite sign, so everything cancels and $\mathbf{CG} = 0$. This is true for any collection of curves which enclose an area. (b) The curl, **C**v, on two faces of a polyhedron are shown. In both cases, the orientation of the face is outwards because the curls have a right-hand-rule sign convention with the face orientation. The divergence, **D**, is the summation of all the face values on the polyhedra (with + for outward faces and – for inward faces). Every edge will contribute twice to the final result, and always in equal and opposite ways, for a final result of zero. So $\mathbf{DC} = 0$.

5. Product rule

The product rule captures the relationship between differentiation and multiplication. This idea can be extended to arbitrary dimensions. In exterior calculus the product rule is often called Leibniz's rule and is written as

$$\mathbf{d}\left(a^j \wedge b^k\right) = \left(\mathbf{d}a^j\right) \wedge b^k + (-1)^j a^j \wedge \left(\mathbf{d}b^k\right) \quad \text{for } j + k < N \tag{9}$$

Let us expand it out to see what this means in precise detail.

5.1. 3D

If $j = 0$, then one of our forms is a 0-form, a point valued scalar quantity, and

$$\mathbf{d}\left(a^0 \wedge b^k\right) = \left(\mathbf{d}a^0\right) \wedge b^k + a^0 \wedge \left(\mathbf{d}b^k\right) \quad \text{for } k < 3$$

which translates into

$$\nabla(st) = (\nabla s)t + s(\nabla t) \quad \text{for } k = 0 \tag{10a}$$

$$\nabla \times (s\mathbf{v}) = (\nabla s) \times \mathbf{v} + s(\nabla \times \mathbf{v}) \quad \text{for } k = 1 \tag{10b}$$

$$\nabla \cdot (s\mathbf{w}) = (\nabla s) \cdot \mathbf{w} + s(\nabla \cdot \mathbf{w}) \quad \text{for } k = 2 \tag{10c}$$

These describe how scalar multiplication commutes with the various differential operators.

If $j = 1$, then one of the operands is a 1-form (vector on an edge) and

$$\mathbf{d}\left(a^1 \wedge b^k\right) = \left(\mathbf{d}a^1\right) \wedge b^k - a^1 \wedge \left(\mathbf{d}b^k\right) \quad \text{for } k < 2$$

This translates into one more expression because $j = 1$ and $k = 0$ is the same as the second expression above (which uses $j = 0$ and $k = 1$). So

$$\nabla \cdot (\mathbf{v} \times \mathbf{w}) = (\nabla \times \mathbf{v}) \cdot \mathbf{w} - \mathbf{v} \cdot (\nabla \times \mathbf{w}) \quad \text{for } k = 1 \tag{10d}$$

if $j = 2$, then $k = 0$ and this case was already found (Eq. (10c)). No higher values of j are permissible. Note that these 4 vector identities are fundamental. Other vector identities (such as the gradient of a dot product of two vectors) exist, but those identities actually involve the Hodge* operator as well which is discussed in Section 6. In the discrete case, the Hodge* always involves some form of approximation and error.

5.2. 2D

In 2D the product rule remains the same, but only two identities result. A 0-form only produces two possibilities,

$$\nabla(st) = (\nabla s)t + s(\nabla t) \quad \text{for } k = 0 \tag{11a}$$

$$\nabla \times_{2D} (s\mathbf{v}) = (\nabla s) \times_{2D} \mathbf{v} + s(\nabla \times_{2D} \mathbf{v}) \quad \text{for } k = 1 \tag{11b}$$

and the 1-form produces an identity

$$\nabla \times_{2D} (\mathbf{v}s) = (\nabla \times_{2D} \mathbf{v})s - \mathbf{v} \times_{2D} (\nabla s) \quad \text{for } k = 0$$

which is identical to the one above it. Remember that the 2D curl produces a scalar result.

5.3. 4D

The 4D case is more interesting. For 0-forms:

$$\mathbf{d}\left(a^0 \wedge b^k\right) = \left(\mathbf{d}a^0\right) \wedge b^k + a^0 \wedge \left(\mathbf{d}b^k\right) \quad \text{for } k < 4$$

So for $k = 0$ using the 4D gradient.

$$\nabla_{4D}(sr) = (\nabla_{4D} s)r + s(\nabla_{4D} r) \quad k = 0 \tag{12a}$$

In 3D notation this is

$$\nabla_{4D}(sr) = \begin{pmatrix} \nabla s \\ \frac{\partial s}{\partial t} \end{pmatrix} r + s \begin{pmatrix} \nabla r \\ \frac{\partial r}{\partial t} \end{pmatrix}$$

Then for $k = 1$

$$\nabla \times_{4D} (s\mathbf{v}) = (\nabla_{4D} s) \times_{4D} \mathbf{v} + s(\nabla \times_{4D} \mathbf{v}) \quad k = 1 \tag{12b}$$

This is a super-vector equation. In 3D notation let $\mathbf{v} = (\mathbf{u}, p)$ be a 1-form 4-vector generator. Then the previous equation can also be written as,

$$\begin{pmatrix} \nabla \times s\mathbf{u} \\ \frac{\partial s\mathbf{u}}{\partial t} - \nabla sp \end{pmatrix} = \begin{pmatrix} \nabla s \\ \frac{\partial s}{\partial t} \end{pmatrix} \times_{4D} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} + s \begin{pmatrix} \nabla \times \mathbf{u} \\ \frac{\partial \mathbf{u}}{\partial t} - \nabla p \end{pmatrix} = \begin{pmatrix} \nabla s \times \mathbf{u} \\ \mathbf{u}\frac{\partial s}{\partial t} - p\nabla s \end{pmatrix} + \begin{pmatrix} s\nabla \times \mathbf{u} \\ s\frac{\partial \mathbf{u}}{\partial t} - s\nabla p \end{pmatrix}$$

For $k = 2$,

$$\nabla \circ (s\mathbf{f}) = (\nabla_{4D} s) \circ \mathbf{f} + s(\nabla \circ \mathbf{f}) \quad \text{for } k = 2 \tag{12c}$$

where $\mathbf{f}$ is a 2-form (super-vector) and $\circ$ is the 4D wedge product for a 1-form times a 2-form. In 3D notation this expression is

$$\begin{pmatrix} \nabla \cdot s\mathbf{f}_a \\ \frac{\partial s\mathbf{f}_a}{\partial t} - \nabla \times s\mathbf{f}_b \end{pmatrix} = \begin{pmatrix} \nabla s \\ \frac{\partial s}{\partial t} \end{pmatrix} \circ \begin{pmatrix} \mathbf{f}_a \\ \mathbf{f}_b \end{pmatrix} + s \begin{pmatrix} \nabla \cdot \mathbf{f}_a \\ \frac{\partial \mathbf{f}_a}{\partial t} - \nabla \times \mathbf{f}_b \end{pmatrix} = \begin{pmatrix} \nabla s \cdot \mathbf{f}_a \\ \mathbf{f}_a \frac{\partial s}{\partial t} - \nabla s \times \mathbf{f}_b \end{pmatrix} + \begin{pmatrix} s\nabla \cdot \mathbf{f}_a \\ s\frac{\partial \mathbf{f}_a}{\partial t} - s\nabla \times \mathbf{f}_b \end{pmatrix}$$

For $k = 3$ the new derivative is used and

$$\nabla \otimes (s\mathbf{w}) = (\nabla_{4D} s) \otimes \mathbf{w} + s(\nabla \otimes \mathbf{w}) \quad k = 3 \tag{12d}$$

where $\mathbf{w}$ is a 3-form. In 3D notation taking $\mathbf{w} = (p, \mathbf{u})$ this is,

$$\frac{\partial sp}{\partial t} - \nabla \cdot s\mathbf{u} = \begin{pmatrix} \nabla s \\ \frac{\partial s}{\partial t} \end{pmatrix} \otimes \begin{pmatrix} p \\ \mathbf{u} \end{pmatrix} + s\left(\frac{\partial p}{\partial t} - \nabla \cdot \mathbf{u}\right) = p\frac{\partial s}{\partial t} - \mathbf{u} \cdot \nabla s + \left(s\frac{\partial p}{\partial t} - s\nabla \cdot \mathbf{u}\right)$$

Two 1-form cases exist in 4D. Multiplying two 1-forms produces

$$\nabla \circ (\mathbf{v} \times_{4D} \mathbf{w}) = (\nabla \times_{4D} \mathbf{v}) \circ \mathbf{w} - \mathbf{v} \circ (\nabla \times_{4D} \mathbf{w}) \tag{12e}$$

which becomes in 3D notation

$$\nabla \circ \begin{pmatrix} \mathbf{v} \times \mathbf{w} \\ \mathbf{w}p - q\mathbf{v} \end{pmatrix} = \begin{pmatrix} \nabla \times \mathbf{v} \\ \frac{\partial \mathbf{v}}{\partial t} - \nabla p \end{pmatrix} \circ \begin{pmatrix} \mathbf{w} \\ q \end{pmatrix} - \begin{pmatrix} \mathbf{v} \\ p \end{pmatrix} \circ \begin{pmatrix} \nabla \times \mathbf{w} \\ \frac{\partial \mathbf{w}}{\partial t} - \nabla q \end{pmatrix} = \begin{pmatrix} \mathbf{w} \\ q \end{pmatrix} \circ \begin{pmatrix} \nabla \times \mathbf{v} \\ \frac{\partial \mathbf{v}}{\partial t} - \nabla p \end{pmatrix} - \begin{pmatrix} \mathbf{v} \\ p \end{pmatrix} \circ \begin{pmatrix} \nabla \times \mathbf{w} \\ \frac{\partial \mathbf{w}}{\partial t} - \nabla q \end{pmatrix}$$

which becomes

$$\begin{pmatrix} \nabla\cdot(\mathbf{v}\times\mathbf{w}) \\ \frac{\partial \mathbf{v}\times\mathbf{w}}{\partial t} - \nabla\times(\mathbf{w}p - q\mathbf{v}) \end{pmatrix} = \begin{pmatrix} \mathbf{w}\cdot\nabla\times\mathbf{v} \\ -\mathbf{w}\times\frac{\partial \mathbf{v}}{\partial t} + \mathbf{w}\times\nabla p + q\nabla\times\mathbf{v} \end{pmatrix} - \begin{pmatrix} \mathbf{v}\cdot\nabla\times\mathbf{w} \\ p\nabla\times\mathbf{w} - \mathbf{v}\times\frac{\partial \mathbf{w}}{\partial t} + \mathbf{v}\times\nabla q \end{pmatrix}$$

A 1-form and a 2-form gives

$$\nabla\otimes(\mathbf{v}\circ\mathbf{f}) = (\nabla\times_{4D}\mathbf{v})\cdot_{4D}\mathbf{f} - \mathbf{v}\otimes(\nabla\circ\mathbf{f}) \tag{12f}$$

which becomes in 3D notation

$$\nabla\otimes\begin{pmatrix} \mathbf{u}\cdot\mathbf{f}_a \\ p\mathbf{f}_a - \mathbf{u}\times\mathbf{f}_b \end{pmatrix} = \begin{pmatrix} \nabla\times\mathbf{u} \\ \frac{\partial \mathbf{u}}{\partial t} - \nabla p \end{pmatrix}\cdot_{4D}\begin{pmatrix} \mathbf{f}_a \\ \mathbf{f}_b \end{pmatrix} - \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix}\otimes\begin{pmatrix} \nabla\cdot\mathbf{f}_a \\ \frac{\partial \mathbf{f}_a}{\partial t} - \nabla\times\mathbf{f}_b \end{pmatrix}$$

or

$$\frac{\partial \mathbf{u}\cdot\mathbf{f}_a}{\partial t} - \nabla\cdot p\mathbf{f}_a + \nabla\cdot(\mathbf{u}\times\mathbf{f}_b) = \mathbf{f}_b\cdot\nabla\times\mathbf{u} + \mathbf{f}_a\cdot\frac{\partial \mathbf{u}}{\partial t} - \mathbf{f}_a\cdot\nabla p - \left(p\nabla\cdot\mathbf{f}_a - \mathbf{u}\cdot\frac{\partial \mathbf{f}_a}{\partial t} + \mathbf{u}\cdot\nabla\times\mathbf{f}_b\right)$$

There are no other unique combinations in 4D. In any dimension, the number of unique fundamental product rules is $(\frac{N}{2}+1)(\frac{N+1}{2})$ where N is the dimension.

5.4. Conservation

An exact discrete product rule is useful for guaranteeing conservation. For example, energy is a key conservation variable. Remember that in 3D energy is often a multiplicative product of a 1-form and a 2-form. Gravitational potential energy is $U = -(m\mathbf{g})\cdot\mathbf{x}$, elastic or spring potential energy is $U = \frac{1}{2}(k\mathbf{x})\cdot\mathbf{x}$. The energy of a magnetic moment $\mathbf{m}$ in an externally produced magnetic B-field $\mathbf{B}$ has potential energy $U = -\mathbf{m}\cdot\mathbf{B}$, the kinetic energy is $T = \frac{1}{2}\mathbf{p}\cdot\mathbf{u}$ where $\mathbf{p}$ is the momentum and $\mathbf{u}$ is the velocity.

A discrete chain-rule that commutes with discrete differentiation is the key to deriving energy conservation statements [17,19]. Energy conservation is a type of numerical stability. If the energy remains bounded for all time, then the solution must remain bounded as well. Numerical methods built using discrete exterior calculus are therefore often stable via their construction.

6. Hodge* and inner product

The definition of the discrete Hodge* (or inner product) is the essence of any numerical method. This is because exterior calculus makes it relatively clear that calculus can be discretized exactly using finite forms. However, the result of using finite forms to discretize any PDE is that one arrives at more unknowns than equations. This is not really surprising, as the same thing happens in the physical formulation of the problem. See Tonti in this issue and in [12], and see Mattiussi [21] and the example in Section 10. In physics, material constitutive relations are required to relate the unknowns and reduce the effective problem size so that it is solvable. The same is true in the numerical setting. The discrete constitutive equations therefore augment the number of equations in the discrete equation system and make the algebraic system square and invertible. The discrete Hodge* is intimately associated with material constitutive laws. Material laws are engineering approximations, and they are also where all the numerical approximation (or numerical error) can be found.

Formally the Hodge* is an operation which takes a form in one dimension and produces a resulting form that is in the reflected dimension,

$$^*a^j = b^{N-j} \tag{13}$$

In 3D this means that a 0-form (scalar point value) goes to a 3-form (cell average of scalar) and vice-versa. And a 1-form (vector component along a line) goes to a 2-form (vector component normal to a face) and vice-versa. The parentheses describe what happens to discrete forms with a discrete Hodge*.

In practice, this means that a discrete Hodge* operation takes values on a mesh and transfers them to a dual mesh (as well as changing their dimension). Interestingly, every well formed mesh has an infinite number of well formed dual meshes. Part of defining a numerical method therefore involves identifying which dual mesh is being used for the definition of the discrete Hodge* matrices. Examples of dual meshes for unstructured (triangular or tetrahedral) primary meshes are the Voronoi dual mesh, or the median dual mesh. The discrete Hodge* takes cell node (vertex) values of a tetrahedral mesh and produces cell average values for the Voronoi cells that surround those nodes. Similarly it could take 1-forms that lie on the line between two cell centers (an edge of the dual mesh) and produce values for the normal flux on the cell faces (2-forms on the primal mesh). The definition of which mesh (the tetrahedral cells or the Voronoi cells around each vertex) is primal and which mesh is dual, is actually totally arbitrary. However, whatever mesh a generation program produces, tends to be called the primal mesh. For differential forms the primary and dual meshes are effectively infinitely refined and therefore the interpolation aspect of the discrete Hodge* disappears in the limit of continuous differential forms.

The Hodge* is closely related to an **inner product**. A positive definite N-form (or norm) can always be defined by multiplying a form by its Hodge*, $E^N = a^j \wedge (^*a^j)$. The Hodge* and the wedge product define this inner product. But

remember, there are actually many Hodge* operators; one for each type of form it operates on. A discrete Hodge* is a matrix (usually square). Sometimes it is a diagonal matrix and often the Hodge* matrix is sparse. Many physical systems can be directly derived from an Energy using either the Hamiltonian or Lagrangian formalism. The discrete system can then be derived directly from the discrete Energy defined by the Hodge*. The only ambiguity in such a derivation is the definition of the discrete Hodge*.

The discrete Hodge* is also closely related to **interpolation**. Given a large finite set of values on one mesh, the Hodge* describes how to interpolate the underlying function and determine the finite set of values on a related, but different, mesh (the dual). The interpolation problem is of course closely related to **basis functions** and how one assumes the solution varies between the data values. Interpolation invariably must assume (explicitly or implicitly) the shape of the function at points between the known data values.

All four viewpoints of the Hodge* (three shown in bold above and the dual/primal mesh transfer in the first paragraph) are essentially equivalent. It is clear that the discrete Hodge* matrix should be positive definite (though not necessarily symmetric) so that it is invertible and defines a reasonable norm (inner-product) or energy. The interpolation interpretation and basis functions allow one to address the issue of the order of accuracy of a discrete Hodge*. All errors in a well developed numerical method enter via the Hodge* operators (the discrete calculus is exact). Therefore the discrete Hodge* accuracy defines the accuracy of the entire PDE solution. For example if a 0-form discrete Hodge* at the primary mesh vertices, $*^0$, takes in a vector of 1's (at the mesh vertices) and produces a vector with components equal to the dual cell volumes then this Hodge* is an exact interpolant between meshes for piecewise constant functions and therefore the method is 1st order accurate. The same method can be second-order accurate if the discrete 0-form values (at nodes) are located at the center of gravity (centroid) of the discrete 3-form dual volumes.

It seems likely that the various discrete Hodge* matrices perform better if they are internally consistent with each other in terms of the assumptions they make about the interpolation basis functions. For example, in the functional framework, if you assume polynomial interpolation for constructing discrete 0-form Hodge* matrices on the primal mesh, then you should probably assume H(curl) polynomials for 1-forms, H(div) polynomials for 2-forms and L2 polynomials for 3-forms on the primal mesh when trying to construct discrete Hodge* operations. Explicit polynomial basis functions (like the ones described above) are very common in Finite Element Methods. Other methods use other internally consistent basis functions (such as Fourier Spectral Methods), or implicit basis functions (such as in Finite Volume methods).

In 2D the discrete $*^0$ operation takes discrete 0-forms (point values) to discrete 2-forms (cell average values). For example a discrete $*^0$ takes mesh vertex values to Voronoi cell values, or cell center values to cell average values. The other discrete Hodge* operation in 2D takes 1-forms (edge values) to 1-forms (dual edge values). So edges between cell centers (i.e. dual edges or edges of the Voronoi dual) are taken to the cell faces (which are really the edges of the primal triangular mesh). Remember, edges in both the primal or dual mesh do not need to be lines. They can be kinked (for median dual meshes) or curved.

In 4D, there are 3 unique discrete Hodge* operations. Point values at a fixed time (0-forms) go to space–time averages (4-forms). Edge values and point values averaged over the time interval (1-forms) are transformed into face normal values integrated over the time interval and volumes averages at a fixed time (3-forms). Finally, face values at a fixed time and edge values integrated over time (2-forms) on the primal mesh go to edge values integrated over time and face values at a fixed time (on the dual space–time mesh).

In the continuous case, the Hodge* operation is almost an identity operation that is used to keep orientations correctly aligned. It also alerts the user to the fact that the resulting form is a pseudo-form (or outer oriented form, or twisted form). There seems to be no definitive terminology. In 3D applying the Hodge* twice is the identity operation for any continuous Hodge*). However in general $** = (-1)^{k(N-k)}$ where k is the dimension of the form being acted on and N is the dimension of the space (and assuming a Riemannian metric). For a pseudo-Riemannian metric like the 4D special relativity space–time Minkowski metric, this is multiplied by -1.

In 2D ($N = 2$), the discrete Hodge $*^1$ takes 1-forms on the primal mesh to 1-forms on the dual mesh. This is a rotation of the vector component of interest by 90 degrees clockwise. Performing the $*^1$ operation again actually rotates again by 90 degrees clockwise and transfers the result back to the primary mesh. But this has reversed the direction of all the original 1-form values. In mathematical terms (and cavalierly representing a 2D 1-form by its vector proxy) this is

$$*^1 \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} y \\ -x \end{pmatrix} \tag{14}$$

and so

$$*^1 *^1 \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -x \\ -y \end{pmatrix} = -\begin{pmatrix} x \\ y \end{pmatrix}$$

In 4D, both the $*^1$ and $*^3$ Hodge operations cause a sign change when applied twice. So for example in 3D vector notation

$$*^1 \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} p \\ -\mathbf{u} \end{pmatrix} \tag{15a}$$

and

$$*^3 \begin{pmatrix} q \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} \mathbf{v} \\ -q \end{pmatrix} \tag{15b}$$

Note that the 4D 2-form Hodge (in 3D vector notation) does not change sign but does flip position, so

$$*^2 \begin{pmatrix} \mathbf{f}_a \\ \mathbf{f}_b \end{pmatrix} = \begin{pmatrix} \mathbf{g}_b \\ \mathbf{g}_a \end{pmatrix} \tag{15c}$$

The discrete matrix versions of the Hodge* operations should have similar properties.

7. Lie derivative (advection)

The Lie derivative is closely associated with advection. Advection involves both a derivative and a vector field which transports the quantity of interest. In exterior calculus, the Lie derivative $L_{\mathbf{X}}$ of a form a is written as

$$L_{\mathbf{X}} a = i_{\mathbf{X}} \mathbf{d} a + \mathbf{d} i_{\mathbf{X}} a \tag{16}$$

(also called Cartan's identity). Here the operator $i_{\mathbf{X}}$ can be interpreted (translated into vector notation) is a vector multiplication and a demotion of the form by one dimension. Demotion of the form does not matter much for the vector proxies but it is still useful to keep track of the corresponding forms dimension. The operator $i_{\mathbf{X}}$ is called the *interior* product. Note that this is different from an *inner* product. For discrete forms, the demotion required by the inner product requires an interpolation (like a Hodge*) and this is therefore an operation that can induce error in the solution.

In 3D the Lie derivative translates into the following classical vector expressions (depending on the starting form).

$$0\text{-form } L_{\mathbf{X}} a^0 \Rightarrow Ls = \mathbf{x} \cdot \nabla s + 0 \tag{17a}$$

$$1\text{-form } L_{\mathbf{X}} a^1 \Rightarrow L\mathbf{v} = -\mathbf{x} \times (\nabla \times \mathbf{v}) + \nabla(\mathbf{x} \cdot \mathbf{v}) \tag{17b}$$

$$2\text{-form } L_{\mathbf{X}} a^2 \Rightarrow L\mathbf{w} = \mathbf{x}(\nabla \cdot \mathbf{w}) - \nabla \times (\mathbf{x} \times \mathbf{w}) \tag{17c}$$

$$3\text{-form } L_{\mathbf{X}} a^3 \Rightarrow Lr = 0 + \nabla \cdot (\mathbf{x} r) \tag{17d}$$

were $\mathbf{x}$ is the vector field that is advecting things (scalars or vectors) about. Our vectors and scalars are not labeled directly as forms, but since we know that they are proxies for forms, we know what (vector proxy) version of the exterior derivative operator ($\mathbf{d}$) to apply in each case. Notice that the interior product $i_{\mathbf{X}}$ has the translation 0, $\mathbf{x}\cdot$, $-\mathbf{x}\times$, $\mathbf{x}()$ when operating on the vector proxies for 0-forms, 1-forms, 2-forms, and 3-forms respectively. This is the reverse of the classical multiply order (or the classic de Rham complex). Note that the result of the Lie derivative is a form of the same dimension.

As with all the operators in exterior calculus, the interior product changes for every form it operates on. It can be written in terms of more fundamental operators (the Hodge* and multiply operators) and a 1-form advection velocity as, $i_{\mathbf{X}} = *x^1 \wedge *$. The sign change in the multiply operations above, obey the classic Hodge star relation $*^{N+1-k} *^k = (-1)^{k(N+1-k)} = (-1)^{kN+1}$ where k is the dimension of the input form. Also note that $i_{\mathbf{X}} i_{\mathbf{X}} = 0$. This is the equivalent of the 3D identities $\mathbf{x} \times (\mathbf{x} s) = 0$ for a 3-form (s is the scalar proxy) and $\mathbf{x} \cdot (-\mathbf{x} \times \mathbf{v}) = 0$ for a 2-form (where $\mathbf{v}$ is the vector proxy for the 2-form).

In 3D the 0-form Lie derivative is the classic passive scalar advection equation.

Expansion of the 1-form version of the Lie derivative gives,

$$\begin{aligned} -\mathbf{x} \times (\nabla \times \mathbf{v}) + \nabla(\mathbf{x} \cdot \mathbf{v}) &= -\varepsilon_{sti} x_t \varepsilon_{ijk} v_{k,j} + (x_k v_k)_{,s} \\ &= -(\delta_{sj}\delta_{tk} - \delta_{sk}\delta_{tj}) x_t v_{k,j} + (x_k v_k)_{,s} = -x_k v_{k,s} + x_k v_{s,k} + (x_k v_k)_{,s} = x_k v_{s,k} + x_{k,s} v_k \end{aligned}$$

which is also the gradient of the 0-form transport equation (passive scalar equation) with $\mathbf{v} = \nabla s$. This is true in general; the Lie derivative commutes with differentiation ($\mathbf{d} L_{\mathbf{X}} = L_{\mathbf{X}} \mathbf{d}$). Note that this 1-form Lie derivative is also the equation for the transport of a passive normal vector associated with an infinitesimally small area which is imbedded in the flow.

Expansion of the 2-form version of the Lie derivative,

$$\begin{aligned} \mathbf{x}(\nabla \cdot \mathbf{w}) + \nabla \times (-\mathbf{x} \times \mathbf{w}) &= x_s w_{k,k} - \varepsilon_{sti} \varepsilon_{ijk} (x_j w_k)_{,t} \\ &= x_s w_{k,k} - (\delta_{sj}\delta_{tk} - \delta_{sk}\delta_{tj})(x_j w_k)_{,t} = x_s w_{k,k} - (x_s w_k)_{,k} + (x_k w_s)_{,k} = (x_k w_s)_{,k} - x_{s,k} w_k \end{aligned}$$

gives the same term as what is found for vorticity advection (or a magnetic flux). It is also the same equation that a passive infinitesimal line imbedded in the flow would obey. This form of the Lie-derivative is also identical to the curl of the 1-form Lie derivative with $\mathbf{w} = \nabla \times \mathbf{v}$.

The 3-form divergence is the form classically seen for conservative Navier–Stokes equations, or for the Reynolds transport theorem. This is because mass and energy are 3-form quantities. The divergence of the 2-form Lie derivative gives the 3-form Lie derivative, with $r = \nabla \cdot \mathbf{w}$. This means that if the vorticity or magnetic flux (2-form variables) start divergence-free, they are assured to remain that way, if the discrete Lie derivative is defined correctly (see Refs. [23–25]).

On a Lagrangian mesh (that moves with the flow), the value of the discrete 1-form may change in time in two ways. Either the underlying generating vector field is varying with time, or the edge itself is moving and changing its orientation as it passively advects. The Lie derivative is constructed so that both these effects are accounted for. For a passive vector field which is in essence 'painted' on the material as it advects and a Lagrangian mesh (which also advects with the material), the Lie derivative is 0.

In 2D the 3-form version of the Lie derivative does not exist. In addition, the 2-form Lie derivative becomes $L_{\mathbf{x}}a^2 \Rightarrow Lw_3 = 0 - \nabla \times_{2D} (\mathbf{x} \times w_3)$. Because the 2-form w_3 is perpendicular to the advection velocity (which lies in the 2D plane), the cross product rotates $\mathbf{x}$ by 90 degrees clockwise. The curl of a 90 degrees counter-clockwise rotated vector field is the same as the 2D divergence, so this is equal to $L_{\mathbf{x}}a^2 \Rightarrow Lw_3 = \nabla \cdot_{2D} (\mathbf{x}w_3)$, which looks like the 3-form version of the Lie derivative in 3D. Similarly, in 2D the 1-form version of the Lie derivative can also be simplified,

$$L_{\mathbf{x}}a^1 \Rightarrow L\mathbf{v} = -\mathbf{x} \times (\nabla \times \mathbf{v}) + \nabla(\mathbf{x} \cdot \mathbf{v}) = R_{90}(\mathbf{x})(\nabla \times_{2D} \mathbf{v}) + \nabla(\mathbf{x} \cdot \mathbf{v})$$

where $R_{90}(\mathbf{x})$ is a 90 degrees rotation of the $\mathbf{x}$ vector.

In 4D, the Lie derivative becomes,

$$0\text{-form } L_{\mathbf{x}}a^0 \Rightarrow Ls = *^4\mathbf{x} \otimes *^1(\nabla s) + 0 \tag{18a}$$

$$1\text{-form } L_{\mathbf{x}}a^1 \Rightarrow L\mathbf{v} = *^3\mathbf{x} \circ *^2(\nabla \times_{4D} \mathbf{v}) + \nabla\left(*^4\mathbf{x} \otimes *^1\mathbf{v}\right) \tag{18b}$$

$$2\text{-form } L_{\mathbf{x}}a^2 \Rightarrow L\mathbf{f} = *^2\mathbf{x} \times_{4D} *^3(\nabla \circ \mathbf{f}) + \nabla \times \left(*^3\mathbf{x} \circ *^2\mathbf{f}\right) \tag{18c}$$

$$3\text{-form } L_{\mathbf{x}}a^3 \Rightarrow L\mathbf{w} = *^1\mathbf{x} *^4 (\nabla \otimes \mathbf{w}) + \nabla \circ \left(*^2\mathbf{x} \times_{4D} *^3\mathbf{w}\right) \tag{18d}$$

$$4\text{-form } L_{\mathbf{x}}a^4 \Rightarrow Lr = 0 + \nabla \otimes \left(*^1\mathbf{x} *^4 r\right) \tag{18e}$$

The flipping and sign changes caused by the $*$ operators are important so (unlike the 3D case) the $*$ operators are explicitly included above. This is an adoption of the Hodge* notation into vector calculus. At the continuous level it mostly flips signs, but at the discrete level it reminds us that a mesh transfer is necessary. One reason differential forms can be a 'better' representation for a physical process is that this Hodge* is not repressed (as it is in notation provided by multivariable calculus).

Assuming time as the 4th dimension, and setting $\mathbf{x} = (\mathbf{c}, 1)$ with $\mathbf{c}$ the velocity field, gives the classic advection term for the 0-form Lie derivative,

$$L_{\mathbf{x}}a^0 \Rightarrow Ls = \frac{\partial s}{\partial t} + \mathbf{c} \cdot \nabla s. \tag{19a}$$

The 1-form becomes

$$L_{\mathbf{x}}a^1 \Rightarrow L\begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} \frac{\partial \mathbf{u}}{\partial t} - \nabla p - \mathbf{c} \times \nabla \times \mathbf{u} + \nabla(p + \mathbf{c} \cdot \mathbf{u}) \\ -\mathbf{c} \cdot \frac{\partial \mathbf{u}}{\partial t} + \mathbf{c} \cdot \nabla p + \frac{\partial}{\partial t}(p + \mathbf{c} \cdot \mathbf{u}) \end{pmatrix} = \begin{pmatrix} \frac{\partial \mathbf{u}}{\partial t} - \mathbf{c} \times \nabla \times \mathbf{u} + \nabla(\mathbf{c} \cdot \mathbf{u}) \\ \frac{\partial p}{\partial t} + \mathbf{c} \cdot \nabla p + \mathbf{u} \cdot \frac{\partial \mathbf{c}}{\partial t} \end{pmatrix} \tag{19b}$$

The first part is the same as the 3D Lie derivative for 1-forms but with the time derivative included, and the second part is the 4D 0-form Lie derivative plus an extra term. This is again identical to the gradient of the 4D 0-form Lie derivative with $(\mathbf{u}, p) = (\nabla s, \frac{\partial s}{\partial t})$.

Similarly, the 2-form becomes

$$\begin{aligned} L_{\mathbf{x}}f^2 \Rightarrow L\begin{pmatrix} \mathbf{f}_a \\ \mathbf{f}_b \end{pmatrix} &= \begin{pmatrix} \frac{\partial \mathbf{f}_a}{\partial t} - \nabla \times \mathbf{f}_b + \mathbf{c} \cdot \nabla \cdot \mathbf{f}_a + \nabla \times (\mathbf{f}_b - \mathbf{c} \times \mathbf{f}_a) \\ \mathbf{c} \times (\frac{\partial \mathbf{f}_a}{\partial t} - \nabla \times \mathbf{f}_b) + \frac{\partial}{\partial t}(\mathbf{f}_b - \mathbf{c} \times \mathbf{f}_a) + \nabla(\mathbf{c} \cdot \mathbf{f}_b) \end{pmatrix} \\ &= \begin{pmatrix} \frac{\partial \mathbf{f}_a}{\partial t} + \mathbf{c}\nabla \cdot \mathbf{f}_a - \nabla \times (\mathbf{c} \times \mathbf{f}_a) \\ \frac{\partial \mathbf{f}_b}{\partial t} - \mathbf{c} \times (\nabla \times \mathbf{f}_b) + \nabla(\mathbf{c} \cdot \mathbf{f}_b) + \mathbf{f}_a \times \frac{\partial \mathbf{c}}{\partial t} \end{pmatrix} \end{aligned} \tag{19c}$$

which has a similar structure to the 1-form Lie derivative.

The 3-form Lie derivative can be derived similarly. The 4-form Lie derivatives produces

$$L_{\mathbf{x}}a^4 \Rightarrow Lr = \frac{\partial r}{\partial t} + \nabla \cdot (\mathbf{c}r) \tag{19d}$$

Proposals about how to properly construct discrete Lie derivatives are discussed in Palha et al. [24] and Mullen et al. [25].

8. Mesh and its dual

At the infinitesimal level the distinction between a mesh and its dual mesh becomes less obvious as both meshes shrink to essentially the same locations. Nevertheless, the distinction is still important to differential forms, and it manifests itself in various terminologies. Texts sometimes refer to straight and twisted forms. Another common description is interior and exterior orientation for the forms. Sometimes this aspect is referred to as vectors and co-vectors. All these distinctions are

necessary for infinitesimal forms where the mesh and its dual shrink into obscurity. At some basic level this type of pairing is fundamental to mathematics (Poincaré duality) and physics.

For finite forms, the most practical distinction remains the mesh and its dual. An example is in order. In electromagnetism, the electric field vector (**E**) is usually a 1-form. It can be represented by its components integrated along the mesh edges. Similarly, the magnetic flux or magnetic induction (**B**) is a 2-form and can represented by its normal component on the mesh faces. Both of these variables are typically referred to as "inner" oriented (or straight forms). This is essentially because they both reside on the primary mesh. One of Maxwell's equations $\frac{\partial \mathbf{B}}{\partial t} + \nabla \times \mathbf{E} = 0$ can be represented exactly with these forms on the primary mesh. However, the other of Maxwell's equations, $\frac{\partial \mathbf{D}}{\partial t} - \nabla \times \mathbf{H} + \mathbf{J} = 0$ uses the electric flux **D** (a 2-form like all fluxes) and magnetic field **H** (a 1-form, like the electric field). But these are twisted forms, or "outer" oriented. They lie on the **dual** mesh. The seemingly simple material relationships $\mathbf{D} = \varepsilon \mathbf{E}$ and $\mathbf{H} = \mu \mathbf{B}$ must interpolate between the primary and dual meshes (and between 1-forms and 2-forms). These material relationships therefore contain the discrete Hodge* operators (and all the numerical errors).

The key idea of a mesh and its dual remains intact when developing higher order methods. A higher order finite volume method based on discrete exterior calculus (or discrete calculus) is described in [26] and the higher-order finite element approaches using discrete exterior calculus ideas are common [5,15,18].

Again, the tetrahedral mesh, or mesh generated by the generator program can be the primary (or straight, or inner oriented) mesh. But it is also just fine if this tetrahedral mesh acts as the dual (or twisted or outer oriented) mesh. The finite element use of the weak form of the PDE is an implicit way of defining a (smeared or averaged) dual mesh [21]. Finite elements put the emphasis on the definition of inner products rather than on a dual mesh, but we know that both ideas are directly related to the discrete Hodge* and are essentially equivalent.

9. Other concepts

This section mentions some other concepts that the reader might see in the literature, but which may be of less importance to the numerical solution of PDEs.

For example, the flat operator, $\flat$, is the formal operator that takes a regular vector (proxy) to its 1-form. And the sharp operator, $\sharp$, takes a 1-form to its vector proxy. We could have used these operators liberally throughout the text, and they would add mathematical rigor to the text. But we felt that they add little in the way of additional insight for an introductory text. It is enough to know that there is a one-to-one effective correspondence between a vector field and a 1-form. Formally one could write $[*\mathbf{d}\mathbf{v}^{\flat})]^{\sharp} = \nabla \times \mathbf{v}$ for 3D. This takes a vector field, **v**, and converts it to a 1-form with the flat operation. It then takes the exterior derivative, which for a 1-form in 3D is (effectively) a curl, which results in a 2-form. The $*$ operation takes a 2-form to a 1-form. And finally the sharp takes that 1-form and converts it back to a vector. Our text above (and below) uses $\mathbf{d}v^1 \Rightarrow \nabla \times \mathbf{v}$, which states the two operations are (effectively) equivalent. The flat and sharp conversions are understood to be entirely a formality when attempting to do a translation.

Exterior calculus sometimes talks about the co-differential operator $\mathbf{d}^*$ as if it was a separate operation. This is really not a primal operation in itself but is made up of the differential operation **d** and the Hodge-star operation (or Hodge*) which was discussed in Section 6. The co-differential takes the form down one dimension (rather than up one dimension, which is what **d** does). It can be written as $\mathbf{d}^* = (-1)^{(Nk+N+1)} * \mathbf{d} *$ (on a Riemannian metric, like Euclidean space) where k is the dimension of the form being operated on. For numerical methods it is useful to think about this operation as the differential operation on the *dual* mesh.

Frequently a notational and formal distinction is made between the differential operation **d**, and its finite equivalent, called the co-boundary operator δ. But be aware that the notational convention is not always the same and that this symbol is also used for the co-differential described just above. This work cavalierly makes no formal distinction between finite and infinitesimal differentiation (or its notation).

A cochain associates to every cell (or face, or edge, or node) in the mesh a number. When reading about a cochain it is sufficient to think of a set of discrete mesh unknowns. They are (for us) the same thing as discrete forms. A set of values, with one value for each of the edges of a mesh are a 1-cochain. Similarly the set of all cell average unknowns in 3D is a 3-cochain (or a discrete 3-form). A chain is effectively a set of the edges themselves, taking values of $+1, -1, 0$. Taking a discrete curl (or gradient or divergence) operation is the same as multiplying the elements of a chain with the equivalent elements of a cochain and summing. Formally the cochain is the function that produces the discrete values, not the values themselves. In practice, it is fine to comingle the function with its result when reading "cochain".

The trace is the act of projecting down a dimension. For example, in 3D one sometimes needs boundary conditions on a 2D surface. So, $tr(a^0) \Rightarrow a^0$, $tr(a^1) \Rightarrow \mathbf{n} \times (\mathbf{a} \times \mathbf{n})$, $tr(a^2) \Rightarrow \mathbf{n} \cdot \mathbf{a}$, and $tr(a^3) \Rightarrow 0$. The trace of a 1-form lies in the plane of the surface, the 2-form lies normal to the surface, and a 3-form cannot be projected into a lower dimension. One only really sees the trace operation for PDEs when there is a discussion of the implementation of boundary conditions. The trace is frequently implemented at the discrete level with 'ghost' cells, or 'very thin' face cells.

10. Example

Here it is shown how differential forms can be used to enable/enhance the discretization of PDEs. A very classic PDE written in terms of differential forms is $\mathbf{d} * \mathbf{d} a^k = b^{N-k}$ where b^{N-k} is a known source and a^k is the unknown solution one

wishes to find. An index k on the unknown solution is explicitly stated here to remind the reader that $\mathbf{d}$ means different things (functionally) depending on what kind of form it is operating on and what the dimension of the space is. In what follows we will assume a 3D space.

10.1. 0-form

For a 0-form unknown, a^0, this equation is analogous (equivalent) to the Poisson equation and the operator on the left-hand side of this equation is analogous to the scalar Laplacian, $\nabla \cdot (\nabla a) = \nabla^2 a = b$. The operator $\mathbf{d}$ operating on a 0-form is similar to taking the gradient of a scalar. It produces a 1-form. If the unknowns for the discrete 0-form are located at mesh nodes (vertices), then the discrete interpretation is that the discrete 1-form, $\mathbf{d}a^0 = \int_{n1}^{n2} \nabla a \cdot d\mathbf{l} = a|_{n2} - a|_{n1}$, is the exact integral of the gradient along the mesh edges. Placing the unknowns at the mesh nodes (vertices) is equivalent to the unknowns for a classic lowest order Finite Element method.

Proceeding to the next operation, The Hodge* operation changes (reflects) the dimension of the form, so in this case it takes the resultant 1-form, $g^1 = \mathbf{d}a^0$, and turns it into a 2-form (3 for 3D space minus 1 for the 1-form is $3 - 1 = 2$). In the discrete sense, the discrete Hodge* must take a discrete 1-form (along the mesh edges) and produce a discrete 2-form (on the dual mesh faces). Note that for most dual meshes there is a one-to-one matching between the number of mesh edges and the number of dual mesh faces (leading to a square and invertible discrete Hodge* matrix). In the simple case of a Voronoi dual mesh, the tetrahedral mesh edges are perpendicular to the Voronoi dual cell faces and one possible discrete Hodge* matrix is a diagonal matrix, with each diagonal entry being $A_{\tilde{f}}/L_e$, where $A_{\tilde{f}}$ is the area of the dual face (tilde indicates the dual mesh), and L_e is the edge length. This is one possibility, but the discrete Hodge* is not unique. The choice of the Hodge* is effectively an interpolation choice that defines the method and the method's accuracy.

For multivariable calculus and PDE's, there is no equivalent to the Hodge* operation in the equation, because in the infinitesimal limit it becomes trivial. This is one disadvantage of discretizing PDEs directly. When pulling back to the discrete case, one ought to remember that the Hodge* exists and is important, otherwise the approximation errors are forced into the calculus operations. Inexact discrete representations of the calculus often violate important mathematical and physical properties of the original PDE. The beauty of the differential forms is that they make it clear that the calculus part of the problem, $\mathbf{d}$, is easy to discretize exactly, and that all errors should emanate from the Hodge* operation.

Finally, the left-most $\mathbf{d}$ is now operating on a 2-form and therefore must produce a 3 form, which equals the Poisson equation source term, $b^{N-0} = b^3$ (in 3D space $N = 3$). This means that this left most $\mathbf{d}$ is analogous to a divergence. In the discrete case this means the source is integrated over the dual cell volumes, and we have one discrete equation for each dual volume. For most meshes there is a one-to-one matching in the number of dual cells and primary mesh nodes, so the resulting discrete system is square and invertible for the solution (given that the Hodge* matrix is reasonably well behaved).

10.2. 1-form

With a different dimension for the unknown, a different PDE is obtained. In the case that the unknown is a 1-form, $\mathbf{d} * \mathbf{d}a^1 = b^2$ is similar (or physically equivalent) to the vector Poisson equation, $\nabla \times \nabla \times \mathbf{a} = \mathbf{b}$ where $\mathbf{a}$ and $\mathbf{b}$ are vectors. An example of this equation is finding the streamfunction from the vorticity in fluid dynamics. Other examples occur in magnetostatics. Usually this equation has a large null space for the unknown which is eliminated by specifying a gauge on the unknown. Only methods with a direct connection to differential forms, such as Nedelec elements or unstructured staggered mesh methods, discretize this equation so that the null space is properly retained (see the eigenmode example in [27]).

Note that for the 1-form unknown the right-most $\mathbf{d}$ operator is analogous to a curl and produces a 2-form result. Discretely this operator takes in values along the edges of the mesh and produces values on the faces of the mesh (with each mesh face receiving contributions from the edges bounding that face). The result of the right-most discrete $\mathbf{d}$ operator is $\mathbf{d}a^1 = \int_{face} (\nabla \times \mathbf{a}) \cdot \mathbf{n}\, dA$ which is the face integral of the normal component of the curl. The Hodge* operation (for this equation) then takes this resulting 2-form back to a 1-form (because $3 - 2 = 1$). In the discrete case this Hodge* takes discrete values on the faces of the mesh and constructs an approximation for the values along the edges of the dual mesh. The edges of the dual mesh are lines (straight, segmented, or curved) that connect the mesh cell centers and pierce through the mesh faces. The left-most $\mathbf{d}$ operator therefore also produces a 2-form and is again analogous to a curl (which is always the $\mathbf{d}$ operator that takes in 1-forms and produces 2-forms, in 3D). But in the discrete case, this discrete curl matrix is not the same as the first curl matrix. This second (left most) curl takes in dual edge values and produces values on the dual faces (the first discrete curl operation acted on objects on the primary mesh). It turns out that on most meshes (where the primal and dual meshes have a one-to-one geometric relationship) that this second (left-most) curl matrix is the transpose of the first (right-most) curl matrix.

The curl–curl equation, as it is written in multivariable calculus hides the fact that the two curl operations are actually different. We might be sorely tempted to discretize them in the same way, and that would be wrong. The notation of differential forms had a $*$ operation which reminds us that the two $\mathbf{d}$ operators are operating on different things (and are therefore different at the basic implementation level).

10.3. 2-form

When the unknown is a 2-form the analogous PDE becomes $\mathbf{d} * \mathbf{d} a^2 = b^1 \Rightarrow \nabla(\nabla \cdot \mathbf{a}) = \mathbf{b}$ where $\mathbf{a}$ and $\mathbf{b}$ are again vectors. This PDE has a very large null space, the solution $\mathbf{a}$ has all solenoidal functions $\mathbf{a} = \nabla \times \mathbf{c}$ as homogeneous solutions to this PDE. The $\mathbf{d}$ on the 2-form is analogous to a divergence, and produces a 3-form. Discretely it takes face integral unknowns and produces a cell integral result. The Hodge* then reflects this result, producing a 0-form. In the discrete case, this discrete Hodge* takes the cell average result and produces a point value quantity at the dual mesh nodes (which are the same as the cell centers). The left-most $\mathbf{d}$ takes this 0-form and is analogous to a gradient, producing a 1-form. In the discrete case, this gradient matrix is different from the gradient matrix found earlier (for the 0-form unknown, $\nabla \cdot \nabla a = b$ equation discretization). This gradient matrix is on the dual mesh. It takes cell center (dual node) values and produces discrete 1-form values along the dual edges connecting those cell centers. This matrix is often the transpose of the divergence matrix (discrete version of the right-most $\mathbf{d}$ operator).

10.4. Dual mesh based methods

A 3-form unknown for the differential form version of the Poisson equation is not possible (in 3D), because a 3-form cannot be promoted by $\mathbf{d}$ to a higher dimension (in 3D). There are, however, another complete set of discretization methods that are possible for each of the 3 cases described above. These solution methods arise by simply flipping the labeling of the dual and the primary meshes. For example, it is possible to treat the 0-form case, $\mathbf{d} * \mathbf{d} a^0 = b^3 \Rightarrow \nabla \cdot (\nabla a) = b$, with the 0-form unknowns at the cell centers. For example, this would be common in some low-order Finite Volume methods or in Discontinuous Galerkin methods. The cell centers are the nodes of the dual mesh. So the first $\mathbf{d}$ now produces an exact integral of the gradient along the edges connecting the cell centers. The Hodge* interpolates those (dual) edge values to the cell faces (which is typically a one-to-one interpolation). The left-most $\mathbf{d}$ then takes an exact divergence (just like a finite volume method does) using the provide face fluxes. Classic Finite Volume methods are not mimetic when the Hodge* and the right-most $\mathbf{d}$ are discretized together (which is usually the case). The resulting matrix cannot then be separated into the two critical parts (discrete Hodge* and exact discrete $\mathbf{d}$). However, a mimetic Finite Volume method (in which the discrete Hodge* is separate from the exact discrete $\mathbf{d}$) is described in Ref. [11], and a higher order version is presented in Ref. [26].

11. Summary

A recurring observation in this text is that differential geometry uses a single symbol to represent many different operations. Sometimes (such as for the derivative and Hodge*) there are N functional operations per symbol, but there can also be order N^2 operations per symbol (such as for the wedge product). Moreover, the operations that are represented by each symbol are not only a set of operations, but a set of operations that changes with the dimension being operated in. The set of operations does not just expand as we go from 2D to 3D (or 4D) it sometimes changes entirely. So a 1-form times (wedge product) another 1-form is different in 2D and in 3D. The first produces a scalar result and the second a vector result, although both results are 2-forms.

Using a single symbol to represent many things and in any dimension is very elegant mathematically. We use the same idea for the addition ($+$) symbol to add many different kinds of things: integers, real numbers, complex numbers, vectors, tensors, etc. But using the same symbol can be confusing when first introducing an idea. The actual mechanics of addition is different in each of the examples listed above, and the mechanics of each type of addition needed to be shown to the reader at least once. This paper looks at the actual mechanics of differential geometry and exterior calculus. It attempts to show (at least once) what each operator really implies at a purely practical implementation level. It also looks at the discrete or finite versions of all these operations. These sorts of prosaic translations are necessary in order to actually use differential forms and exterior calculus to solve PDEs numerically with a computer program (as shown in Section 10).

It is clear that at the discrete level, calculus is intimately related to topology and dimension. Topology is inherent in Gauss' and Stokes' theorems and even the Fundamental theorem of calculus. In addition, the ideas of differential geometry make it clear that the calculus and physics of PDEs can be represented exactly at the finite level (using appropriate defined integral unknowns, which are discrete forms). Forms and exterior calculus are valuable because they make the connection between geometry and calculus explicit in the notation. When discretizing PDEs all approximation can (and probably should) appear only in the Hodge* operation (or Hodge* derivable operations such as the interior product or Lie derivative). It is not surprising that in multivariable calculus the temptation is to discretize the div, grad, and curl operators since the Hodge* (and its action) is suppressed in that notation.

Interestingly, the Hodge* only appears in material constitutive relations. These material relationships are invariably engineering simplifications and approximations of the real material anyway. In this way all errors (physical and numerical) appear in the same place in the equation system. When using discrete forms the physics and mathematics of the PDE systems are never violated because the calculus ($\mathbf{d}$ operation) can be discretized exactly.

Acknowledgements

This work was supported, in part, by the National Science Foundation CBET program.

Appendix A

A possible reference for more information on differential forms and their application to physics is Frankel [28]. YouTube videos by David Metzler are also a useful introduction. We present a brief overview of forms here.

We mentioned earlier that forms are the "things inside integrals". So a 1-form is $\alpha^1 = \mathbf{F}\cdot d\mathbf{l} = F_1(\mathbf{x})\,dx_1 + F_2(\mathbf{x})\,dx_2 + F_3(\mathbf{x})\,dx_3$. This is only sort of true. It is the inspiration for forms, but not really their definition.

In particular, the "things inside integrals" are small (infinitesimal) so that we can perform an infinite sum (integration) of them and get a finite number. But forms are typically not thought of as small. The distinction/confusion may emanate from the integral sign which actually means two different things depending on the context. $\int_{n1}^{n2}\mathbf{F}\cdot d\mathbf{l}$ means "sum a bunch of tiny numbers". But $\int_{n1}^{n2}\alpha^1$ means something more like "count the number of level set crossings" (where levels go up in unit increments) along the curve. So in this case, the form does not have to be small, it just has to be a measure of the number of level sets. In practice, only some forms (exact forms) can be represented by level sets, but the idea is still a useful first description.

When a math book says $\alpha^1 = F_1(\mathbf{x})\,dx_1 + F_2(\mathbf{x})\,dx_2 + F_3(\mathbf{x})\,dx_3$ they do not mean to imply that the dx_1 are the small things you and I would immediately assume they are. They actually treat dx_1 very much like it was a basis vector (not even a single small number). In fact, in our notation they really mean to define a 1-form as $\alpha^1 = F_1(\mathbf{x})\,\mathbf{d}x + F_2(\mathbf{x})\,\mathbf{d}y + F_3(\mathbf{x})\,\mathbf{d}z$ where $\mathbf{d}$ is the exterior derivative and x, y, z are functions (0-forms). This is essentially defining a 1-form as any linear combination of 3 special/simple 1-forms $(\mathbf{d}x, \mathbf{d}y, \mathbf{d}z)$. These three special 1-forms are derived by taking the exterior derivative of three very simple functions $f(\mathbf{x}) = x$, $f(\mathbf{x}) = y$, $f(\mathbf{x}) = z$.

If we think of the $(\mathbf{d}x, \mathbf{d}y, \mathbf{d}z)$ as unit basis co-vectors, much of the literature on forms makes a better connection with regular vectors. The distinction between a co-vector and a vector is unimportant for a Cartesian reference frame. If you don't like non-Cartesian frames, then just stay in a Cartesian frame when thinking about forms. One of the mathematical glories of forms are that they are independent of coordinate systems (and even metrics), and vectors are not. But insisting on that level of abstraction to maintain their purity doesn't help us explain them, so we will stay in a Cartesian frame and assume we can define basis functions for that frame.

For example, often a form is defined as a "linear functional on vectors". That is, a thing that takes a vector argument and produces a scalar result. This functional definition of a form appears in expressions like $\alpha^1(\mathbf{v}) = F_1 v_1 + F_2 v_2 + F_3 v_3$ (note the $(\mathbf{v})$ means "is a function of"). Initially it looks as if the "rule" is to throw out the $\mathbf{d}x, \mathbf{d}y, \mathbf{d}z$ and replace it with the vector component. But why? However, if we think of the $(\mathbf{d}x, \mathbf{d}y, \mathbf{d}z)$ as unit basis co-vectors then this "function" is really like a dot product, $\alpha^1(\mathbf{v}) = F_1\,\mathbf{d}x\cdot\mathbf{v} + F_2\,\mathbf{d}y\cdot\mathbf{v} + F_3\,\mathbf{d}z\cdot\mathbf{v} = (F_1\,\mathbf{d}x + F_2\,\mathbf{d}y + F_3\,\mathbf{d}z)\cdot\mathbf{v}$ and dot products take in a vector argument (and in this case have a second co-vector argument which is the 1-form) and produces a scalar result. For example, the simple 1-form $\alpha^1 = \mathbf{d}y$ has the result of extracting the second component of a vector $\alpha^1(\mathbf{v}) = (1)\,\mathbf{d}y(\mathbf{v}) = (1)\,\mathbf{d}y\cdot\mathbf{v} = v_2$, which is what a unit basis vector would do. Note that the components (or functions comprising) the differential form $\mathbf{d}y$ are $F_1 = 0, F_2 = 1, F_3 = 0$, which is also strongly reminiscent of a unit basis vector. Formally, the 1-form is related to a covariant vector and the vector it operates on is a contravariant vector, and the dot product of those two types of vectors is coordinate system independent. And really, even that interpretation is considered simplistic by mathematicians (because it uses a metric), but we will not delve deeper here.

The notion of a 1-form as a linear function on vectors is not very useful for a computer simulation. Computers require that we be able to store some "thing" in a memory slot. Functions are difficult to store. The discrete forms are the forms evaluated along some point, curve, area, volume in a mesh. We have supplied the thing that the forms act on (the mesh), so that we can store the result (a number) in the computer.

Consider the 1-form $\alpha^1 = F_1\,\mathbf{d}x + F_2\,\mathbf{d}y + F_3\,\mathbf{d}z$. In a different coordinate system the components of the 3-tuple $\mathbf{F}$ change, but the value produced by the 1-form α^1 does not. This is because the $\mathbf{d}x_i$ are also coordinate system dependent but in a way that cancels the changes in the vector $\mathbf{F}$. In discrete terms, $\int\mathbf{F}\cdot d\mathbf{l}$ along a line gives the same result no matter what the coordinate system even though the components of the vector $\mathbf{F}$ and the components of the vector defining the line, $d\mathbf{l}$, are different in different coordinate systems.

Also note that if the $F_i(\mathbf{x})$ are the result of a gradient of a function $F_i(\mathbf{x}) = \frac{\partial f}{\partial x_i}$ then the 1-form is the classic definition of a "differential", $\alpha^1 = \frac{\partial f}{\partial x_1}dx_1 + \frac{\partial f}{\partial x_2}dx_2 + \frac{\partial f}{\partial x_3}dx_3 = df$. But the notation has been generalized to $\alpha^1 = \frac{\partial f}{\partial x_1}\mathbf{d}x_1 + \frac{\partial f}{\partial x_2}\mathbf{d}x_2 + \frac{\partial f}{\partial x_3}\mathbf{d}x_3 = \mathbf{d}f$, so that the $\mathbf{d}x_i$ and the $\mathbf{d}f$ are no longer required to be "small" or infinitesimal.

When the 1-form associated with the coefficients $\mathbf{F}$ acts on a small line segment, $d\mathbf{l}$, then $\alpha^1(d\mathbf{l}) = F_1(\mathbf{x})\,dl_1 + F_2(\mathbf{x})dl_2 + F_3(\mathbf{x})dl_3 = \mathbf{F}\cdot d\mathbf{l}$ which is the thing that occurs in line integrals. But in general, the idea is to think of the form as acting on the entire curve, not a tiny section of it called $d\mathbf{l}$.

References

[1] F.E. Harlow, J.E. Welch, Numerical calculations of time dependent viscous incompressible flow of fluid with a free surface, Phys. Fluids 8 (12) (1965) 2182–2189.

[2] P.A. Raviart, J.M. Thomas, A Mixed Finite Element Method for Second Order Elliptic Problems, Springer Lecture Notes in Mathematics, vol. 606, Springer-Verlag, 1977, pp. 292–315.

[3] J.-C. Nedelec, Mixed finite elements in R^3, Numer. Math. 50 (1980) 315–341.

[4] J.B. Perot, Discrete conservation properties of unstructured mesh schemes, Annu. Rev. Fluid Mech. 43 (2011) 299–318.
[5] A. Bossavit, I. Mayergoyz, Edge elements for scattering problems, IEEE Trans. Magn. 25 (4) (1989) 2816–2821.
[6] W. Burke, Div, Grad, Curl Are Dead, book preprint, http://count.ucsc.edu/~rmont/papers/Burke_DivGradCurl.pdf, 1996.
[7] R. Hiptmair, Discrete Hodge operators: An algebraic perspective, Prog. Electromagn. Res. 32 (2001) 247–269.
[8] H. Flanders, Differential Forms: With Applications to the Physical Sciences, Academic Press, New York, 1963.
[9] E. Cartan, Sur certaines expressions différentielles et sur le probleme de Pfaff, Ann. Éc. Norm. 16 (1899) 239–332.
[10] M. Desbrun, A.N. Hirani, M. Leok, J.E. Marsden, Discrete Exterior Calculus, arXiv:math/0508341, 2005.
[11] J.B. Perot, V. Subramanian, Discrete calculus methods for diffusion, J. Comput. Phys. 224 (1) (2007) 59–81.
[12] E. Tonti, The reason for analogies between physical theories, Appl. Math. Model. 1 (1976) 37–50.
[13] J.B. Perot, D. Vidovic, P. Wesseling, Mimetic reconstruction of vectors, in: D.N. Arnold, P.B. Bochev, R.B. Lehoucq, R.A. Nicolaides, M. Shashkov (Eds.), Compatible Spatial Discretizations, in: IMA Volumes in Mathematics and its Applications, vol. 142, Springer, New York, 2006, pp. 173–188, http://www.ima.umn.edu/springer/description.html.
[14] M. Shashkov, B. Swartz, B. Wendroff, Local reconstruction of a vector field from its normal components on the faces of grid cells, J. Comput. Phys. 139 (1998) 406–409.
[15] G. Rodrigue, D. White, A vector finite element time-domain method for solving Maxwell equations on unstructured hexahedral grids, SIAM J. Sci. Comput. 23 (3) (2001) 683–706.
[16] J.B. Perot, V. Subramanian, A discrete calculus analysis of the Keller box scheme and a generalization of the method to arbitrary meshes, J. Comput. Phys. 226 (1) (2007) 494–508.
[17] J.B. Perot, Conservation properties of unstructured staggered mesh schemes, J. Comput. Phys. 159 (2000) 58–89.
[18] M. Costabel, M. Dauge, Computation of Resonance Frequencies for Maxwell Equations in Nonsmooth Domains, Lecture Notes in Computational Science and Engineering, vol. 31, Springer, 2003.
[19] X. Zhang, D. Schmidt, J.B. Perot, Accuracy and conservation properties of a three-dimensional unstructured staggered mesh scheme for fluid dynamics, J. Comput. Phys. 175 (2002) 764–791.
[20] T. Euler, Consistent discretization of Maxwell's equations on polyhedral grids, Dissertation, Technischen Universität Darmstadt, 2007.
[21] C. Mattiussi, An analysis of finite volume, finite element, and finite difference methods using some concepts from algebraic topology, J. Comput. Phys. 133 (1997) 289–309.
[22] J.B. Perot, R. Nallapati, A moving unstructured staggered mesh method for the simulation of incompressible free-surface flows, J. Comput. Phys. 184 (2003) 192–214.
[23] W. Chang, F. Giraldo, J.B. Perot, Analysis of an exact fractional step method, J. Comput. Phys. 179 (2002) 1–17.
[24] A. Palha, J. Kreeft, M. Gerritsma, Numerical solution of advection equations with the discretization of the Lie derivative, in: J.C.F. Pereira, A. Sequeira (Eds.), V European Conference on Computational Fluid Dynamics ECCOMAS CFD 2010, Lisbon, Portugal, June 2010, pp. 14–17.
[25] P. Mullen, A. McKenzie, D. Pavlov, L. Durant, Y. Tong, E. Kanso, J.E. Marsden, M. Desbrun, Discrete Lie advection of differential forms, Found. Comput. Math. 11 (2) (2011) 131–149, http://dx.doi.org/10.1007/s10208-010-9076-y.
[26] V. Subramanian, J.B. Perot, Higher-order mimetic methods for unstructured meshes, J. Comput. Phys. 219 (1) (2006) 68–85.
[27] D. Arnold, R. Falk, R. Winther, Finite element exterior calculus: from Hodge theory to numerical stability, Bull. Am. Math. Soc. 47 (2) (2010) 281–354.
[28] T. Frankel, The Geometry of Physics: An Introduction, Cambridge University Press, 1997.