Journal of Computational Physics 424 (2021) 109853

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A mimetic method for polygons

J. Blair Perot [a,*], Chris Chartrand [b,*]

[a] Theoretical and Computational Fluid Dynamics Laboratory, University of Massachusetts, Amherst, MA, 01003, United States of America
[b] Sandia National Laboratory, Albuquerque, NM, United States of America

ARTICLE INFO

Article history:
Received 16 January 2020
Received in revised form 13 September 2020
Accepted 14 September 2020
Available online 22 September 2020

Keywords:
Mimetic
Reconstruction
Harmonic
Interpolation
Polygon
Discrete inner product

ABSTRACT

A new method for mimetic interpolation on polygonal meshes is described. This new method is based on harmonic function interpolation. Explicit formulas for harmonic functions on general polygons do not exist, so truncated harmonic polynomial expansions are used for computational efficiency. We show that the naive harmonic polynomial expansion, is not stable for arbitrary polygons. However, higher level truncations of harmonic interpolations are stable and accurate. This new method is shown to be a direct extension of the lowest order Raviart-Thomas finite elements to polygons. This method is also a direct extension of the finite volume MAC method to polygons. The accuracy of the interpolation is shown to be first-order irrespective of the polynomial truncation level. However the accuracy of vector Laplace equation solutions using this inner product is shown to be second-order accurate, in keeping with other lowest order Mimetic methods. The versatility of this numerical method is demonstrated on a multiphase incompressible flow problem with a density jump of 1000, on a moving polygonal mesh.

© 2020 Elsevier Inc. All rights reserved.

1. Introduction

1.1. Mimetic reconstruction

Mimetic reconstruction methods are primarily used to develop physics capturing numerical methods for the solution of partial differential equations (PDEs) (for examples see [1–4]). The problem that is addressed within this paper can be framed in a number of different ways, but in this work it is primarily posed as an interpolation problem. Given a set of discrete data items distributed within a spatial domain, determine a reasonable finite dimensional approximation for a function in that same domain. What is interesting in the mimetic case is that the input data is not typically provided at discrete point locations, but rather as a finite number of integrals over various sub-regions. In non-mimetic interpolation, a vector field discretization is based on the scalar discretization of each of the Cartesian components of the field. In this way, each degree of freedom can be traced back to one particular component of the original field. In contrast, mimetic vector interpolations do not rely on the Cartesian decomposition of the vector field. Only certain directional components of the vector integrated over sub-regions are provided as inputs. This removes the possibility of interpolating each Cartesian component of the vector individually using a scalar interpolation methodology. Finally, the polygon mesh interpolation problem is the most interesting of all, because the number of polygon sides is not a fixed value.

* Corresponding authors.
E-mail address: perot@umass.edu (J.B. Perot).

https://doi.org/10.1016/j.jcp.2020.109853
0021-9991/© 2020 Elsevier Inc. All rights reserved.

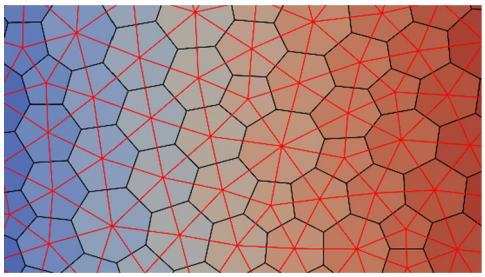

Fig. 1. A computational domain is discretized into polygons which contain information along their faces (black). A vector field is reconstructed at the triangle vertices from face information and used to describe integral quantities along the dual mesh edges (red). (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

A low-order mimetic vector reconstruction problem is illustrated in Fig. 1. This figure shows a set of polygons covering the domain in which the integrals of the normal components of the vector field on each polygon face are known (shown in black), and the goal is to determine a reasonable representation of the vector field within each polygon. Perhaps even more importantly, this work will focus on how to obtain reasonable estimates for other integral quantities of that vector field, for example, integrals of the interpolated vector field along the red (dual mesh) lines shown in Fig. 1.

The example shown in Fig. 1 might initially look odd. But the location and the form of the data (as normal components) is critical to designing mimetic methods that capture physics well. Assuming each polygon represents a potentially different material then certain types of physical vector fields are very well represented by this data format. For example, the normal component of a magnetic flux density vector field ($\boldsymbol{B}$) is always continuous between two different materials even though its tangential component can jump discontinuously across the material interface. Similarly, in a multi-material inviscid fluid flow (say air and water), the normal velocity component is continuous on the interface between two moving materials (or a vacuum/void would be created) even though the tangential velocity component is discontinuous. A final example occurs in heat conduction, where the heat flux normal to (or through) a material interface is the same on each side of the interface but the heat flux tangential to the interface depends on the thermal conductivity on either side of the material interface. For these vector quantities, only the normal vector component is well-defined on the material interface.

Mimetic methods obtain their remarkable physics capturing properties by discretizing both the mathematics and the physics of PDEs exactly. However, no numerical method is totally exact, so mimetic methods therefore invariably force all numerical errors to occur in the material relations of the problem (which are entirely empirical anyway). Enforcing the correct continuity on vector quantities at material interfaces is therefore critical to obtaining a mimetic method. A comprehensive description on the characteristics constituting a mimetic method is provided in Bochev and Hyman [5]. There are mimetic finite difference [6–9], finite volume [10–13], and finite element methods [14–22], but they all share the common feature of the underlying vector data representation that will be investigated in this paper.

To borrow from the notation of differential forms, physical vector fields that should have continuous normal components across material interfaces correspond to, 2-forms or 2-fields. Not all physical vector fields are 2-fields. Electric field strengths ($\boldsymbol{E}$) have continuous tangential components across material interfaces and jumps in the normal component. In fluid flow, the pressure gradient vector field has continuous tangential components and a discontinuous normal component. And in heat transfer, temperature gradient vector fields are continuous in the tangential direction to material interfaces but not in the normal direction. These types of vector fields correspond to 1-forms or 1-fields. The gradient of a scalar quantity is very often a 1-field. The mimetic information layout for 1-forms is typically along (vector components tangential to) polygon edges. The descriptive notation is relative to 3D space even though our pictures and problem are in 2D for this paper. In 3D, the faces of polyhedra are 2D objects and hold 2-field data. The edges of polyhedra (boundaries of the faces) are 1D objects and hold 1-field data. In 2D problems the topological distinction is less obvious, and is primarily if something is normal (2-field) or tangential (1-field) to the edges of the mesh. Far more details on mimetic methods and their relationship to exterior calculus and algebraic topology can be found in reference [23].

This paper focuses on the low-order mimetic reconstruction of 2-fields on polygons in 2D. The equivalent low-order mimetic vector reconstruction on triangles [24] and rectangles [25] is already well understood.

1.2. Related methods

In the case of triangles, the reconstruction of 2-fields is a solved problem. The underlying interpolation is called the Raviart-Thomas element [14]. For the lowest order Raviart-Thomas element the underlying interpolation is a sub-linear polynomial in each triangle of the form. $\mathbf{u} = \mathbf{a} + \frac{d}{m}\mathbf{x}$. Note that m is the number of spatial dimensions (2 or 3), and d is the divergence of the vector field. The divergence is assumed to be a constant in each triangle, and $\mathbf{a}$ is a constant vector. The spatial dimension, m, allows this same formula to apply to tetrahedra and the lowest order Nedelec [15] face element in 3D. This interpolation (which is not a full linear polynomial) produces a normal vector component on each triangle face that is constant everywhere along that face. Even though the position $\mathbf{x}$ varies along the face, the normal component of the

position is always constant on a planar face. The tangential component of the vector interpolation is not constant and varies linearly on each face and discontinuously across the face.

In the case of rectangular meshes, the reconstruction is also called the Raviart-Thomas element. But now it has the form $\mathbf{u} = \mathbf{a} + \mathbf{D}\mathbf{x}$ where the matrix $\mathbf{D}$ is diagonal. This also has a constant divergence ($d =$ trace of $\mathbf{D}$) inside each rectangle and a constant normal component along each face of the rectangle. The 3D versions of the Raviart-Thomas elements on bricks are called Nedelec face-elements (or Nedelec face elements of the first kind). There are also 2D and 3D elements for 1-fields called Nedelec edge elements [15] which are not the focus of this paper. Our polygon face-elements will recover Raviart-Thomas elements when the polygons are triangles or rectangles, but will differ when the polygons are general quadrilaterals.

Polygon interpolation has been extensively used within the context of computer graphics. The interpolation methods for visualization are typically focused on interpolation using scalar data located at the polygon corners (vertices). In that application, generalized barycentric coordinates are often used, including Wachspress [26], mean value [27,28], and discrete harmonic [29], coordinates, among many others [30–32]. These methods use an interpolation that uses only the three closest corner data points. Of the ones listed, only mean-value coordinates produce non-negative interpolations on a concave polygon. However, Wachspress coordinates are interesting because they become classical FE scalar basis functions (linear and bilinear) on triangles and rectangles. Natural neighbor interpolation [33] and non-Sibson methods [34] use all the corner data but are more expensive to calculate because they require constructing a Voronoi diagram for every point of interest in the polygon interior. This makes the evaluation of integral quantities on the interpolant fairly expensive.

Polygonal FE methods [35,36] use scalar barycentric basis functions and specified quadrature rules to define the method. Like classical FE methods the unknowns are defined on the polygon corners and polygonal FE methods are not mimetic or related to Raviart-Thomas (RT) or Nedelec FE methods. The specific quadrature rules for polygonal FE methods are important for these methods, because the scalar barycentric basis functions are rational polynomials (rather than simple polynomials used for most FE methods) and therefore the quadrature rules are no longer exact and influence the final method. Alternatively, taking the gradient of scalar barycentric basis functions and then rotating by 90 degrees produces a possible basis function for mimetic 2-field vector interpolation. These basis function have a constant normal on each polygon face (like RT elements), but they do not have a constant divergence (like RT elements) on rectangular meshes or on polygons. These basis functions are the same as RT elements on triangles but are not the same as RT elements on rectangles.

This work is perhaps more closely related to Virtual element methods (VEM) [37–41] and mimetic finite difference (MFD) methods [42]. These methods are an extension of mimetic SOM (Support Operator Method) finite difference ideas [42–45] to polygons. The elements are “virtual” in these methods because the basis functions are not explicitly known or formed. The fact that basis functions never need to be formally constructed or explicitly known is a useful insight that is also used in finite volume methods. In virtual element methods, the focus is switched (as it is in all SOM) to be on the definition of the discrete inner product. The proposed reconstruction in this paper could be interpreted as a very specific type of VEM.

It some ways this work is also closely related to harmonic interpolation. Harmonic functions have excellent mathematical interpolation properties. But typically harmonic functions are too expensive to compute in general polygons to be a practical solution. First our work will use the gradient of a harmonic function to obtain the correct 2-field continuity properties to produce a mimetic method. Second we will use a truncated expansion of the harmonic function into harmonic polynomial series to make the method computationally efficient. Our method differs from the RT element on a quadrilateral because it solves for the harmonic function on the original quadrilateral mesh, whereas FE methods use a mapping to a reference square. RT elements therefore struggle on concave quadrilaterals where the mapping is not unique, whereas our functions are well defined for any (untwisted) polygon.

1.3. Summary

Section 2 of this paper describes a mimetic vector reconstruction on polygons using harmonic polynomials. In section 3 mimetic vector reduction (integration of the interpolant) is discussed. These two processes then allow us to relate this method to other polygon discretization approaches. Section 4 discusses the accuracy of the interpolation and shows its use for PDE solutions. A brief discussion of the results is found in Section 5.

2. Vector interpolation

2.1. Interpolant

The proposed method is based on the assumption that the vector interpolant in each polygon has the functional form,

$$\mathbf{u} = \nabla\phi + \frac{d}{m}\mathbf{x} \tag{1a}$$

where ϕ is a harmonic function so its Laplacian is zero,

$$\nabla^2\phi = 0 \tag{1b}$$

and which obeys the Neumann boundary condition on each polygon face of,

$$\frac{\partial \phi}{\partial n} = (\mathbf{v}\cdot\hat{\mathbf{n}})_f - \frac{d}{m}\mathbf{x}\cdot\hat{\mathbf{n}}_f \tag{1c}$$

where $(\mathbf{v}\cdot\hat{\mathbf{n}})_f$ is the known vector data (face normal components) that will be interpolated by the continuous function, $\mathbf{u}$, in each polygon. There is also a solvability condition, because of the Neumann boundary conditions, that the average is zero

$$\int_{\Omega_p} \phi dV = 0 \tag{1d}$$

where Ω_p is the polygon volume. This last condition makes the harmonic function definition unique, but it is not critical, because the interpolation (1a) really only needs the gradient, so the constant can be set in any reasonable way.

In practice, the origin of the position vector is best chosen to be inside the polygon to keep the divergence terms (with d) relatively small. Using the centroid defined as the average position (and equivalent to the center of gravity, given constant density) of the polygon can simplify later mathematical results (because the cell average of the divergence term is then zero).

The vector interpolant in (1a) has a number of very attractive properties:

P1. Harmonic functions are in some sense the smoothest interpolant (least variation) possible and they never create a new maximum or minimum of the function inside the polygon. The gradient will therefore also be smooth (low variation) as well. And each Cartesian component of the gradient is also a harmonic function itself.
P2. Harmonic functions are well defined even for polygons with concave corners. Many (but not all) polygon interpolants struggle with this situation.
P3. The gradient of a harmonic function never goes to zero inside the polygon unless that zero gradient is at a saddle point (stagnation point for a fluid flow velocity field).
P4. This functional form is a direct generalization of the Raviart-Thomas triangle and Nedelec tetrahedral face elements, with the gradient of a harmonic function replacing the constant vector, **a**.

This functional form for the interpolant also obeys the three fundamental criteria of the lowest-order Raviart-Thomas and Nedelec face elements:

C1. The divergence of the interpolant velocity field is constant inside each polygon and is entirely due to the final term in equation (1a).
C2. The curl of the interpolant velocity field is zero inside each polygon. Face normal vector data (our input) has zero information about the curl contained in it.
C3. The value of the interpolant that is normal to each face is constant on each face.

The inability of the interpolant to represent vector fields with a finite curl inside each polygon should not be interpreted as a weakness of the method. The interpolation given by (1a) will place all the rotation, or curl, in thin layers that reside on the faces between the polygons. For a flow field with rotation, the vorticity therefore lies in thin sheets on the faces of the polygons. Rotation of the vector field is therefore captured by the interpolant, but only at the mesh scale level. Rotation at the subgrid level, inside each polygon, is assumed to be zero. This is the correct assumption because the integral average subgrid rotation within the polygon is entirely dictated by the tangential velocity on the polygon faces, ($\int_{\Omega_p}(\nabla\times\mathbf{u})\cdot\hat{\mathbf{n}}dA = \oint_{\partial\Omega_p}\mathbf{u}\cdot d\mathbf{l}$ where $\partial\Omega_p$ is the polygon boundary and $\hat{\mathbf{n}}$ points out of the 2D plane), of which we have absolutely no information in a mimetic 2-field reconstruction. The identical situation also exists for Raviart-Thomas and Nedelec face elements which are widely and very successfully used in electromagnetics where rotation is very common in the magnetic field.

The primary difficulty of the proposed interpolant (equation (1a)) is that the harmonic problem (eqn (1b)) does not have a closed form solution (except for triangles/tetrahedra and Cartesian meshes of rectangles/bricks). Methods have been proposed [46,47] that precompute approximations to the harmonic basis functions in each polygon, but the computational effort does not warrant the numerical return for PDE problems. The proposed method, described in the next section, is not computationally expensive, nor does it require a numerical solution for the harmonic function ϕ. This approach has some similarities with virtual element methods [37,38] and to mimetic finite difference methods [42].

2.2. Moments

The method uses the polygon face normal component data to build up information about the moments of the function using various manifestations of the Gauss divergence theorem. Given a vector 2-field, **v**, then the simplest property of the face data is that it directly specifies the average divergence,

$$\int_{\Omega_p} \nabla\cdot\mathbf{v}dV = \sum_{f\subset\partial\Omega_p}\int_f \mathbf{v}\cdot\hat{\mathbf{n}}dA \tag{2a}$$

The fluxes on the right hand side are exactly the method's assumed input data. Also for any function, H

$$\int_{\Omega_p} \nabla H \cdot \mathbf{v} dV = \sum_{f \subset \partial\Omega_p} \int_f H \mathbf{v} \cdot \hat{\mathbf{n}} dA - \int_{\Omega_p} H \nabla \cdot \mathbf{v} dV \tag{2b}$$

So moments of the vector field with respect to ∇H can also be directly determined solely by the face normal component data and the divergence (found from (2a)).

More specifically, we now make the lowest-order Raviart-Thomas (and Nedelec face element) fundamental assumptions C1-C3. This gives the following algebraic expressions where $U_f = \int_f \mathbf{v} \cdot \hat{\mathbf{n}} dA$ is the input normal data, and V_p represents the polygon volume.

$$V_p d = \sum_{f \subset \partial\Omega_p} U_f \tag{3a}$$

and the moment expression

$$\int_{\Omega_p} \nabla H \cdot \mathbf{u} dV = \sum_{f \subset \partial\Omega_p} U_f \frac{1}{A_f} \int_f H dA - d \int_{\Omega_p} H dV \tag{3b}$$

which simplifies to

$$\int_{\Omega_p} \nabla H \cdot \mathbf{u} dV = \sum_{f \subset \partial\Omega_p} U_f \left(\frac{1}{A_f} \int_f H dA - \frac{1}{V_p} \int_{\Omega_p} H dV \right) = \sum_{f \subset \partial\Omega_p} \overline{H}^{f-p} U_f \tag{3c}$$

where, A_f is the face area and the notation for the face average minus the polygon average, $\overline{()}^{f-p}$, is now introduced. Note that equation (3c) applies in both 2 or 3 dimensions. But it is restricted to the lowest-order reconstruction assumption. At lowest order, only the face normal data is provided as an input to the reconstruction. The next higher order RT reconstruction assumes the normal vector component varies linearly on the faces, and the divergence varies linearly in the polygon, which would give a more complex algebraic expression than (3c).

The method will use the moment expression (3c) extensively. And most of the time it will use expression (3c) when H is a harmonic polynomial.

2.3. Harmonic polynomials

In 2D harmonic polynomials are typically represented using a single complex number. But that representation makes the extension of the method to 3D difficult, so this paper will not use complex numbers.

The interpolation expands the harmonic function in equation (1a) in terms of harmonic polynomials P_i and Q_i, so the interpolant can be written,

$$\mathbf{u} = \frac{d}{m}\mathbf{x} + \sum_i (a_i \nabla P_i + b_i \nabla Q_i) \tag{4}$$

where a_i and b_i are constants and where formally the summation should be to infinity for the perfect harmonic interpolant, but in practice a very short truncation of the series (at 2 or 3 terms) is practically sufficient which is shown in the results section. The truncated interpolant still has a constant divergence (C1), and zero curl (C2), but the truncated interpolant does not have a perfectly constant normal component on each face (condition C3 is relaxed). On triangles, one term of the series is sufficient to exactly satisfy C3. On rectangles (of any orientation) two terms of the series is sufficient to exactly satisfy C3. However, on general quadrilaterals (and general polygons with more than four sides) the infinite series is formally required to exactly satisfy C3.

The *standard* harmonic polynomials (which come from Pascal's triangle and the binomial theorem) are:

$$\begin{aligned}
&\tilde{P}_1 = x \quad \tilde{Q}_1 = y \\
&\tilde{P}_2 = x^2 - y^2 \quad \tilde{Q}_2 = 2xy \\
&\tilde{P}_3 = x^3 - 3xy^2 \quad \tilde{Q}_3 = 3x^2 y - y^3 \\
&\tilde{P}_4 = x^4 - 6x^2 y^2 + y^4 \quad \tilde{Q}_4 = 4x^3 y - 4xy^3
\end{aligned} \tag{5}$$

Note that a generating sequence for these is

$$\tilde{P}_{i+1} = x\tilde{P}_i - y\tilde{Q}_i \quad \text{and} \quad \tilde{Q}_{i+1} = y\tilde{P}_i + x\tilde{Q}_i \tag{6a}$$

This work is concerned with the gradients of harmonic polynomials and these are also related,

$$\nabla \tilde{P}_{i+1} = (i+1)\begin{pmatrix} \tilde{P}_i \\ -\tilde{Q}_i \end{pmatrix} \quad \text{and} \quad \nabla \tilde{Q}_{i+1} = (i+1)\begin{pmatrix} \tilde{Q}_i \\ \tilde{P}_i \end{pmatrix} \tag{6b}$$

In practice it will be easier for the method to use the *orthogonal* harmonic polynomials. If the position is specified to have the origin at the polygon centroid then the orthogonal polynomials are identical to equation (5) for the first two levels. However, the orthogonal third level polynomials generalize equation (6a) and become,

$$\begin{aligned} P_3 &= (x-c_x)(P_2 - 3\overline{P_2}^p) - (y-c_y)(Q_2 - 3\overline{Q_2}^p) \\ Q_3 &= (y-c_y)(P_2 - 3\overline{P_2}^p) + (x-c_x)(Q_2 - 3\overline{Q_2}^p) \end{aligned} \tag{7a}$$

where a bar represents a polygon cell average, and the constant vector $\mathbf{c}$ is given by

$$\mathbf{c} = \frac{3\overline{\mathbf{x}(\mathbf{x}^2)}^p}{2\overline{\mathbf{x}^2}^p} \tag{7b}$$

which captures the geometric skew of the polygon. This vector is zero for any symmetric polygon.

The derivation of equations (7a) and (7b) is in Appendix A, and we note that it is the gradients of these polynomials that we require to be orthogonal, so $\int_{\Omega_p} \nabla P_i \cdot \nabla P_j dV = 0$ and $\int_{\Omega_p} \nabla Q_i \cdot \nabla Q_j dV = 0$ for all $i \neq j$, and $\int_{\Omega_p} \nabla P_i \cdot \nabla Q_j dV = 0$ always. The orthogonal polynomials have the same highest order polynomial terms as the standard polynomials, but add lower order terms in the correct amounts to produce orthogonality.

2.4. Reconstruction

The constants a_i and b_i in the interpolation expansion (equation (4)) are determined by inserting this into the moment equation (3c).

$$\int_{\Omega_p} \nabla H \cdot \{\frac{d}{m}\mathbf{x} + \sum_i (a_i \nabla P_i + b_i \nabla Q_i)\} dV = \sum_{f \subset b(\Omega_p)} \overline{H}^{f-p} U_f \tag{8}$$

When H is each of the orthogonal harmonic polynomials we obtain

$$V_p \begin{pmatrix} \overline{\nabla P_1 \cdot \nabla P_1}^p a_1 \\ \overline{\nabla Q_1 \cdot \nabla Q_1}^p b_1 \\ \overline{\nabla P_2 \cdot \nabla P_2}^p a_2 \\ \vdots \end{pmatrix} = \begin{bmatrix} \overline{P_1}^{f1-p} & \overline{P_1}^{f2-p} & \dots & \overline{P_1}^{fN-p} \\ \overline{Q_1}^{f1-p} & \overline{Q_1}^{f2-p} & & \overline{Q_1}^{fN-p} \\ \overline{P_2}^{f1-p} & & & \\ \vdots & & & \end{bmatrix} \begin{pmatrix} U_{f1} \\ U_{f2} \\ \vdots \\ U_{fN} \end{pmatrix} - V_p \frac{d}{m} \begin{pmatrix} \overline{\nabla P_1 \cdot \mathbf{x}}^p \\ \overline{\nabla Q_1 \cdot \mathbf{x}}^p \\ \overline{\nabla P_2 \cdot \mathbf{x}}^p \\ \vdots \end{pmatrix} \tag{9a}$$

And when H is $\frac{1}{2}\mathbf{x}^2$ then

$$\frac{d}{m}\int_{\Omega_p} \mathbf{x}^2 dV + \int_{\Omega_p} \sum_i (a_i \mathbf{x} \cdot \nabla P_i + b_i \mathbf{x} \cdot \nabla Q_i) dV = \sum_{f \subset b(\Omega_p)} \overline{\frac{1}{2}\mathbf{x}^2}^{f-p} U_f \tag{9b}$$

Both expressions can be simplified due to the fact that the polynomials are harmonic,

$$V_p \overline{\nabla P_1 \cdot \mathbf{x}}^p = \int_{\Omega_p} \nabla P_i \cdot \mathbf{x} dV = \int_{\Omega_p} \{\nabla \cdot (P_i \mathbf{x}) - P_i (\nabla \cdot \mathbf{x})\} dV = \sum_{f \subset b(\Omega_p)} \int_f P_i \mathbf{x} \cdot \hat{\mathbf{n}} dA - m \int_{\Omega_p} P_i dV$$

which simplifies (using the centroid coordinate system) to

$$V_p \overline{\nabla P_1 \cdot \mathbf{x}}^p = \sum_{f \subset b(\Omega_p)} m V_f \frac{1}{A_f} \int_f P_i dA - m \int_{\Omega_p} P_i dV = m \sum_{f \subset b(\Omega_p)} V_f \overline{P_i}^{f-p} = m V_p \widehat{\overline{P_i}} \tag{10a}$$

where V_f is the volume formed by the triangle made by the face and the polygon centroid. And

$$\widehat{\overline{P_i}} = \frac{1}{V_p} \sum_{f \subset b(\Omega_p)} V_f \overline{P_i}^{f-p} \tag{10b}$$

is a volume weighted average of the face polynomial values for each polygon.

This means that (9b) can be rewritten as,

$$V_p \frac{d}{m}\overline{\mathbf{x}^2}^p + mV_p \sum_i (a_i \widehat{\overline{P_i}} + b_i \widehat{\overline{Q_i}}) = \sum_{f \subset b(\Omega_p)} \overline{\frac{1}{2}\mathbf{x}^2}^{f-p} U_f$$

And we define a new unknown for each polygon

$$c = \sum_i (a_i \widehat{\overline{P_i}} + b_i \widehat{\overline{Q_i}}) \tag{10c}$$

that is useful like the divergence, $d = \frac{1}{V_p}\sum_{f \subset b(\Omega_p)} U_f$ (equation (3a)).

Equations (9a) and (9b) then become,

$$V_p \begin{bmatrix} 0 & 1 & & & & & \\ 1 & \frac{1}{m^2}\overline{\mathbf{x}^2}^p & & & & & \\ & & \overline{(\nabla P_1)^2}^p & & & & \\ & & & \overline{(\nabla Q_1)^2}^p & & & \\ & & & & \overline{(\nabla P_2)^2}^p & & \\ & & & & & \ddots & \end{bmatrix} \begin{pmatrix} c \\ d \\ a_1 \\ b_1 \\ a_2 \\ \vdots \end{pmatrix}$$

$$= \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \frac{1}{m}\frac{1}{2}\overline{\mathbf{x}^2}^{f1-p} & \frac{1}{m}\frac{1}{2}\overline{\mathbf{x}^2}^{f2-p} & & \frac{1}{m}\frac{1}{2}\overline{\mathbf{x}^2}^{fN-p} \\ \overline{P_1}^{f1-p} - \widehat{\overline{P_1}} & \overline{P_1}^{f2-p} - \widehat{\overline{P_1}} & & \overline{P_1}^{fN-p} - \widehat{\overline{P_1}} \\ \overline{Q_1}^{f1-p} - \widehat{\overline{Q_1}} & & & \\ \overline{P_2}^{f1-p} - \widehat{\overline{P_2}} & & & \\ \vdots & & & \end{bmatrix} \begin{pmatrix} U_{f1} \\ U_{f2} \\ \vdots \\ U_{fN} \end{pmatrix} \tag{11a}$$

This is the primary algebraic equation for the reconstruction. Once the divergence and the a_i and b_i constants are determined from equation (11a) the interpolant (given by equation (4)) is known. In this paper, the "level" of the reconstruction refers to the number of harmonic polynomials used (or the order of the polynomial truncation). Level-1 reconstruction uses just the top 4 rows of the matrix in eqn (11a). Level-1 reconstruction uses only the harmonic functions with a subscript of 1. These are linear functions, and this reconstruction only finds the mean velocity in each polygon. Level-2 reconstruction uses 6 rows (and 4 harmonic polynomials). Level-3 reconstruction uses 8 rows in the matrix and produces 8 results per polygon.

The use of the orthogonal harmonic polynomials and the added new variable, c, means that the left-side matrix is nearly diagonal and very easy to invert. Only the divergence term in the interpolant causes some slight coupling with the new variable, c. The polygon value of c is useful to keep the computational costs low and to see some of the method's inherent symmetries. We will sometimes write equation (11a) in a matrix notation. This same matrix notation can apply to a single polygon or to a large set of them.

$$\mathbf{V}a = \mathbf{R}U_f \tag{11b}$$

The two matrices depend entirely on the polygon geometry. The use of $\widehat{\overline{P_i}}$ in the right hand side matrix **R** is a mathematical manipulation that keeps the divergence terms and the harmonic polynomial contributions fully uncoupled from each other in **V**. The matrix **R** is typically tall. The number of possible rows is set by the level of the desired series truncation and the number of columns is set by the number of polygon faces.

The right-hand side of equation (11) produces various integral moments of the velocity field on the polygon. The third and fourth rows correspond to the mean velocity in the polygon. The fifth and sixth rows are related to linear velocity moments. Similarly, the a_i and b_i constants also have a physical interpretation. The a_1 and b_1 values correspond to the velocity at the centroid. The level-2 constants are related to some of the velocity gradients at the centroid.

Note that the use of orthogonal polynomials is what makes **V** a nearly diagonal matrix. The top left corner is not orthogonal because the first term in the expansion given by equation (4), the divergence producing term, is not a harmonic polynomial. It is possible to use the standard harmonic polynomials (in equation (5)) for the method, but then **V** is not diagonal (though it still has many zeros in the lower level entries if the centroid is used for the position origin). The assumption of this work is that using slightly more complex polynomials at level-3 and higher is worth the cost saving of not inverting the matrix **V** in each polygon. The inverse of the nearly diagonal **V** is used frequently and is,

$$\mathbf{V}^{-1} = \frac{1}{V_p}\begin{bmatrix} -\frac{1}{m^2}\overline{\mathbf{x}^2}^p & 1 & & & & \\ 1 & 0 & & & & \\ & & \frac{1}{\overline{(\nabla P_1)^2}^p} & & & \\ & & & \frac{1}{\overline{(\nabla Q_1)^2}^p} & & \\ & & & & \frac{1}{\overline{(\nabla P_2)^2}^p} & \\ & & & & & \ddots \end{bmatrix} \tag{11c}$$

3. Discrete vector inner product

3.1. Dual

The transpose operation to interpolation, is a reduction operation. Interpolation takes the normal component data at the polygon faces and produces a large set of moments (or centroid values and gradients) for each polygon. In contrast, reduction takes the polygon centroid values and gradients and produces a scalar 1-form data item along the dual mesh edges (the red lines in Fig. 1). The result is equivalent in number, but not in information content, to the original data set used for interpolation.

The combination of an interpolation and a reduction is a transformation from 2-field face normal data on the polygonal mesh to 1-field edge tangential data on a dual mesh. This entire transformation can be viewed as a discrete Hodge star operator in the language of differential forms. It can alternatively be viewed as a Finite Element (FE) mass matrix, or a discrete vector inner product for a Support Operator Method (SOM). It can also be viewed as the inner product matrix for a virtual element method (VEM).

In this work, the dual mesh that we will be using is not actually the red lines of figure 1. We will be using the simple average of all possible dual meshes. That is the average of all possible curve segments (not necessarily straight lines) with one endpoint somewhere inside the polygon and the other endpoint somewhere on the face. This average of all possible dual meshes is a common, though largely unrecognized, result for FE methods as shown in Matiussi [48]. The awkwardness of an average over many dual meshes is one reason the FE method typically avoids the explicit specification of any dual mesh in favor of describing the method in terms of the weak form of the partial differential equation (PDE). However, the two viewpoints are entirely equivalent.

3.2. Average line integral

The vector interpolant in the polygon can also be written as a gradient

$$\mathbf{u} = \nabla\{\frac{d}{m}\frac{1}{2}\mathbf{x}^2 + \sum_i (a_i P_i + b_i Q_i)\}$$

This means that the integral along any curve only depends on the endpoint values,

$$\int_{\tilde{e}} \mathbf{u}\cdot d\mathbf{l} = \{\frac{d}{m}\frac{1}{2}\mathbf{x}^2 + \sum_i (a_i P_i + b_i Q_i)\}|_f - \{\frac{d}{m}\frac{1}{2}\mathbf{x}^2 + \sum_i (a_i P_i + b_i Q_i)\}|_p$$

where one endpoint is on the face and one endpoint is inside the polygon. And the tilde on the integral subscript indicates a dual edge (red) (and not a polygon edge, black). Averaging over all possible curves of this type gives,

$$g_{\tilde{e}} = \overline{\int_{\tilde{e}} \mathbf{u}\cdot d\mathbf{l}} = \frac{d}{m}\overline{\frac{1}{2}\mathbf{x}^2}^{f-p} + \sum_i (a_i \overline{P_i}^{f-p} + b_i \overline{Q_i}^{f-p})$$

rewriting this in term of existing quantities, and the variable c, this becomes,

$$g_{\tilde{e}} = \overline{\int_{\tilde{e}} \mathbf{u}\cdot d\mathbf{l}} = \frac{d}{m}\overline{\frac{1}{2}\mathbf{x}^2}^{f-p} + \sum_i (a_i\{\overline{P_i}^{f-p} - \widehat{\overline{P_i}}\} + b_i\{\overline{Q_i}^{f-p} - \widehat{\overline{Q_i}}\}) + c$$

This can be written for all the dual edges emanating from the polygon in matrix (transpose) form as,

$$\begin{pmatrix} g_{\tilde{e}1} \\ g_{\tilde{e}2} \\ \vdots \\ g_{\tilde{e}N} \end{pmatrix} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ \frac{1}{m}\overline{\frac{1}{2}\mathbf{x}^2}^{f1-p} & \frac{1}{m}\overline{\frac{1}{2}\mathbf{x}^2}^{f2-p} & & \frac{1}{m}\overline{\frac{1}{2}\mathbf{x}^2}^{fN-p} \\ \overline{P_1}^{f1-p} - \widehat{\overline{P_1}} & \overline{P_1}^{f2-p} - \widehat{\overline{P_1}} & & \overline{P_1}^{fN-p} - \widehat{\overline{P_1}} \\ \overline{Q_1}^{f1-p} - \widehat{\overline{Q_1}} & & & \\ \overline{P_2}^{f1-p} - \widehat{\overline{P_2}} & & & \\ \vdots & & & \end{bmatrix}^T \begin{pmatrix} c \\ d \\ a_1 \\ b_1 \\ a_2 \\ \vdots \end{pmatrix} \tag{12a}$$

Or in shorthand notation,

$$g_{\tilde{e}} = \mathbf{R}^T a \tag{12b}$$

where this operation uses the same $\mathbf{R}$ matrix as in equation (11b) but with the transpose.

The average line integral operation (to construct a discrete dual 1-field) is nearly the transpose of the interpolation operation (that uses a discrete 2-field), but it does not use the nearly diagonal matrix $\mathbf{V}$.

3.3. Discrete Hodge star

The FE mass matrix, or SOM inner product, or the discrete Hodge star from 2 forms to dual 1-forms, is

$$g_{\tilde{e}} = \mathbf{R}^T \mathbf{V}^{-1} \mathbf{R} U_f. \tag{13}$$

This matrix is square and symmetric and has a size equal to the number of polygon faces (or dual edges, which are 1-to-1). This matrix is non-singular if enough levels of orthogonal polynomials are used. For triangles, only an expansion including level 1 is required (4 rows per polygon for $\mathbf{R}$). Quadrilaterals and pentagons require expansions with level 2 (polynomials with subscripts of 2, 6 rows per polygon for $\mathbf{R}$). Hexagons and septagons appear to require level 3 (up to and including the harmonic cubics) polynomials in theory, but in practice level 2 is actually sufficient in all our test problems, because of the coupling between the many polygons. Singularity comes from having an eigenvalue equal to zero. Zero eigenmodes which are possible for a simple mesh (say a single many sided polygon) are not possible when coupled with other polygons. An eigenmode is a fully coupled feature of the entire mesh.

It is useful to consider what this matrix looks like in the limit of the infinite level of harmonic polynomial expansion. To do this, we restructure the interpolation potential into a basis function for each face, so

$$\mathbf{u} = \sum_f \nabla \phi^f U_f \tag{14}$$

where each of these face's basis functions obeys the following properties in the polygon.

$$\begin{aligned} &\nabla^2 \phi^f = \frac{1}{V_p} && \text{Solution of the Poisson Equation with constant source.} && \text{(a)} \\ &\frac{\partial \phi^f}{\partial n}|_g = \frac{1}{A_g}\delta_{fg} && \text{The normal derivative is 1 on its own face and 0 on all others.} && \text{(b)} \\ &\int_{\Omega_p} \phi^f dV = 0 && \text{Extra solvability condition. Zero average value.} && \text{(c)} \end{aligned} \tag{15}$$

Equation (14) is an equally valid expansion that satisfies the three RT constraints C1-C3. Constant divergence, zero curl, and constant normal component on each face. Note that,

$$\nabla \cdot \mathbf{u} = \sum_f \nabla^2 \phi^f U_f = \frac{1}{V_p} \sum_f U_f = d$$

$$\mathbf{u} \cdot \hat{\mathbf{n}}|_g A^g = \sum_f U_f \frac{\partial \phi^f}{\partial n}|_g A^g = U_g$$

The line integral over a curve is

$$\int_p^f \mathbf{u} \cdot d\mathbf{l} = \sum_g (\phi^g|_f - \phi^g|_p) U_g$$

So the average integral over all possible curves is,

$$g_{\tilde{e}} = \overline{\int_p^f \mathbf{u} \cdot d\mathbf{l}} = \sum_g \overline{\phi^g}^f U_g \tag{16a}$$

where the cell average is zero because of (15c).

In matrix terms this is

$$\begin{pmatrix} g_1 \\ g_2 \\ \cdots \\ g_N \end{pmatrix} = \begin{bmatrix} \overline{\phi^1}^1 & \overline{\phi^2}^1 & & \overline{\phi^N}^1 \\ \overline{\phi^1}^2 & \overline{\phi^2}^2 & & \\ & & \cdots & \\ \overline{\phi^1}^N & & & \overline{\phi^N}^N \end{bmatrix} \begin{pmatrix} U_1 \\ U_2 \\ \cdots \\ U_N \end{pmatrix} \tag{16b}$$

on each polygon. It is possible to show that this matrix is symmetric. Because,

$$\int_{\Omega_p} \nabla\phi^g \cdot \nabla\phi^f dV = \int_{\Omega_p} \nabla \cdot (\phi^g \nabla\phi^f) dV = \oint_{\partial\Omega_p} \phi^g (\nabla\phi^f) \cdot \hat{\mathbf{n}} dA = \overline{\phi^g}^f$$

$$\int_{\Omega_p} \nabla\phi^f \cdot \nabla\phi^g dV = \int_{\Omega_p} \nabla \cdot (\phi^f \nabla\phi^g) dV = \oint_{\partial\Omega_p} \phi^f (\nabla\phi^g) \cdot \hat{\mathbf{n}} dA = \overline{\phi^f}^g$$

And using the same logic each diagonal entry is positive.

This shows that in the limit, the Hodge star or mass matrix only depends on average face values of the N different Poisson solutions. The entire harmonic solution in the polygon interior is not required. Equation (16b) is the limit of equation (13). Equation (13) is easier to compute, and it gives us detailed insight into the vector field itself because it can be broken into three parts. Equation (16) only reveals the entire flux to line integral transformation, but not its intermediate parts that tell us vector field information.

3.4. Discrete sharp and flat

It is hypothesized here that every mimetic method for solving PDEs uses Hodge star operators (or FE mass matrices, or discrete inner products) either explicitly or implicitly. We assume that every Hodge star can be split into two submatrices, such as the **V** and **R** matrices in equation (13), where those matrices provide addition method insight. For example, in references [49,50] **R** and its pseudo-inverse (**R** is not square) are used to show conservation of momentum. Conservation of a vector quantity like momentum is a non-trivial proof for most mimetic methods because vectors do not exist within the discretization and only normal components exist. A matrix like **R** allows a vector representation (the values of the constants a_1 and b_1 are the mean Cartesian vector values) to exist in a well-defined sense in every polygon. This allows one to then prove properties about these quantities.

In exterior calculus (using differential forms) the sharp operator converts 1-forms to traditional vectors, and the flat operator converts from vectors to 1-forms. Hodge star operators convert from 2-forms to 1-forms and vice versa. The matrix $\mathbf{R}^T$ is acting very much like a discrete flat operator for the dual mesh taking polygon vector information to a discrete 1-form. The matrix **R** is a discrete analog of a sharp operator for 2-forms on the primary (polygon) mesh. Or more formally, it is a discrete Hodge star, to take the 2-form to a 1-form, and then a discrete sharp operator, to obtain the vector in the polygon. We note that at the discrete level the Hodge star operators can be fundamental operations (as in equation (16b)) or they can be derived from discrete sharp and flat operators as in equation (13). And more insight and utility is derived at the discrete level by making the discrete sharp operator fundamental. And then deriving the flat (transpose), and the Hodge star (product) from that initial well defined discrete sharp operator.

3.5. Virtual elements

Virtual element methods (VEM) [37–41] are an extension of SOM, which was original framed as a finite difference method. The idea is to focus on defining a discrete inner product. This discrete inner product is often called a mass matrix in FE, the particular mass matrix in this work (and VEM) is a mass matrix for vector quantities. The VEM (and often the SOM) vector mass matrices use the mimetic face normal vector components as their input. In SOM this inner product is often built using intuition and many different versions have been proposed. In the case of the VEM which is specifically designed for polygons, the inner product is defined in two parts. The first part enforces linear accuracy but is singular, and the second part is called "stabilization" and adds an additional term which makes the inner product matrix positive definite, while not changing the accuracy. These expressions for the inner product are written in matrix form similar to our equation (13) with *wide* matrices $\mathbf{R}_1$ and $\mathbf{R}_2$.

$$\mathbf{M} = \mathbf{R}_1^T \mathbf{V}_1^{-1} \mathbf{R}_1 + \mathbf{R}_2^T \mathbf{V}_2^{-1} \mathbf{R}_2 \tag{17}$$

where the first term is for accuracy, and the second term for positive definiteness of $\mathbf{M}$. For the low-order case $\mathbf{R}_1$ is only two rows per polygon. It reconstructs an estimate for the average vector in the polygon. The second "stability" term is often formed using one of the many different polygonal barycentric coordinates [26–30].

Because the approach proposed in this paper produces a nearly diagonal matrix $\mathbf{V}$ (due to the orthogonal harmonic polynomials), it can also be pulled apart into two terms like equation (17). $\mathbf{R}_1$ then contains the third and fourth rows of equation (11a), which are the level 1 polynomials, and $\mathbf{R}_2$ contains the divergence terms (first two rows) and the level-2 polynomials and higher. The $\mathbf{V}$ matrix is split similarly. This shows that the proposed method could be considered as a type of VEM. But the "stability" term is very specifically constructed for additional properties than stability, and barycentric coordinates are not used. The attractive interpolative properties of harmonic functions could give our particular "stability" matrix some additional benefits.

3.6. Computational implementation

It is common to build the sparse matrix $\mathbf{R}^T\mathbf{V}^{-1}\mathbf{R}$ and store it. This matrix has a nominal size $NF \times NF$ where NF is the number of polygon faces. The common polygon mesh is on average hexagonal, so on average $NF = 3NP$ where NP is the number of polygons. On average each polygon has 6 faces, and so on average each row (or column) of the matrix has 1 diagonal and 10 off-diagonal entries.

A naïve implementation of the matrix, storing each non-zero value, will require 10 gather operations per face and 30 gather operations per polygon on average in order to do the off-diagonal components of a matrix-vector multiply. Each gather operation will also require an aligned (not random) memory retrieval for the matrix data and a multiply and add operation. This will result in 32 aligned memory accesses per polygon on average. It will also require the construction of the matrix sparsity structure which we will call the F2F (face to face) list. The gather and scatter operations require random memory access and they dominate the computational time on any modern computer. The matrix data is memory aligned (and not random) and requires about 1/5 the cost of the gather (depending on the hardware). The math operations are about 1/40 the cost of the gather. This implementation is identical to the CRS (compressed row storage) format frequently used in matrix packages.

A better approach is to use the structure inherent in the matrix itself. A matrix-vector multiply operation can be performed by looping over the polygons, rather than the faces. Each polygon gathers an average of 6 face data values. It then computes the interpolation and reduction using an average of 36 aligned data retrievals (of matrix data) and 36 multiplies and adds. An average of 6 results per polygon are then scattered back to the faces. This is a total of 12 gather/scatter and 36 aligned memory reads per polygon. The number of expensive random memory operations (gather/scatter) have been reduced by 60%. The number of aligned memory accesses and math operation have increased by 12.5%. The overall performance of this matrix-vector multiply is roughly twice as fast as CRS. This method requires a C2F (cell to face) connectivity structure.

The F2C (face to cell) data structure is far simpler to use and store. It is always 2 integers for each face, which are the cell numbers bordering that face. This data structure is most closely aligned with the representation $\mathbf{R}^T\mathbf{V}^{-1}\mathbf{R}$. One loop is performed for each of the 3 matrices. The matrix $\mathbf{R}$ has its values stored on every face. It requires an average of 12 scatters per face or 36 per polygon. The matrix $\mathbf{V}$ is just math and aligned data access and is very fast. The transpose operation requires 12 gather operations per face and 36 per polygon. This approach has a cleaner connectivity structure (F2C) but is 2x slower than the naïve CRS approach and 4x slower than the polygon based reconstruction loop.

These precise trade-offs will change if the polygons that form the mesh are predominantly triangles or quadrilaterals and not hexagons as assumed above. On some hardware (such as GPUs) the scatter operation (which involves data coalescing) is more difficult to implement than the gather (which is data splitting), and the naïve CRS approach might be more competitive.

4. Results

4.1. Basic test cases

In this section, the proposed interpolation is tested on a series of nearly uniform sized polygonal meshes of different resolution on a unit square. Table 1 shows the four test cases chosen along with their velocity profiles and corresponding stream functions.

The four cases chosen are a sinusoidal rotation case, a stagnation point flow, solid body rotation and a flow with a discontinuous jump in velocity and density at an interface modeling air/water flow. Fig. 2 shows the exact velocity field for each of these cases.

The inputs to these test cases are the exact values of the normal flux of the vector field on the polygon faces, and the output uses equation (11) to determine the moments and the average values of the velocity vector in each polygon. Each of the four cases was tested on four different mesh resolutions with approximately uniform discretization. The meshes were chosen to approximately halve the grid spacing (Δs) with each refinement. The two coarsest meshes used are shown below in Fig. 3.

Table 1
List of cases studied.

Case	Velocity	Stream function
A. Sinusoidal Rotation (constant density)	$\mathbf{v} = \begin{pmatrix} \pi \sin(\pi x)\cos(\pi y) \\ -\pi \cos(\pi x)\sin(\pi y) \end{pmatrix}$	$\psi = \sin(\pi x)\sin(\pi y)$
B. Stagnation Point (constant density)	$\mathbf{v} = \begin{pmatrix} x \\ -y \end{pmatrix}$	$\psi = xy$
C. Solid Body Rotation (constant density)	$\mathbf{v} = \begin{pmatrix} -y \\ x \end{pmatrix}$	$\psi = -\frac{1}{2}\left(x^2 + y^2\right)$
D. Discontinuous Flow: $\rho = \rho_b = 1000$ for $y < 0$, and $\rho = \rho_a = 1$ for $y > 0$	$\mathbf{v} = \begin{pmatrix} 2\frac{\rho_a+\rho_b-\rho}{\rho_a+\rho_b} \\ 0 \end{pmatrix}$	$\psi = \frac{1}{\rho_a+\rho_b} + 2y\frac{\rho_a+\rho_b-\rho}{\rho_a+\rho_b}$

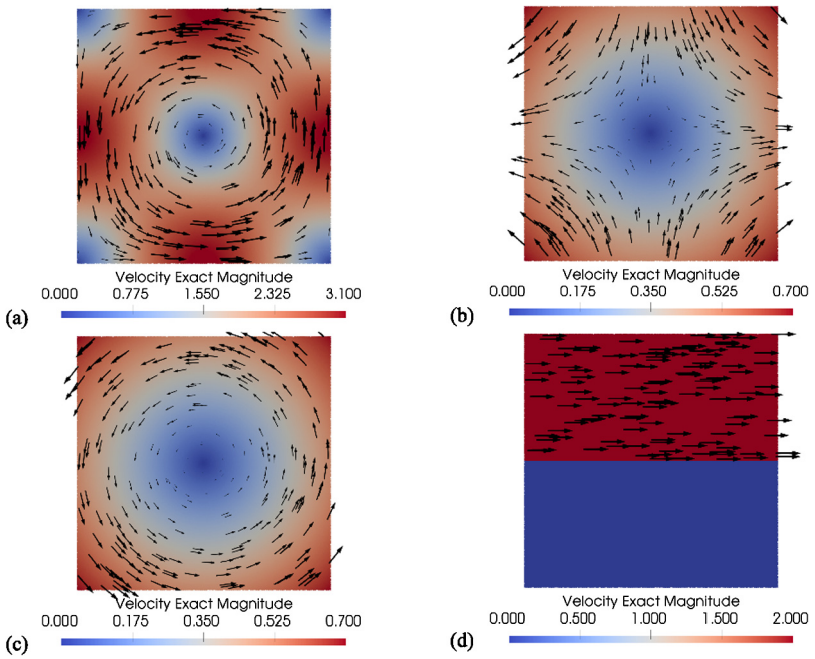

Fig. 2. Velocity fields for (a) sinusoidal rotation, (b) stagnation point, (c) solid body rotation, and (d) discontinuous flow cases.

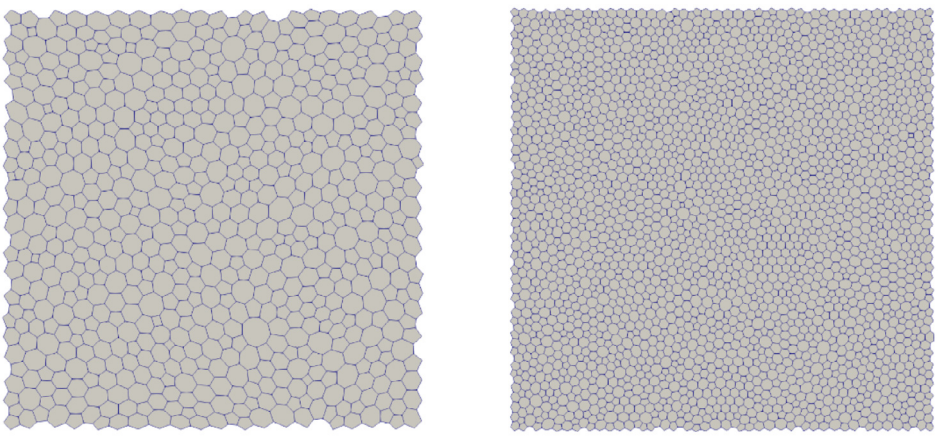

Fig. 3. Mesh discretization for the two coarsest meshes used in the tests.

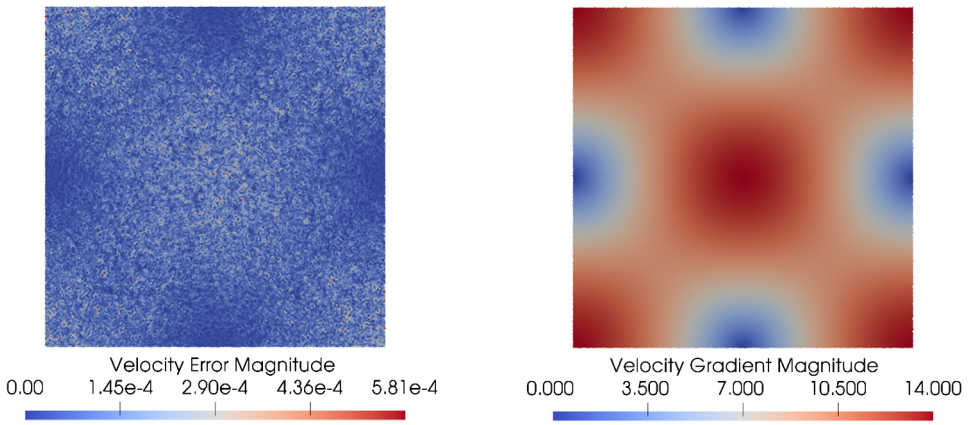

Fig. 4. Contour plots of the velocity error magnitude (left) and the analytic velocity gradient magnitude (right) for the finest resolution sinusoidal rotation run (case A).

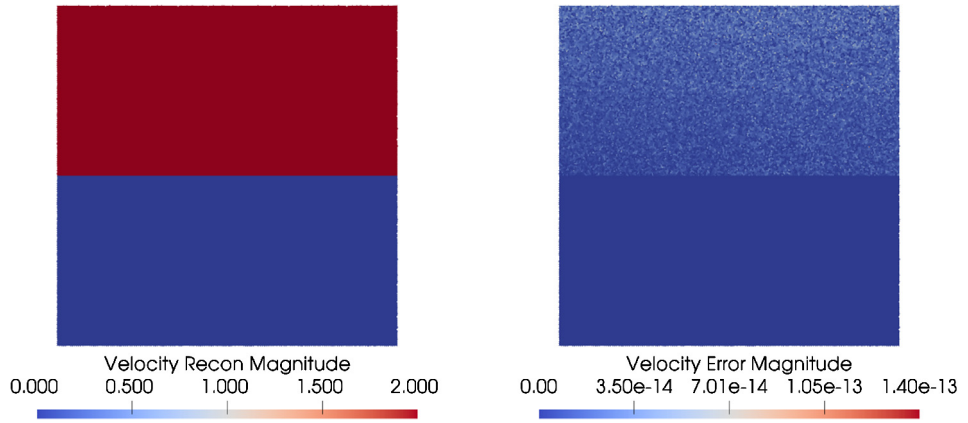

Fig. 5. Contour plot of the reconstructed velocity magnitude and error magnitude for the finest resolution Discontinuous Flow run (case D).

4.2. Reconstruction error

The mean velocity reconstructed from the face flux was compared to the exact known solution for each test case. The error in the velocity solution for the Level 2 reconstruction of the sinusoidal rotation case on the finest mesh is shown on the left side of Fig. 4. On the right in Fig. 4 is a contour plot of the analytic velocity gradient magnitude (defined as $\sqrt{\frac{\partial u_i}{\partial x_j}\frac{\partial u_i}{\partial x_j}}$) for the same case. Both plots show a clear "X" pattern with the regions of highest error corresponding to the regions of highest velocity gradient. This is a qualitative demonstration that the reconstruction error is proportional to the first derivatives and is first order accurate.

In Fig. 5, the reconstructed velocity solution for the discontinuous flow case is shown next to a contour plot of its error which is on the order of magnitude of machine precision. This shows the ability of the method to exactly reconstruct piecewise constant discontinuous flow. This ability is due to the mimetic nature of the method which forces continuity of the velocity across faces but allows discontinuous tangential velocity on either side of a face. Each cell solution is independent of the neighboring velocity behavior.

A convergence study was performed on each of the four cases using the four grids (with the coarsest two discretizations shown in Fig. 3). The error norm of the velocity was calculated using the relation: $\|\varepsilon\| = \sqrt{\frac{\sum_p |(\mathbf{u}_p^{exact} - \mathbf{u}_p^{recon})|^2 V_p}{V_{total}}}$ where the velocity at its exact value are evaluated at the polygon centroid (2nd order midpoint rule). The errors were plotted against the average mesh distance between polygon centers on a log-log plot, and these results are shown in Fig. 6. Also in the plot is a baseline slope of 1 for convergence rate comparison. Note that the errors for the discontinuous flow case were not plotted as they were within machine precision for all test grids. The plot indicates that the first three test cases approach first order convergence. First order convergence is consistent with the fundamental assumption of the interpolation, which is equation (1c), that assumes the normal velocity is constant on each polygon face. This the fundamental assumption of all lowest-order Raviart-Thomas elements.

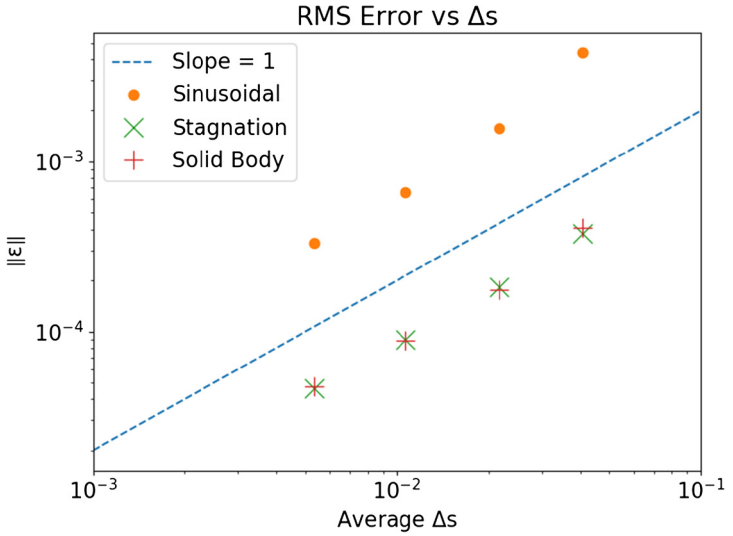

Fig. 6. Log-log plot of RMS velocity magnitude error for each flow condition compared to an exact first order convergence slope.

4.3. PDE solution

In PDE solution methods, reconstruction is more often found in its symmetric form, as a discrete vector inner-product, or a discrete Hodge star operation. This is the focus of section 3. Interestingly, there is error cancelation in this combination, and the lowest-order mimetic methods are second order accurate when solving PDEs on nearly uniform meshes. The discrete Hodge star (or inner product) operation takes in the normal components on mesh faces and produces the tangential component along dual mesh edges. The operator is a square, symmetric matrix. This section will evaluate this matrix and its error.

To do this we look at the discrete vector Laplacian problem

$$\nabla \times (\rho \nabla \times \psi) = \tilde{\boldsymbol{\omega}} \tag{18}$$

In discretized form, this PDE problem requires a discrete Hodge star, or vector inner product. Using the current reconstruction method, this problem is discretized as,

$$\mathbf{C}^T[\mathbf{R}^T \rho \mathbf{V}^{-1}\mathbf{R}]\mathbf{C}\psi_c = V_c \tilde{\boldsymbol{\omega}}_c \tag{19}$$

and $\mathbf{R}^T \rho \mathbf{V}^{-1}\mathbf{R}$ is the discrete Hodge star that takes primary mesh 2-forms (fluxes on faces) to dual mesh 1-forms (integrals along line segments on the dual mesh). The curl operations, $\mathbf{C}$ and $\mathbf{C}^T$ are exact implementations of the Stokes curl theorem. ψ_c is the streamfunction at each polygon corner (pointing out of the 2D plane) and $\mathbf{C}\psi_c$ is the difference of two corner values on each polygon face. The velocity is recovered from $\mathbf{u} = \nabla \times \psi$ which is the discrete expression $\mathbf{u}_p = V^{-1}\mathbf{R}\mathbf{C}\psi_c$, because $\mathbf{C}\psi_c$ produces the exact flux.

Equation (19) was solved for each of the four cases, and the stream function solution error ($e_\psi = [\frac{1}{N}\sum(\psi_c - \psi_c^{exact})^2]^{1/2}$) is shown in Fig. 7 as a function of average polygon centroid spacing. The plot shows second order convergence for the sinusoidal rotation, stagnation flow and solid body rotation, cases. The solution for the discontinuous density case (case D) is not shown as the method is able to solve this case exactly. The stagnation point case (case B) is a Laplace equation solution because the right-hand side of equation (18) is zero for that flow case.

The exact eigenvalues of the Laplacian on a rectangle are known analytically (which is why the problem was chosen). The eigenvalues for the discrete operators (for the 3 different meshes for which eigenvalues were calculated) should be very close to the exact eigenvalues. Fig. 8 shows a plot of the sorted eigenvalues of the discrete Laplacian matrix for the three coarsest discretizations compared to the exact eigenvalues ($\lambda_{exact} = \pi^2(m^2+n^2)$ for $n = 1, 2, \dots$ and $m = 1, 2, \dots$) [51]. The plot shows that for level 2 reconstruction (4 harmonic basis functions, or 6 results per polygon after applying the matrix R), increasing the mesh discretization improves the number of accurately captured eigenvalues of the solution matrix. Note that the very smallest eigenvalues converge exponentially quickly to the exact answers as the mesh resolution improves, but they are not exact.

In contrast, the level 1 Hodge star reconstruction fails (only 2 harmonic basis functions which recover only the mean velocity, or 4 results per polygon after applying the matrix R). It shows a large number of nearly zero eigenvalues, showing that this Hodge star is singular, or very close to singular. The level 1 reconstruction has spurious solution modes. In the linguistics of the VEM, it does not have any stabilization term. Eigenvalues are the natural frequencies captured by a method, and it can be inferred from this plot that increasing the mesh discretization increases the method's ability to capture higher

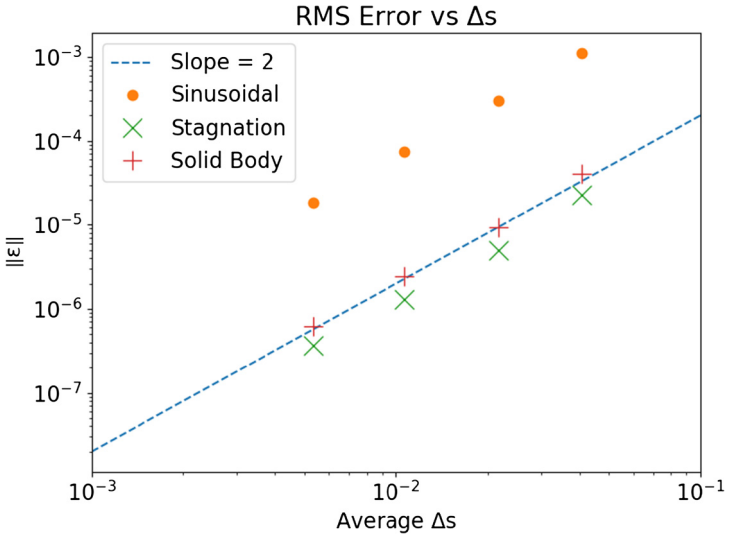

Fig. 7. Log-log plot of RMS stream function error for each flow condition compared to an exact second order convergence slope.

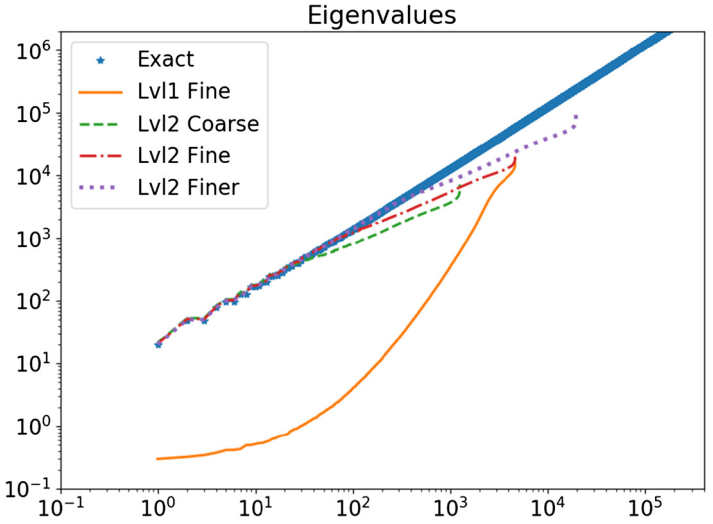

Fig. 8. Eigenvalues vs index for the Laplacian solution matrix on a log-log scale.

frequency solution modes for level 2 reconstruction, whereas the level 1 reconstruction is unable to capture even the largest frequency (smallest eigenvalue) modes. Note that level 1 reconstruction is known to be effective (and non-singular) on triangular meshes, but it is not sufficient for quadrilateral or more-sided polygon meshes.

4.4. Moving polygon Navier-Stokes

The streamfunction solution described above can be expanded to solve the incompressible and inviscid Navier-Stokes equations on a moving polygonal mesh with variable density. In rotational form this is $\nabla \times \left(\rho \frac{D\mathbf{u}}{Dt}\right) = \nabla \times [\rho \mathbf{g} - \nabla p]$ with $\mathbf{u} = \nabla \times \psi$. In this test case, the following finite volume discretized equation for the streamfunction will be solved,

$$\{\mathbf{C}^T[\mathbf{R}^T \rho \mathbf{V}^{-1}\mathbf{R}]^{n+1}\mathbf{C}\}\psi_c^{n+1} = \mathbf{C}^T[\mathbf{R}^T\mathbf{V}^{-1}]^{n+1}[\rho \mathbf{u}_p^n + \Delta t \rho \mathbf{g}] \tag{20}$$

In Equation (20), the unknown is ψ_c^{n+1}, and the right hand side is composed of the density weighted vorticity at time-n and body forces on the polygon volume (typically gravity). Everything to the left of the stream function, ψ_c^{n+1}, needs to be formally inverted by the solver, meaning the discrete vector Laplacian matrix $\mathbf{C}^T[\mathbf{R}^T \rho \mathbf{V}^{-1}\mathbf{R}]\mathbf{C}$ needs to be formally inverted for each timestep.

Here, the compound operation, **RC**, represents a discrete curl operation, where the **C** operation converts the stream function at corner points into a face flux along the polygon edge exactly and **R** reconstructs the velocity field from the face

fluxes. Symmetrically, the operation $\mathbf{C}^T\mathbf{R}^T$ is the curl operation which constructs line integrals and polygon-corner vorticity from the polygon velocity.

Equation (20) has been implemented into a full transient solver to test the usefulness of the lowest order mimetic polygon reconstruction for a full Navier-Stokes simulation. Flow was initialized at a non-equilibrium state for a standing water wave, and gravity was applied to each polygon cell. Equation (20) was solved for the stream function and the polygon centroid velocities were reconstructed from the streamfunction using $\mathbf{u}_p = \mathbf{V}^{-1}\mathbf{R}\mathbf{C}\psi_c$. The polygons were then advected with this velocity, and new polygons were partitioned using a Voronoi methodology [11,50].

Shown in Fig. 9 is the free surface simulation results for a sloshing wave in a square box of length $L = 1$, compared to an OpenFOAM solution on a stationary Cartesian mesh. The vertical velocity solution is shown in Fig. 10. The two materials, air and water, were initialized, above and below the curve $y = \frac{L}{2} + 0.1L\cos\left(2\pi\frac{x}{L}\right)$ respectively, with zero initial starting velocity and then allowed to evolve due to gravity forces and their density difference.

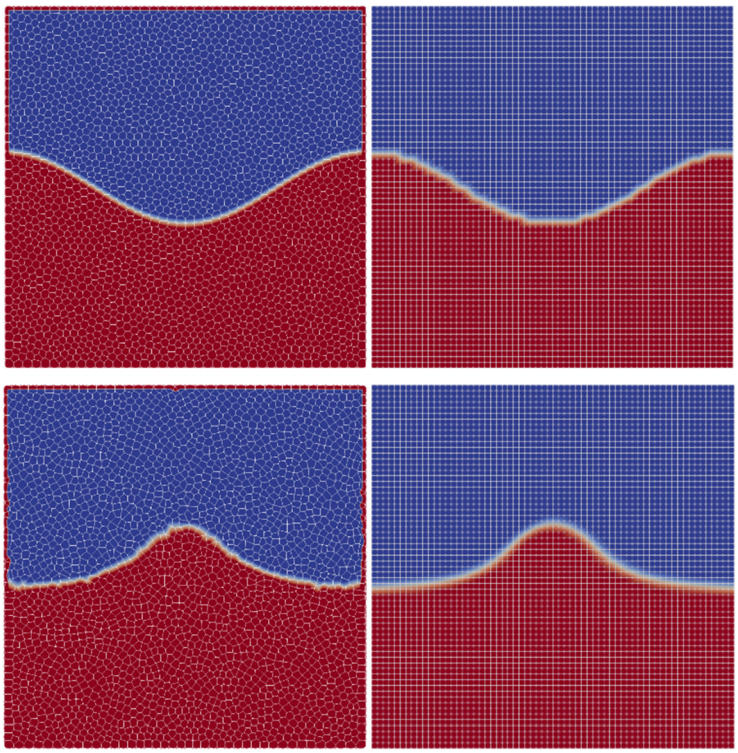

Fig. 9. Comparison of the current polygon reconstruction method implemented into a moving-mesh Navier-Stokes solver (left side) compared to an OpenFOAM Volume-of-fluid solution (right side) on a fixed Cartesian mesh. Cells are colored by density, with red indicating water, and blue indicating air. Top row is the initial condition and bottom row is the solution after half a period.

The polygon simulation was initialized on a polygon mesh with a total of 2431 cells and an average polygon size of $0.021L$. Slip conditions were applied at all four walls. The OpenFOAM simulation was run with a linear upwinding convection scheme, using the interFoam solver. While discrete cells of uniform material were used in the polygon mesh, the OpenFOAM solution used a Volume of Fluids (VOF) method to solve for the cell water fraction (α). The free surface is typically approximated as an iso-surface at $\alpha = 0.5$. OpenFOAM was run on a uniform 64×64 Cartesian mesh with almost twice as many cells (4086) as the polygon mesh, and with a mesh spacing of $\Delta x = 0.016L$. Both cases were run with all four slip walls.

The material interface of the solution using the reconstruction method evolves similarly to the OpenFOAM solution, and the velocity solution (Fig. 10) also shows good agreement. The OpenFOAM solutions show some spuriously high velocity on the domain boundary where the density interface meets it.

5. Discussion

This work shows a mimetic method for the reconstruction of vector fields on polygons. The reconstruction is a valid extension of lowest-order Raviart-Thomas elements to polygons using a truncated series of harmonic polynomials. The approach can also be interpreted as a specific type of virtual element method. The method produces classic Raviart-Thomas interpolations for triangles and Cartesian meshes. Given the popularity of the lowest-order Raviart-Thomas elements, this work extends their applicability.

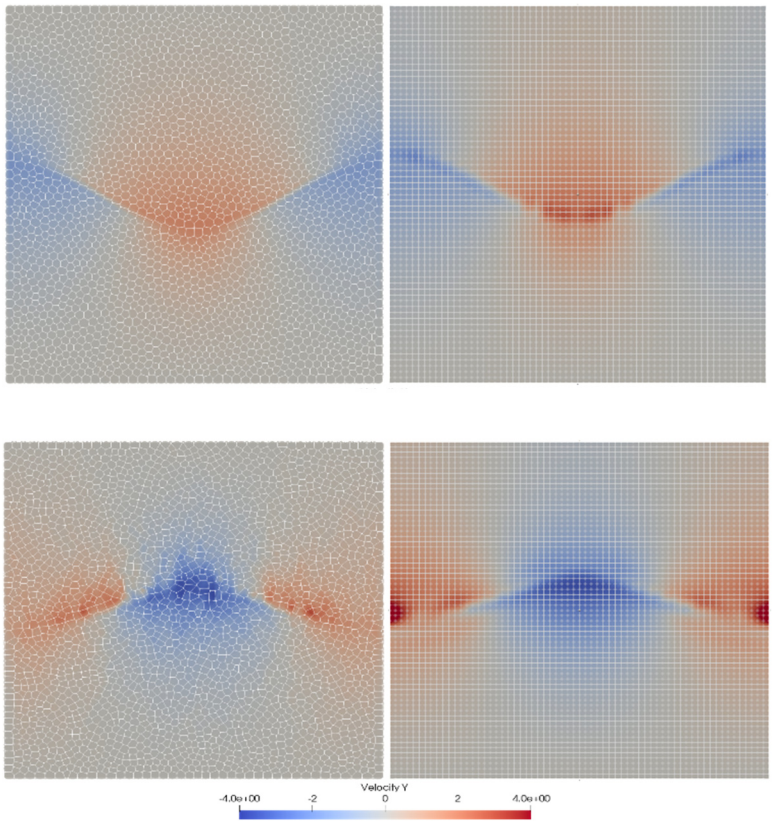

Fig. 10. Comparison of the reconstruction method implemented into a moving mesh Navier-Stokes solver (left side) compared to a stationary Cartesian mesh OpenFOAM VOF solution (right side). Cells are colored by vertical velocity. The vertical velocity solution is smoother for the mimetic method (left side).

The mathematical analysis of the interpolation showed its relationship to full harmonic function (not polynomial) interpolation. The method is a truncated harmonic polynomial approximation to the harmonic interpolant. At this lowest order, the interpolant is curl-free in the polygon and has a constant divergence. In addition, the normal component of the vector field is assumed constant on each face, and that value is the input data of the interpolation. For a second-order interpolant this method would be generalized to allow for a divergence that was linear and for a constant curl, with linear normal velocity on each polygon face.

The truncation of the harmonic interpolant to harmonic polynomials allows for different levels of truncation. This work primarily explored level 1, 2, and 3 truncation. Increasing the level of the truncation, does not improve the order of accuracy of the method. The order of the method is dictated by the assumption of constant normal velocity on each face, irrespective of the accuracy of the harmonic interpolant. However, level 1 truncation (piecewise constant velocity fields) was shown to be insufficient on polygons with more sides than 3 (i.e. on anything other than triangles) because it does not pass the patch test and produces zero eigenvalues, or spurious modes. Level 2 truncation which recovers the symmetric part of the velocity gradient tensor along with the average velocity, was sufficient for stability on all the meshes we tested. Higher levels of truncation add more work with little pay-off. It is possible that on highly distorted polygon meshes with a large numbers of faces per polygon, higher levels of truncation might add some stability.

Test cases showed the reconstruction on its own to be first order accurate. They also showed that locally constant, but globally discontinuous vector fields (piecewise constant velocity fields) can be interpolated exactly. Most importantly, the test cases showed that despite the fact that the local interpolation within each polygon is curl-free, the method captures globally non-harmonic functions well. In fact the accuracy of the stagnation point flow (with zero vorticity) and the solid body rotation flow (with vorticity everywhere) is virtually identical. In addition, in the important case of PDE solution, where the reconstruction and its transpose are always used together (as a vector inner product, or discrete Hodge star), the PDE method is second-order accurate on nearly uniform meshes.

Finally we showed the application of this polygonal reconstruction method to a reasonably complex moving mesh problem that computed sloshing waves with a density jump of 1000. This shows the potential of mimetic polygonal reconstruction methods to be used in real-world simulation problems and Lagrangian particle methods. Future work will extend this approach to higher order and 3D polyhedra.

CRediT authorship contribution statement

J. Blair Perot: Conceptualization, Software, Supervision, Writing - review & editing. **Chris Chartrand:** Software, Visualization, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The first author was supported, in part, by the National Science Foundation (grant No. 1353942) and DARPA (grant No. DARPA-SN-19-74). The second author was supported, in part, by Sandia National Laboratories.

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

Appendix A

A.1. Integration formulas

The matrices **R** and **V** require the calculation of various geometric integrals. All the cell integrals can actually be found by computing face integrals. In 2D, faces integrals are one dimensional and can be evaluated with classic quadrature formulas. For specific order polynomials (like the non-orthogonal harmonic polynomials) we have,

$$mV_p = \int (x_i)_{,i} dV = \sum_{f \subset b(\Omega_p)} mV_f$$

$$(m+1)V_p \overline{x_j}^p = \int (x_i x_j)_{,i} dV = \sum_{f \subset b(\Omega_p)} \overline{x_j}^f mV_f$$

$$(m+2)V_p \overline{x_j x_k}^p = \int (x_i x_j x_k)_{,i} dV = \sum_{f \subset b(\Omega_p)} \overline{x_j x_k}^f mV_f$$

$$(m+3)V_p \overline{x_j x_k x_m}^p = \int (x_i x_j x_k x_m)_{,i} dV = \sum_{f \subset b(\Omega_p)} \overline{x_j x_k x_m}^f mV_f$$

and so on.

The orthogonal polynomials have to be decomposed into their different order terms. These polynomials have the position origin at the center of gravity.

Level-1 $\overline{P_1}^p = \overline{Q_1}^p = 0$.

Level-2 $\overline{P_2}^p = \frac{2}{4}\sum_f \overline{P_2}^f \frac{V_f}{V_p}$ and $\overline{Q_2}^p = \frac{2}{4}\sum_f \overline{Q_2}^f \frac{V_f}{V_p}$ for 2D problems.

Level-3 $P_3 = (x - c_x)(P_2 - 3\overline{P_2}) - (y - c_y)(Q_2 - 3\overline{Q_2})$ so separate by order into,

$$P_3 = (xP_2 - yQ_2) - (c_x P_2 - c_y Q_2) - 3(x\overline{P_2} - y\overline{Q_2}) + 3(c_x\overline{P_2} - c_y\overline{Q_2})$$

$$Q_3 = (yP_2 + xQ_2) - (c_y P_2 + c_x Q_2) - 3(y\overline{P_2} + x\overline{Q_2}) + 3(c_y\overline{P_2} + c_x\overline{Q_2})$$

so $\overline{P_3}^p = \overline{(xP_2 - yQ_2)}^p + 2(c_x\overline{P_2}^p - c_y\overline{Q_2}^p)$

$$\overline{Q_3}^p = \overline{(yP_2 + xQ_2)}^p + 2(c_y\overline{P_2}^p + c_x\overline{Q_2}^p)$$

and since the first term is 3rd order.

$$\overline{P_3}^p = \frac{2}{5}\sum_f \overline{(xP_2 - yQ_2)}^f \frac{V_f}{V_p} + 2(c_x\overline{P_2}^p - c_y\overline{Q_2}^p)$$

Similarly $\mathbf{c} = \frac{\frac{3}{2}\overline{\mathbf{x}(\mathbf{x}^2)}^p}{\overline{\mathbf{x}^2}^p} = \frac{6\overline{\mathbf{x}(\mathbf{x}^2)}^p}{4\overline{\mathbf{x}^2}^p}$ becomes

$$\mathbf{c} = \left\{ \frac{3}{2}\frac{2}{5} \sum_f \overline{\mathbf{x}(\mathbf{x}^2)}^f \frac{V_f}{V_p} \right\} / \left\{ \sum_f \overline{\frac{1}{2}\mathbf{x}^2}^f \frac{V_f}{V_p} \right\} = \frac{6}{5} \left\{ \frac{1}{V_p} \sum_f \overline{\mathbf{x}(\mathbf{x}^2)}^f 2V_f \right\} / \left\{ \frac{1}{V_p} \sum_f \overline{\mathbf{x}^2}^f 2V_f \right\}$$

to compute **c** from face data.

Also note that because the functions are harmonic (and therefore the Laplacian is zero), the diagonal terms in the matrix **V**, can be computed from face data.

$$\int_{\Omega_p} \nabla P_i \cdot \nabla P_i dV = \sum_{f \subset b(\Omega_p)} \int_f P_i \nabla P_i \cdot \hat{\mathbf{n}} dA = \sum_{f \subset b(\Omega_p)} \int_f \nabla (\frac{1}{2} P_i P_i) \cdot \hat{\mathbf{n}} dA$$

which allows the 2D integrals to be evaluated using a sum of simpler 1D face integrals. A similar expression holds for the Q pair and gives the exact same final numerical answer. For the case of the level 3 polynomials in the equation above, this would require a 4th order polynomial cell integration on the left hand side, or a 4th order polynomial face integration on the right hand side (not 5th order, because the normal removes an order).

An alternative approach is to integrate polygon integrals by subdividing the polygon into triangles and applying triangle quadrature rules. However, this can become more expensive as the polynomial order increases. For example 4th order polynomials require a minimum of 6-points per triangle to integrate.

In 2D the face integrals are one-dimensional. Simpson rule (3-point) can be used up to cubic terms. That would take care of most integration for up to level-3 orthogonal polynomials. Except one of the diagonal values in **V**. 4-point Gauss-Lobatto (1/12 end points and 5/12 at $\pm L\sqrt{\frac{1}{20}}$) is good up to 5th-order polynomials. 3-point Gauss is also 6th order, and so exact for 5th-order polynomials. This is a weight of 8/18 in center and a weight of 5/18 at $\pm L\sqrt{\frac{3}{20}}$ from the center.

Perhaps even easier than numerical integration on the faces is to simply work out the integrals analytically (perhaps using a tool like Mathematica) for an arbitrarily oriented line segment.

A.2. Orthogonal polynomials

The required orthogonal harmonic polynomials are orthogonal in the slightly unusual sense that their gradients are orthogonal on average over the polygon. The gradients of the orthogonal harmonic polynomials are

$$\nabla P_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \nabla Q_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$\nabla P_2 = 2\begin{pmatrix} x \\ -y \end{pmatrix} \quad \nabla Q_2 = 2\begin{pmatrix} y \\ x \end{pmatrix}$$

$$\nabla P_3 = \begin{pmatrix} (P_2 - 3\overline{P_2}^p) \\ -(Q_2 - 3\overline{Q_2}^p) \end{pmatrix} + (x - c_x)\nabla P_2 - (y - c_y)\nabla Q_2$$

$$\nabla Q_3 = \begin{pmatrix} (Q_2 - 3\overline{Q_2}^p) \\ (P_2 - 3\overline{P_2}^p) \end{pmatrix} + (y - c_y)\nabla P_2 + (x - c_x)\nabla Q_2$$

Clearly $\nabla P_i \cdot \nabla Q_i = 0$ for $i = 1$ and 2. This is a statement of local orthogonality at every point in the polygon. The first and second level are therefore locally orthogonal to themselves.

The second level is also orthogonal to the first level because all products give a function that is proportional to x or y. The integrals of these functions give the centroid values for the position. And by construction the centroid is at the origin of the coordinate system and therefore has a value of zero.

The third level orthogonal polynomials are orthogonal to the first level, if their average value (0th moment) on the polygon is zero.

$$\overline{\nabla P_3}^p = -2\begin{pmatrix} \overline{P_2}^p \\ -\overline{Q_2}^p \end{pmatrix} + \overline{x\nabla P_2 - y\nabla Q_2}^p - c_x\overline{\nabla P_2}^p + c_y\overline{\nabla Q_2}^p$$

$$\overline{\nabla Q_3}^p = -2\begin{pmatrix} \overline{Q_2}^p \\ \overline{P_2}^p \end{pmatrix} + \overline{y\nabla P_2 + x\nabla Q_2}^p - c_y\overline{\nabla P_2}^p - c_x\overline{\nabla Q_2}^p$$

which is

$$\overline{\nabla P_3}^p = -2\begin{pmatrix} \overline{P_2}^p \\ -\overline{Q_2}^p \end{pmatrix} + 2\overline{x\begin{pmatrix} x \\ -y \end{pmatrix} - y\begin{pmatrix} y \\ x \end{pmatrix}}^p - 2c_x\overline{\begin{pmatrix} x \\ -y \end{pmatrix}}^p + 2c_y\overline{\begin{pmatrix} y \\ x \end{pmatrix}}^p = 0$$

$$\overline{\nabla Q_3}^p = -2\begin{pmatrix} \overline{Q_2}^p \\ \overline{P_2}^p \end{pmatrix} + 2\overline{y\begin{pmatrix} x \\ -y \end{pmatrix} + x\begin{pmatrix} y \\ x \end{pmatrix}}^p - 2c_y\overline{\begin{pmatrix} x \\ -y \end{pmatrix}}^p - 2c_x\overline{\begin{pmatrix} y \\ x \end{pmatrix}}^p = 0$$

The first two terms are equal to each other and the last two are zero via the centroid origin.

The third level orthogonal polynomials are orthogonal to the second level because

$$\overline{\nabla P_3 \cdot \nabla P_2} = \overline{\begin{pmatrix} (P_2 - 3\overline{P_2}^p) \\ -(Q_2 - 3\overline{Q_2}^p) \end{pmatrix} \cdot 2\begin{pmatrix} x \\ -y \end{pmatrix} + (x-c_x)2\begin{pmatrix} x \\ -y \end{pmatrix} \cdot 2\begin{pmatrix} x \\ -y \end{pmatrix} - (y-c_y)2\begin{pmatrix} y \\ x \end{pmatrix} \cdot 2\begin{pmatrix} x \\ -y \end{pmatrix}}$$

$$\overline{\nabla P_3 \cdot \nabla P_2} = 2\overline{x(x^2+y^2)} + 4\overline{(x-c_x)(x^2+y^2)}$$

which is zero if $c_x\overline{(x^2+y^2)} = \frac{3}{2}\overline{x(x^2+y^2)}$.

Similarly

$$\overline{\nabla P_3 \cdot \nabla Q_2} = \overline{\begin{pmatrix} (P_2 - 3\overline{P_2}^p) \\ -(Q_2 - 3\overline{Q_2}^p) \end{pmatrix} \cdot 2\begin{pmatrix} y \\ x \end{pmatrix} + (x-c_x)2\begin{pmatrix} x \\ -y \end{pmatrix} \cdot 2\begin{pmatrix} y \\ x \end{pmatrix} - (y-c_y)2\begin{pmatrix} y \\ x \end{pmatrix} \cdot 2\begin{pmatrix} y \\ x \end{pmatrix}}$$

$$\overline{\nabla P_3 \cdot \nabla Q_2} = -2\overline{y(x^2+y^2)} - 4\overline{(y-c_y)(x^2+y^2)}$$

which is zero if $c_y\overline{(x^2+y^2)} = \frac{3}{2}\overline{y(x^2+y^2)}$.

The two conditions are summarized in the vector expression $\mathbf{c} = \frac{3\overline{\mathbf{x}(\mathbf{x}^2)}^p}{2\overline{\mathbf{x}^2}^p}$.

Also for the level 3 Q polynomial

$$\overline{\nabla Q_3 \cdot \nabla P_2} = \overline{\begin{pmatrix} (Q_2 - 3\overline{Q_2}^p) \\ (P_2 - 3\overline{P_2}^p) \end{pmatrix} \cdot 2\begin{pmatrix} x \\ -y \end{pmatrix} + (y-c_y)2\begin{pmatrix} x \\ -y \end{pmatrix} \cdot 2\begin{pmatrix} x \\ -y \end{pmatrix} + (x-c_x)2\begin{pmatrix} y \\ x \end{pmatrix} \cdot 2\begin{pmatrix} x \\ -y \end{pmatrix}}$$

$$\overline{\nabla Q_3 \cdot \nabla P_2} = 2\overline{y(x^2+y^2)} + 4\overline{(y-c_y)(x^2+y^2)} = -\overline{\nabla P_3 \cdot \nabla Q_2}$$

which is zero because this was set above.

$$\overline{\nabla Q_3 \cdot \nabla Q_2} = \overline{\begin{pmatrix} (Q_2 - 3\overline{Q_2}^p) \\ (P_2 - 3\overline{P_2}^p) \end{pmatrix} \cdot 2\begin{pmatrix} y \\ x \end{pmatrix} + (y-c_y)2\begin{pmatrix} x \\ -y \end{pmatrix} \cdot 2\begin{pmatrix} y \\ x \end{pmatrix} + (x-c_x)2\begin{pmatrix} y \\ x \end{pmatrix} \cdot 2\begin{pmatrix} y \\ x \end{pmatrix}}$$

$$\overline{\nabla Q_3 \cdot \nabla Q_2} = 2\overline{x(x^2+y^2)} + 4\overline{(x-c_x)(x^2+y^2)} = \overline{\nabla P_3 \cdot \nabla P_2}$$

which is zero because this was also set above.

Finally, using $\nabla P_2 \cdot \nabla Q_2 = 0$ and $\nabla P_2 \cdot \nabla P_2 = \nabla Q_2 \cdot \nabla Q_2$ then

$$\nabla P_3 \cdot \nabla Q_3 = \{(x-c_x)\nabla P_2 - (y-c_y)\nabla Q_2\} \cdot \begin{pmatrix} (Q_2 - 3\overline{Q_2}^p) \\ (P_2 - 3\overline{P_2}^p) \end{pmatrix} + \begin{pmatrix} (P_2 - 3\overline{P_2}^p) \\ -(Q_2 - 3\overline{Q_2}^p) \end{pmatrix} \cdot \{(x-c_x)\nabla Q_2 + (y-c_y)\nabla P_2\}$$

and

$$\begin{aligned} \nabla P_3 \cdot \nabla Q_3 = \{&(x-c_x)[\nabla P_2 \cdot \begin{pmatrix} (Q_2 - 3\overline{Q_2}^p) \\ (P_2 - 3\overline{P_2}^p) \end{pmatrix} + \nabla Q_2 \begin{pmatrix} (P_2 - 3\overline{P_2}^p) \\ -(Q_2 - 3\overline{Q_2}^p) \end{pmatrix}] \\ &- (y-c_y)[\nabla Q_2 \cdot \begin{pmatrix} (Q_2 - 3\overline{Q_2}^p) \\ (P_2 - 3\overline{P_2}^p) \end{pmatrix} - \nabla P_2 \cdot \begin{pmatrix} (P_2 - 3\overline{P_2}^p) \\ -(Q_2 - 3\overline{Q_2}^p) \end{pmatrix}] = (x-c_x)[0] - (y-c_y)[0] \end{aligned}$$

because ∇P_2 is ∇Q_2 rotated 90 degrees. So $\nabla P_i \cdot \nabla Q_i = 0$ for all i (not just $i = 1$ and 2).

This level 3 orthogonality with each other is true locally at every point in the polygon (as in the level 1 and 2 cases) and does not even require a polygon average to be enforced.

References

[1] A. Palha, P.P. Rebelo, R. Hiemstra, J. Kreeft, M. Gerritsma, Physics-compatible discretization techniques on single and dual grids, with application to the Poisson equation of volume forms, J. Comput. Phys. 257 (2014) 1394–1422.
[2] A. Palha, B. Koren, F. Felici, A mimetic spectral element solver for the Grad–Shafranov equation, J. Comput. Phys. 316 (2016) 63–93.
[3] A. Palha, M. Gerritsma, A mass, energy, enstrophy and vorticity conserving (MEEVC) mimetic spectral element discretization for the 2D incompressible Navier–Stokes equations, J. Comput. Phys. 328 (2017) 200–220.
[4] D. Lee, A. Palha, M. Gerritsma, Discrete conservation properties for shallow water flows using mixed mimetic spectral elements, J. Comput. Phys. 357 (2018) 282–304.
[5] P.B. Bochev, J.M. Hyman, Principles of mimetic discretizations of differential operators, in: Compatible Spatial Discretizations, Springer, 2006, pp. 89–119.
[6] J.M. Hyman, M. Shashkov, The orthogonal decomposition theorems for mimetic finite difference methods, SIAM J. Numer. Anal. 36 (1999) 788–818.
[7] L.G. Margolin, M. Shashkov, Finite volume methods and the equations of finite scale: a mimetic approach, Int. J. Numer. Methods Fluids 56 (2008) 991–1002.
[8] K. Lipnikov, M. Shashkov, I. Yotov, Local flux mimetic finite difference methods, Numer. Math. 112 (2009) 115–152.
[9] K. Lipnikov, G. Manzini, M. Shashkov, Mimetic finite difference method, J. Comput. Phys. 257 (2014) 1163–1227.
[10] R.A. Nicolaides, Incompressible Computational Fluid Dynamics, Cambridge Univ. Press, Cambridge, 1993.
[11] J.B. Perot, Conservation properties of unstructured staggered mesh schemes, J. Comput. Phys. 159 (2000) 58–89.
[12] 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.
[13] V. Subramanian, J.B. Perot, Higher-order mimetic methods for unstructured meshes, J. Comput. Phys. 219 (2006) 68–85.
[14] P.A. Raviart, J.M. Thomas, A mixed finite element method for 2-nd order elliptic problems, in: Mathematical Aspects of Finite Element Methods, Berlin, in: Lecture Notes in Mathematics, 1977.
[15] J.-C. Nédélec, Mixed finite elements in $\mathbb{R}$ 3, Numer. Math. 35 (1980) 315–341.
[16] C.A. Hall, J.S. Peterson, T.A. Porsching, F.R. Sledge, The dual variable method for finite element discretizations of Navier/Stokes equations, Int. J. Numer. Methods Eng. 21 (1985) 883–898.
[17] G. Rodrigue, D. White, A vector finite element time-domain method for solving Maxwell's equations on unstructured hexahedral grids, SIAM J. Sci. Comput. 23 (2001) 683–706.
[18] A. Bossavit, On the geometry of electromagnetism (1): Euclidean space, 日本AEM学会誌 (Journal of the Japan Society of Applied Electromagnetics) 6 (1998) 17–28.
[19] A. Bossavit, On the geometry of electromagnetism (2): geometrical objects, 日本AEM学会誌 (Journal of the Japan Society of Applied Electromagnetics) 6 (1998) 114–123.
[20] A. Bossavit, On the geometry of electromagnetism (3): Faraday's law, 日本AEM学会誌 (Journal of the Japan Society of Applied Electromagnetics) 6 (1998) 233–240.
[21] A. Bossavit, On the geometry of electromagnetism (4): Maxwell's house, 日本AEM学会誌 (Journal of the Japan Society of Applied Electromagnetics) 6 (1998) 318–326.
[22] D.N. Arnold, R.S. Falk, R. Winther, Finite element exterior calculus, homological techniques, and applications, Acta Numer. 15 (2006) 1–155.
[23] J.B. Perot, C.J. Zusi, Differential forms for scientists and engineers, J. Comput. Phys. 257 (2014) 1373–1393.
[24] 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.
[25] J.B. Perot, D. Vidovic, P. Wesseling, Mimetic reconstruction of vectors, in: Compatible Spatial Discretizations, Springer, 2006, pp. 173–188.
[26] E. Wachspress, Rational bases for convex polyhedra, Comput. Math. Appl. 59 (2010) 1953–1956.
[27] M.S. Floater, Mean value coordinates, Comput. Aided Geom. Des. 20 (2003) 19–27.
[28] K. Hormann, M.S. Floater, Mean value coordinates for arbitrary planar polygons, ACM Trans. Graph. (TOG) 25 (2006) 1424–1441.
[29] U. Pinkall, K. Polthier, Computing discrete minimal surfaces and their conjugates, Exp. Math. 2 (1993) 15–36.
[30] M.S. Floater, K. Hormann, G. Kós, A general construction of barycentric coordinates over convex polygons, Adv. Comput. Math. 24 (2006) 311–331.
[31] D. Anisimov, Analysis and new constructions of generalized barycentric coordinates in 2D, PhD Thesis, Università della Svizzera italiana, 2017.
[32] M. Meyer, A. Barr, H. Lee, M. Desbrun, Generalized barycentric coordinates on irregular polygons, J. Graph. Tools 7 (2002) 13–22.
[33] N. Sukumar, B. Moran, T. Belytschko, The natural element method in solid mechanics, Int. J. Numer. Methods Eng. 43 (1998) 839–887.
[34] N. Sukumar, A. Tabarraei, Conforming polygonal finite elements, Int. J. Numer. Methods Eng. 61 (2004) 2045–2066.
[35] G. Manzini, A. Russo, N. Sukumar, New perspectives on polygonal and polyhedral finite element methods, Math. Models Methods Appl. Sci. 24 (2014) 1665–1699.
[36] S. Martin, P. Kaufmann, M. Botsch, M. Wicke, M. Gross, Polyhedral finite elements using harmonic basis functions, in: Computer Graphics Forum, 2008.
[37] L. Beirao da Veiga, F. Brezzi, A. Cangiani, G. Manzini, L.D. Marini, A. Russo, Basic principles of virtual element methods, Math. Models Methods Appl. Sci. 23 (2013) 199–214.
[38] F. Brezzi, K. Lipnikov, V. Simoncini, A family of mimetic finite difference methods on polygonal and polyhedral meshes, Math. Models Methods Appl. Sci. 15 (2005) 1533–1551.
[39] F. Brezzi, R.S. Falk, L.D. Marini, Basic principles of mixed virtual element methods, ESAIM Math. Model. Numer. Anal. 48 (2014) 1227–1240.
[40] L. Beirao da Veiga, F. Brezzi, L.D. Marini, A. Russo, Mixed virtual element methods for general second order elliptic problems on polygonal meshes, ESAIM Math. Model. Numer. Anal. 50 (2016) 727–747.
[41] L. Beirao da Veiga, F. Brezzi, L.D. Marini, A. Russo, H (div) and H (curl)-conforming virtual element methods, Numer. Math. 133 (2016) 303–332.
[42] A. Cangiani, G. Manzini, Flux reconstruction and solution post-processing in mimetic finite difference methods, Comput. Methods Appl. Mech. Eng. 197 (2008) 933–945.
[43] K. Lipnikov, M. Shashkov, D. Svyatskiy, The mimetic finite difference discretization of diffusion problem on unstructured polyhedral meshes, J. Comput. Phys. 211 (2006) 473–491.
[44] M. Shashkov, S. Steinberg, Support-operator finite-difference algorithms for general elliptic problems, J. Comput. Phys. 118 (1995) 131–151.
[45] L. Beirao da Veiga, K. Lipnikov, G. Manzini, The Mimetic Finite Difference Method for Elliptic Problems, vol. 11, Springer, 2014.
[46] P. Joshi, M. Meyer, T. DeRose, B. Green, T. Sanocki, Harmonic coordinates for character articulation, ACM Trans. Graph. (TOG) 26 (2007) 71.
[47] M. Ben-Chen, O. Weber, C. Gotsman, Variational harmonic maps for space deformation, ACM Trans. Graph. (TOG) 28 (2009) 34.
[48] 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.
[49] J.B. Perot, Discrete conservation properties of unstructured mesh schemes, Annu. Rev. Fluid Mech. 43 (2011) 299–318.
[50] 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.
[51] J.R. Kuttler, V.G. Sigillito, Eigenvalues of the Laplacian in two dimensions, SIAM Rev. 26 (1984) 163–193.