Construction of Arbitrary Order Finite Element Degree-of-Freedom Maps on Polygonal and Polyhedral Cell Meshes

MATTHEW W. SCROGGS and JØRGEN S. DOKKEN, Department of Engineering, University of Cambridge, United Kingdom
CHRIS N. RICHARDSON, BP Institute, University of Cambridge, United Kingdom
GARTH N. WELLS, Department of Engineering, University of Cambridge, United Kingdom

We develop a method for generating degree-of-freedom maps for arbitrary order Ciarlet-type finite element spaces for any cell shape. The approach is based on the composition of permutations and transformations by cell sub-entity. Current approaches to generating degree-of-freedom maps for arbitrary order problems typically rely on a consistent orientation of cell entities that permits the definition of a common local coordinate system on shared edges and faces. However, while orientation of a mesh is straightforward for simplex cells and is a local operation, it is not a strictly local operation for quadrilateral cells and, in the case of hexahedral cells, not all meshes are orientable. The permutation and transformation approach is developed for a range of element types, including arbitrary degree Lagrange, serendipity, and divergence- and curl-conforming elements, and for a range of cell shapes. The approach is local and can be applied to cells of any shape, including general polytopes and meshes with mixed cell types. A number of examples are presented and the developed approach has been implemented in open-source libraries.

CCS Concepts: • **Mathematics of computing** → **Discretization**; *Partial differential equations*; Quadrature; *Mathematical software;*

Additional Key Words and Phrases: Finite element methods, degrees-of-freedom, polyhedral cells

1 INTRODUCTION

The construction of conforming finite element spaces requires neighbouring cells to agree on the layout of shared degrees-of-freedom (DOFs), and—in the case of vector-valued elements—these

Fig. 1. (a) In the example on the left, the local orderings of the DOFs on the shared facet agree. On the right, the orderings do not agree. Here, a gap between the two cells has been added to help illustrate the orderings on each side. (b) The orientations of two DOFs on the faces of a higher-degree Nédélec first kind element on a tetrahedron from the point of view of two neighbouring cells. The directions on one face are rotated relative to the directions on the other face.

cells need to agree on the orientation of shared edges and faces. These requirements can be handled straightforwardly for low-order spaces with DOFs associated with vertices or a single DOF on each edge or face, but are non-trivial for higher-order finite element spaces. The finite element method permits arbitrary order approximations, but to exploit this feature it must be possible to construct appropriate degree-of-freedom maps and to agree to the necessary vector orientations that provide the required continuity.

To illustrate the orientation issue, Figure 1(a) shows shared DOFs on an edge as viewed from neighbouring cells. If the cells do not agree on the orientation of the edge, the DOFs may be mismatched and the finite element space will not be conforming. For discontinuous spaces, similar issues arise with the location of quadrature points on a cell edge or face. Vector-valued spaces bring additional complexity as the DOFs have an associated direction that needs to be agreed upon b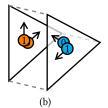y cells sharing the DOF. In some cases, it suffices to store a common sign. In other cases, agreement between cells may require linear combinations. To illustrate, two of the DOFs on a face of a higher-degree Nédélec first kind element [23] on two differently ordered neighbouring cells are shown in Figure 1(b). It can be seen that the directions from the point of view of one cell are a linear combination of the directions from the point of view of the other cell.

It is helpful to have an approach to the consistent ordering and orientation of DOFs on mesh entities for triangular, quadrilateral, tetrahedral, and hexahedral cells, mixed cell meshes, and for arbitrary polytopes. The latter case would support less common methods, such as the virtual element method [12] which considers arbitrary polygonal and polyhedral cells. Before presenting an approach for general polytope cells, we consider the approaches used in some widely used finite element libraries.

1.1 Current Approaches

Traditional finite element libraries tie the degree of the approximation space to the degree of the input mesh, with 'nodes' for point evaluation DOFs explicitly provided (and numbered) in the input mesh. This provides a unique global index for DOFs and makes a DOF map trivial to construct for specific element types. However, the approach allows only a limited set of finite element spaces to be constructed, usually low-degree Lagrange elements, and requires a new mesh to change the degree of the finite element space.

Modern finite element libraries usually decouple the input mesh from the family and degree of the finite element space that is defined on the mesh. The mesh defines the cell geometry, and different finite element spaces can then be defined on a given mesh. To construct DOF maps and to agree on the orientation of any vectors, it is common to require that the cells in a mesh are
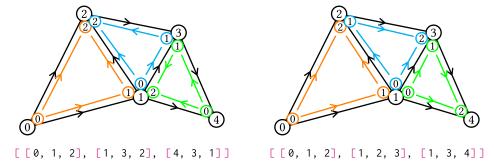
Fig. 2. An example mesh with three cells. The lists underneath each mesh show the global numbers of the vertices of each cell, and the coloured numbers show the local ordering of the vertices inside each cell. The mesh on the left has been defined so that the vertices of each cell are locally ordered in anticlockwise order, but the mesh is not ordered, so the local low-to-high edge orientation does not match the global low-to-high orientation. The mesh on the right is the result of ordering the vertices of each cell of the mesh on the left in ascending order: the local order of vertices inside each cell now agrees with the global numbering of the vertices, and so the local low-to-high edge orientation matches the global low-to-high orientation. The vertices, however, are no longer locally ordered in an anticlockwise order. If the mesh of triangles is embedded in a 3D space—for example if it is a mesh representing the surface of an object—this leads to an ambiguity when deciding on the normal vector to each face. In the mesh on the left, these normals all naturally point out of the page. For the ordered mesh on the right, the face normals for the leftmost cell will point out of the page, but for the other two cells, it points into the page. For some applications the cell orientation still needs to stored despite the cell being reordered.

locally numbered in a way that guarantees that entities are consistently ordered. For meshes with simplex cells, it is straightforward to reorder the local vertex indices of cells to ensure that the local orientations of the mesh entities on the reference element agree with their globally agreed upon orientations; the global orientation can be inferred from the global vertex indices. The reordering can be performed locally by cell without reference to neighbouring cells. An example triangular mesh is shown on the left in Figure 2. The mesh obtained by sorting the vertices in each cell in ascending order is shown on the right in Figure 2. In this reordered mesh, the local and global directions of each edge agree.

The re-ordering approach is implemented for simplex cells in a number of software libraries, including MFEM [6], FEniCS [3, 21, 22, 27], and Firedrake [25]. Once this re-ordering is done, it is guaranteed that the orientations of mesh entities are consistent between cells, allowing a DOF map to be created based on global indices assigned to each mesh entity, and the directions of any vectors will be consistent between cells. NGSolve [29]—a finite element solver based around the NetGen mesh generator [28]—avoids having to re-order the cells by generating meshes in a way that ensures that orientations are consistent. Higher-degree elements are only supported on meshes imported from an external source when they have a consistent ordering.

The re-ordering approach is a straightforward, local operation for simplex cells and all meshes of simplex cells can be ordered. For quadrilateral and hexahedral cells, the re-ordering approach is more complicated. Figure 3 shows the three distinct reference cells that would be needed for a naïvely low-to-high ordered mesh of quadrilateral cells. For a mesh of hexahedral cells, 501 distinct reference elements would be needed [33]. deal.II [7] implements a serial re-ordering strategy for quadrilateral and hexahedral cells to generate consistent edge orderings [1]. After reordering for quadrilateral cells, each cell is represented by the first reference shown in Figure 3.
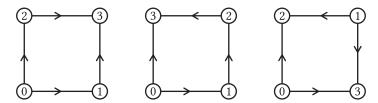
Fig. 3. These three quadrilateral reference elements are distinct. All other local orderings due to the low-to-high convention can be obtained by rotating and reflecting these three reference cells.

While re-ordering leads to consistent edge orientations for all quadrilateral cell meshes of an orientable manifold (for example, not a Möbius strip), the operation is not local to each cell. For the case of hexahedra, not all meshes can be re-ordered to give a consistent edge direction. If a consistent edge ordering for a mesh of hexahedra cannot be computed, deal.II will instead store orientation flags for each sub-entity of the cell so that these can be used during assembly [1]. A distributed memory parallel version of the algorithm in deal.II for quadrilaterals is presented in Homolya and Ham [17], and is used in the Firedrake library.

While the re-ordering approach allows high-degree finite element spaces to be constructed with ease for a range of problems, it is not without drawbacks. As already mentioned, not all meshes of hexahedral cells can be re-ordered. In other cases, explicit orientation markers are still required: for example, if using a two-dimensional mesh embedded in a three-dimensional space, or when solving a domain decomposition problem by coupling the finite element method with the boundary element method [13, 18], the normals to a two-dimensional mesh are required. These normals are reversed in reflected cells, and so whether or not the cells are reflected must be stored.

When using the *p*-version or *hp*-version of the finite element method [2, 31, 34] it is common to construct hierarchical basis functions 'on-the-fly' rather than *a priori* via the Ciarlet definition of a finite element. Basis functions in the *p*-version are typically associated with cell entities, with 'blending functions' used to ensure that functions associated with a cell entity have the required trace other cell entities. As shown in [16] for H^1 spaces, continuity for higher-degree spaces can be easily guaranteed with this approach by permuting the local coordinates on a cell entity such that all cells sharing the entity agree on the coordinate system, with basis functions then computed for each cell. This approach was extended to some $H(\mathrm{div})$ and $H(\mathrm{curl})$ elements in [15]. An advantage of the approach is simplicity. Drawbacks are that basis functions cannot be tabulated *a priori* on a reference cell and must be computed per cell at runtime, and it is not always clear how more unusual basis functions should be constructed. The former point limits the application of code generation techniques for offline basis function evaluation.

1.2 Proposed Approach

We present an approach for ensuring compatible spaces of arbitrary degree without creating multiple reference elements, mesh re-ordering or runtime evaluation of the basis, and which can be applied to meshes with mixed polytope cells. We consider Ciarlet-type elements and, instead of re-ordering each cell, we transform the DOFs on each cell sub-entity to ensure that the required continuity of finite element fields between cells is maintained. Given unique global indices for each mesh entity, the approach is entirely local to each cell. Care is required when considering integrals over facets and vector-valued basis functions, as in each of these cases the orientation of the facet is important and is no longer guaranteed to agree on each side of the facet.

In contrast to the approaches discussed in Section 1.1, the approach we propose can be applied to an arbitrary element on an arbitrary cell type without requiring any specialised implementation for a specific element or cell. While some parts of our approach—in particular, the permutation of the order of scalar point evaluation DOFs—are already implemented in other libraries, it does not appear that the full approach for handling $H(\text{div})$ and $H(\text{curl})$ elements is currently used elsewhere. By representing the transformations as the product of one (for edges) or two (for faces) base transformations per sub-entity, we vastly reduce the number of transformation matrices that need to be stored.

The approach developed in this article is implemented in the open-source FEniCSx libraries *DOLFINx* and *Basix*. The presented functionality can be accessed directly from the Basix element library via C++ and Python interfaces; its application in practice can be inspected in the finite element library DOLFINx. The libraries are available at https://github.com/FEniCS.

1.3 Outline

The remainder of this article is arranged as follows. In Section 2, we look at the definition of a finite element space and the different types of DOF that we consider. In Section 3, we describe the method for transforming the basis functions for each of these DOF types to achieve a consistent ordering. We then present some examples of how this method can be used in Section 4. In Section 5, we examine the method's implementation in FEniCSx. We finish with some concluding remarks in Section 6.

2 DEGREES-OF-FREEDOM

Before we look at transforming basis functions and constructing DOF maps, we define the finite element spaces and DOFs that we will consider. In general, finite elements can be defined by the following [14, 20].

Definition 2.1 (Ciarlet Finite Element). A finite element is defined by the triplet $(R, \mathcal{V}, \mathcal{L})$, where

- $R \subset \mathbb{R}^d$ is the reference element, usually a polygon or polyhedron;
- $\mathcal{V}$ is a finite dimensional polynomial space on R of dimension n;
- $\mathcal{L} := \{l_0, \dots, l_{n-1}\}$ is a basis of the dual space $\mathcal{V}^* := \{f : \mathcal{V} \to \mathbb{R}, |, f \text{ is linear}\}$.

The basis functions $\{\phi_0, \dots, \phi_{n-1}\}$ of the space $\mathcal{V}$ are defined by

$$l_i(\phi_j) = \begin{cases} 1 & i = j, \\ 0 & i \neq j. \end{cases} \tag{1}$$

Additionally, a map is defined that maps the basis functions on the reference element to functions on an arbitrary cell [8, 19, 27]. The mapping for an element is chosen so that it preserves important properties of the element's basis functions.

The functionals $l_i \in \mathcal{L}$ are the (local) DOFs of the finite element (note that we enumerate functionals and basis functions from 0). When a finite element is mapped to a mesh, a global DOF number is assigned to each local DOF on each cell. To ensure that the mapped space has the desired continuity properties, each functional $l_i \in \mathcal{L}$ is associated with a sub-entity of the reference element. On neighbouring cells, local DOFs that are associated with a shared sub-entity are assigned the same global DOF. For example, Figure 4 shows a basis function in a degree 1 Lagrange space that is associated with vertex 0 and a basis function of a degree 1 Lagrange space on a mesh: the local DOFs on each triangle that are associated with the central vertex have each been assigned the same global DOF, causing the function to be continuous.
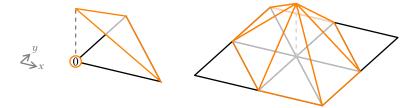
Fig. 4. Left: A basis function on the reference of a degree 1 Lagrange space on a triangle that is associated with vertex 0. Right: A basis function of a degree 1 Lagrange space on a triangular mesh. Each local DOF associated with the central vertex is associated with the same global DOF to ensure continuity.

Table 1. The Entities of Cells with Topological Dimensions 0 to 3

| Topological | Entities by dimension | | | | Entities by codimension | | | |
dimension	0	1	2	3	0	1 (facets)	2 (ridges)	3 (peaks)
0 (a vertex)	the cell	–	–	–	the cell	–	–	–
1 (an interval)	vertices	the cell	–	–	the cell	vertices	–	–
2 (a polygon)	vertices	edges	the cell	–	the cell	edges	vertices	–
3 (a polyhedron)	vertices	edges	faces	the cell	the cell	faces	edges	vertices

For clarity, we define the terms we will use to describe meshes and cells that we will use throughout. By *entities*, we refer to any of the vertices, edges, faces, or volumes of a cell: these have dimensions 0, 1, 2, and 3, respectively. The *topological dimension* is the dimension of the cell itself. The *geometric dimension* is the dimension of the space in which the mesh is embedded. For example, a mesh of a subset of $\mathbb{R}^2$ with triangular cells has topological dimension 2 and geometric dimension 2; a mesh of a 2-dimensional manifold in $\mathbb{R}^3$ with triangular cells has topological dimension 2 and geometric dimension 3. The *codimension* of an entity is given by subtracting the dimension of the entity from the topological dimension of the cell. Entities of codimensions 1, 2, and 3 are called *facets*, *ridges*, and *peaks*, respectively. The usual names given to entities of cells of topological dimensions 0 to 3 are shown in Table 1.

We now proceed to examine the different functional types that are commonly used to define DOFs. The example elements in the following sections are based on the definitions in the Periodic Table of the Finite Elements [11] and DefElement [32].

2.1 Point Evaluations

The first—and simplest—functional type is a point evaluation, given by evaluating the function at a fixed point on the reference element. In other words, these DOFs are defined, for $v \in \mathcal{V}$, by

$$l_i(v) := v(\boldsymbol{p}_i), \tag{2}$$

for some $\boldsymbol{p}_i \in R$. For example, a degree 1 (linear) Lagrange finite element on a triangle is defined by taking R to be the triangle with vertices at $(0, 0)$, $(1, 0)$, and $(0, 1)$, $\mathcal{V} := \text{span}\{1, x, y\}$, and placing a point evaluation DOF at each of the corners of the triangle. An example basis function of this space was shown in Figure 4; note that the basis function is equal to 1 at one vertex and 0 at the other vertices, and so Equation (1) is satisfied.

2.2 Point Evaluations of Face Tangents

For vector-valued function spaces, point evaluations must additionally include a direction in which the evaluation is taken. Vector point evaluations in directions tangential to a face of a three-dimensional cell require extra attention. There are two independent vectors tangent to a face, so
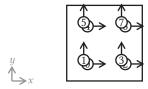
Fig. 5. A possible arrangement of vector point evaluation DOFs in directions tangential to the face.
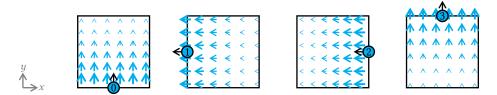

Fig. 6. The four basis functions of a Raviart–Thomas degree 1 space on a quadrilateral. The four DOFs are the integral moments over each edge with the normal to that edge. The normals here have been chosen to be 90° anticlockwise rotations of the low-to-high edge directions shown in the leftmost reference element in Figure 3.

it is common to collocate two DOFs at each point. Figure 5 shows how eight face tangent DOFs could be arranged on a quadrilateral face. If two cells meet at a face that has face tangent DOFs, care must be taken to ensure that the directions assigned to the DOFs by each cell agree.

2.3 Integral Moments

Numerous finite elements are defined using integral moment DOFs. For scalar-valued function spaces, these are defined, for $v \in \mathcal{V}$, by

$$l_i(v) := \int_\Xi v\phi, \tag{3}$$

where $\Xi \subseteq R$ is some subset of the reference element, usually the whole cell or one of its faces or edges, and $\phi : \Xi \to \mathbb{R}$ is a weight function. For vector-valued function spaces, integral moment DOFs are defined, for $\boldsymbol{v} \in \mathcal{V}$, by

$$l_i(v) := \int_\Xi \boldsymbol{v} \cdot \boldsymbol{\phi}. \tag{4}$$

In this case, $\boldsymbol{\phi} : \Xi \to \mathbb{R}^d$ is a vector-valued weight function.

For example, a degree 1 Raviart–Thomas element [26] on a quadrilateral is defined by taking R to be the quadrilateral with vertices at $(0, 0)$, $(1, 0)$, $(0, 1)$, and $(1, 1)$,

$$\mathcal{V} := \operatorname{span}\left\{\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}x\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}0\\y\end{pmatrix}\right\},$$

and defining four DOFs by

$$l_i(\boldsymbol{v}) := \int_{e_i} \boldsymbol{v} \cdot \hat{\boldsymbol{n}}_i \, \mathrm{d}s, \tag{5}$$

where e_i is the i-th edge of the quadrilateral, and $\hat{\boldsymbol{n}}_i$ is the unit vector normal to e_i. The basis functions of this space are shown in Figure 6.

As a second example, a degree 2 Nédélec first kind edge element [23] on a tetrahedron is defined by taking R to be the tetrahedron with vertices at $(0, 0, 0)$, $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$, and

$$
\mathcal{V} := \mathrm{span} \left\{ \begin{array}{l} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} x \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} y \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ x \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ y \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ x \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ y \end{pmatrix}, \\[18pt] \begin{pmatrix} z^2 \\ 0 \\ -xz \end{pmatrix}, \begin{pmatrix} xz \\ 0 \\ -x^2 \end{pmatrix}, \begin{pmatrix} yz \\ 0 \\ -xy \end{pmatrix}, \begin{pmatrix} 0 \\ z^2 \\ -yz \end{pmatrix}, \begin{pmatrix} 0 \\ yz \\ -y^2 \end{pmatrix}, \begin{pmatrix} 0 \\ xz \\ -xy \end{pmatrix}, \begin{pmatrix} xy \\ -x^2 \\ 0 \end{pmatrix}, \begin{pmatrix} y^2 \\ -xy \\ 0 \end{pmatrix} \end{array} \right\}.
$$

Twenty DOFs are used to define this space. The first twelve DOFs are defined by placing two integral moments on each edge; these are given by

$$
l_{2i+j}(\boldsymbol{v}) := \int_{e_i} \boldsymbol{v} \cdot (\boldsymbol{t}_i p_{i,j}) \, \mathrm{d}s,
$$

where e_i is the i-th edge of the tetrahedron, $p_{i,j}$ is the j-th basis function of a degree 1 Lagrange finite element space on e_i, and $\hat{\boldsymbol{t}}_i$ is the unit vector tangential to e_i. We define the remaining eight DOFs as integral moments on the faces of the tetrahedron; these are given by

$$
l_{6+2j+i}(\boldsymbol{v}) := \int_{f_j} \boldsymbol{v} \cdot \boldsymbol{\psi}_{j,i} \, \mathrm{d}s,
$$

where f_j is the i-th face of the tetrahedron, and $\boldsymbol{\psi}_{j,0}$ and $\boldsymbol{\psi}_{j,1}$ are vectors pointing in the directions of two of the edges of face f_j.

3 PERMUTING AND TRANSFORMING CELL BASIS FUNCTIONS

Now that we have examined functionals that can be used to define DOFs, we look at how the reference basis functions can be adjusted on each cell to ensure that the required continuity between cells is maintained. The key idea in this section is to consider basis functions by the mesh entity with which the corresponding DOF is associated and then treat these sets independently.

Let $\{\phi_0, \ldots, \phi_{n-1}\}$ be a set of basis functions, as ordered on the reference cell. Our general approach is to define a matrix $M \in \mathbb{R}^{n \times n}$ for each cell such that the transformed basis functions for that cell $\{\tilde{\phi}_0, \ldots, \tilde{\phi}_{n-1}\}$ are given by

$$
\begin{pmatrix} \tilde{\phi}_0 \\ \vdots \\ \tilde{\phi}_{n-1} \end{pmatrix} = M \begin{pmatrix} \phi_0 \\ \vdots \\ \phi_{n-1} \end{pmatrix}. \tag{6}
$$

The transformation M is computed by comparing the orientation of the entities of each cell to the orientation of the reference element. The chosen orientation of the reference element, and the numbering of each entity, is arbitrary. In all examples presented, we follow the ordering convention used by Basix [30]: for simplices, this follows the UFC ordering convention that orders sub-entities by the numbers of the vertices that are non-incident to the sub-entity [4]. For other cells, the sub-entities are ordered using the numbers of the vertices that are incident to the sub-entity.

The method proposed here associates a set of transformations to each finite element that represent how the element's DOFs are arranged and how they are affected by reflecting or rotating a sub-entity of the reference element. Although these transformations are the same for some elements, they vary greatly for other elements. For example, Nédélec second kind elements [24] have integral moments with the basis functions of a Raviart–Thomas space on their facets, meaning that the DOFs on these facets do not have the same arrangement as the DOFs of a Lagrange element (as shown in the top row of Figure 7). For more exotic elements, the DOF arrangements can be

even more unusual: for example, serendipity spaces [9] and the divergence- and curl-conforming spaces based on them [10] are defined for hexahedral cells, but the DOFs on their faces are integral moments against spaces defined on triangles. Therefore, the transformations must be associated with the element and not the cell type. The operator M is then constructed as a composition of operations for each element cell entity.

We first consider how spaces of scalar-valued functions are transformed before moving on to spaces of vector-valued functions. We consider two types of vector-valued function space: divergence-conforming spaces and curl-conforming spaces. These are distinguished by the type of continuity required between cells: a space is divergence-conforming if the components normal to the facets of the cell are continuous; a space is curl-conforming if the components tangent to the facets of the cell are continuous.

3.1 Scalar Spaces

For spaces of scalar functions, we must ensure that the DOFs on a mesh entity appear in the same order when viewed from cells adjacent to the entity, as shown in Figure 1(a). Vertices are trivially handled as they have no orientation. For two-dimensional cells, the shared entities of interest are edges; for three-dimensional cells, the shared entities of interest are edges and faces. When permuting a cell, the direction of each of its edges may be reversed; hence, the order of the DOFs on each edge may need to be reversed. Faces of a cell can be rotated and reflected when permuting a cell, so their DOFs may need to be rotated or reflected.

DOFs are associated with a single mesh entity only, so permuting one entity will have no effect on the DOFs of other entities. Therefore, each entity can be considered independently of other entities. We therefore split the transformation of the DOFs on a cell into *base transformations*. For each edge of a two- or three-dimensional reference cell, we store one base transformation that describes the effect of reversing that edge. For each face of a three-dimensional reference cell, we store two base transformations: one that describes the effect of rotating the face and one that describes the effect of reflecting the face. The rotations and reflection of a face of a tetrahedron and hexahedron that are represented by base transformations are shown in Figure 7. For each cell in the mesh, it can then be calculated how many times each base transformation needs to be applied to the cell by comparing the order of the global vertex numbers on each entity to the ordering of the reference element. We use a low-to-high orientation of the entities to do this, but other conventions could be used. The transformation M for the cell can then be calculated by multiplying each base transformation the necessary number of times. Matrices representing rotations and reflections of the same face of a cell do not commute, so it must be decided beforehand in which order these transformations will appear in the product of base transformations.

For many scalar-valued spaces, the base transformations will be permutation matrices as permuting a mesh entity leads to swapping the positions of the DOFs. This is not true in all cases, however: for higher-degree serendipity spaces on a hexahedron, for example, the base transformations for DOFs on the faces will be linear transformations and not permutations. The implementation of higher-degree serendipity spaces will be discussed in Section 5.1.

3.2 Divergence-Conforming Spaces

For vector-valued functions, the direction of the vector used to define the DOF must be taken into account alongside the ordering of DOFs on an entity. If two adjacent cells do not agree on the direction of a DOF, the basis functions on one of the cells must be adjusted to correct for this.

For divergence-conforming spaces, neighbouring cells must agree on the direction of the normal vector on facets. If two cells disagree on the direction of a DOF, then we can multiply the basis functions associated with the facet on one of the cells by -1 to correct the disagreement. In order
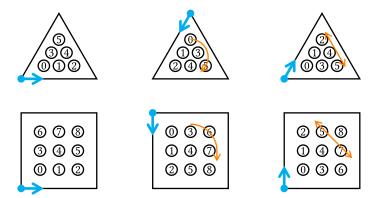
Fig. 7. The effect of rotating and reflecting a face of a tetrahedron (top) and hexahedron (bottom) on a possible layout of DOFs on that face. The blue arrows have been included to indicate the orientation of the triangle or quadrilateral following each transformation.

Fig. 8. The effect of rotating and reflecting a face of a tetrahedron on the direction of face tangents. A rotation leads to $\tilde{l}_0 = -l_1$ and $\tilde{l}_1 = l_0 - l_1$. A reflection leads to $\tilde{l}_0 = l_1$ and $\tilde{l}_1 = l_0$. The blue arrows have been included to indicate the orientation of the triangle following each transformation.

to decide when to multiply values by -1, we need to know whether each entity of a cell has been reflected. This information has already been calculated when determining how many times to apply each base transformation to the cell, so can be reused here.

3.3 Curl-Conforming Spaces

For curl-conforming spaces, the tangential component of the vector will be continuous between cells. For two-dimensional cells, directions can be corrected in the same way as for divergence-confirming spaces. For three-dimensional cells, however, the situation is more complicated. As we saw in Section 2.2, there will commonly be two face tangent DOFs collocated at points on the face of a three-dimensional cell, as the space of vectors tangent to the face has dimension 2. For a given face of a tetrahedral or hexahedral cell, we take the directions for these DOFs to be parallel to two of the edges of the face (other directions could be used and would lead to slight variants of the base transformations we arrive at). The effect of rotating and reflecting the face on the direction of the DOFs on triangular and quadrilateral faces are shown in Figures 8 and 9, respectively.

For face tangent DOFs on faces shared by two cells, linear transformations must be applied to the basis functions to counteract the effect of rotating and reflecting a face. As for div-conforming spaces, the data calculated when working out how many base transformations to apply to each cell can be reused to determine which transformations need to be applied.

3.4 Composing the Base Transformations for Each Entity

The transformation matrix M for a given cell can be calculated by comparing the orientation of each sub-entity to the reference orientation, then composing (multiplying) the required base

Fig. 9. The effect of rotating and reflecting a face of a hexahedron on the direction of face tangents. A rotation leads to $\tilde{l}_0 = -l_1$ and $\tilde{l}_1 = l_0$. A reflection leads to $\tilde{l}_0 = l_1$ and $\tilde{l}_1 = l_0$. The blue arrows have been included to indicate the orientation of the quadrilateral following each transformation.

transformations. The transformation matrix can then be applied at cell level during the finite element assembly process (while the transformation can be expressed as a matrix multiplication, in practice it is typically applied without forming M).

In cases in which the base transformations are permutation matrices, the permutations can be applied in the construction of the DOF map, avoiding the application of transformations during assembly. For such scalar-valued spaces, the permutations we apply are the same as those proposed in [16], although we apply these permutations to the DOF numbering of each cell rather than applying them during the creation of the reference basis functions.

For sub-entities that are edges, we compare the orientation of the edge to the reference orientation inferred from the global vertex numbers of the edge's endpoints. If these orientations disagree, then we apply the base transformation for that edge. Note that, in some cases, two cells adjacent to an edge may agree on the orientation of the edge, but both disagree with the reference orientation. In this case, both cells will apply their base permutation, as each cell will compare the edge's orientation with the reference only and not with the other cell.

For sub-entities that are faces, we use two base transformations: one of these represents moving the vertices one position clockwise (i.e., a 120° rotation for a equilateral triangle or a 90° rotation for a square); the other represents a reflection of the face. The cell must decide how many times this rotation needs to be applied to the face and whether to apply the rotation. This can be done by first identifying which of the face's vertices has the lowest global number. The number of rotations to apply is calculated by looking at how much the face needs to be rotated to move this vertex to the position of the locally lowest-numbered vertex. We then look at the two vertices next to the one that we have just identified. If the lower-numbered vertex is not in the same direction as the corresponding direction for the local vertex numbering, we apply a reflection to this face. Once the lowest globally numbered vertex and its lowest-numbered neighbour have been identified, we are able to fully orient the face in a consistent way to neighbouring cells. The global numbers of the other vertices are irrelevant and any permutation of the order of these can be ignored.

3.5 Discontinuous Spaces and Facet Integrals

When using, for example, discontinuous Galerkin schemes, integrals over shared facets are computed. It is common for these integrals to involve jump and average operations. This requires the evaluation of basis functions from both neighbouring cells at a series of quadrature points on the facet. In order for integrals to be calculated correctly, the two cells sharing a facet must agree on the orientation of the facet so that both cells agree on the location of the quadrature points. In an unordered mesh, these orientations may not agree. The quadrature points must be rotated and reflected to correct for this. Again, the data calculated when working out how many times to apply the base transformations can be reused here. The transformation of quadrature points cannot be amalgamated into the base transformation matrices, so must be considered separately.
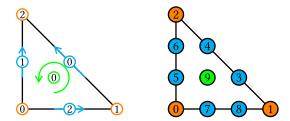
Fig. 10. Left: The numbering and orientation of the entities of the reference triangle. Right: The DOF arrangement for a degree 3 Lagrange space. Each DOF is a point evaluation.

Table 2. The Permutations that the Base Transformations Represent
for a Degree 3 Lagrange Space on a Triangular Cell

0	1	2	**4**	**3**	5	6	7	8	9	edge ⓪ reversed	(B_0)
0	1	2	3	4	**6**	**5**	7	8	9	edge ① reversed	(B_1)
0	1	2	3	4	5	6	**8**	**7**	9	edge ② reversed	(B_2)

The DOFs that are on the entity permuted in each row are highlighted.

The transformation of quadrature points for facet (and ridge) integrals is essentially the same as the transformation of DOFs in a space containing point evaluations, with the quadrature points taking the place of the DOF points.

4 EXAMPLES

In this section, we discuss how our method can be applied to four different elements: a degree 3 Lagrange element on a triangle, a degree 2 Raviart–Thomas element on a quadrilateral, a degree 4 Lagrange space on a hexahedron, and a degree 2 Nédélec first kind element on a tetrahedron.

4.1 Computing the Base Transformations for a Cell

4.1.1 Example 1: Lagrange on a Triangle. The reference element and DOF layout for a degree 3 continuous Lagrange space are shown in Figure 10. The base transformations for this element will describe how the order of the DOFs is affected by reversing an edge. From Figure 10, we can see that reversing edge 0 will lead to the positions of DOFs 3 and 4 being swapped; reversing edge 1 will lead to the positions of DOFs 5 and 6 being swapped; and reversing edge 2 will lead to the positions of DOFs 7 and 8 being swapped. These base transformations can be represented by the following matrices:

$$B_0 := \begin{pmatrix} I_3 & & & \\ & 0 & 1 & \\ & 1 & 0 & \\ & & & I_5 \end{pmatrix}, \qquad B_1 := \begin{pmatrix} I_5 & & & \\ & 0 & 1 & \\ & 1 & 0 & \\ & & & I_3 \end{pmatrix}, \qquad B_2 := \begin{pmatrix} I_7 & & & \\ & 0 & 1 & \\ & 1 & 0 & \\ & & & 1 \end{pmatrix}, \qquad (7)$$

where I_n is an n by n identity matrix and the omitted blocks of each matrix are 0. For this element, the base transformation matrices are permutations. Therefore, the permutations shown in Table 2 could be applied directly when creating the DOF map.

4.1.2 Example 2: Raviart–Thomas on a Quadrilateral. The reference element and DOF layout for a degree 2 Raviart–Thomas element are shown in Figure 11. In the same way as in the first example, the base transformations for this element can be calculated by considering the effect that
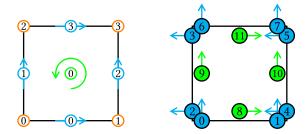
Fig. 11. Left: The numbering and orientation of the entities of the reference quadrilateral. Right: The DOF arrangement for a degree 2 Raviart–Thomas space. DOFs 0 to 7 are integral moments with degree 1 Lagrange basis functions on the edges. DOFs 8 to 11 are integral moments against degree 1 Nédélec basis functions on the face.

reversing an edge would have on the DOFs on that edge. For this element, however, we must also consider the effect of reversing each edge on the directions of the DOFs on that edge: when an edge is reversed, these directions need to be reversed. For example, DOFs 0 and 1 are on edge 0, so they must be reversed if this edge is reversed. Note that DOF 8 is associated with the interior of the cell, so it is unaffected by the reversal of edge 0. The following base transformation matrices describe the effect on this element of reversing each edge:

$$
B_0 := \begin{pmatrix} 0 & -1 & \\ -1 & 0 & \\ & & I_{10} \end{pmatrix}, \qquad
B_1 := \begin{pmatrix} I_2 & & & \\ & 0 & -1 & \\ & -1 & 0 & \\ & & & I_8 \end{pmatrix},
$$

$$
B_2 := \begin{pmatrix} I_4 & & & \\ & 0 & -1 & \\ & -1 & 0 & \\ & & & I_6 \end{pmatrix}, \qquad
B_3 := \begin{pmatrix} I_6 & & & \\ & 0 & -1 & \\ & -1 & 0 & \\ & & & I_4 \end{pmatrix}. \tag{8}
$$

In each matrix, the DOFs on an edge are swapped (as their positions are swapped) and multiplied by -1 (as their directions are reversed).

4.1.3 Example 3: Lagrange on a Hexahedron. The reference element and DOF layout for a degree 4 Lagrange element are shown in Figure 12. As in the previous two examples, the order of the DOFs on the edges of this element will be reversed if an edge is reversed. As the cell is three-dimensional in this example, we must also include base transformations describing the effect of rotating and reflecting a face of the cell. Face 1 includes the DOFs 53 to 61. Referring to Figure 7, we can see the effect of rotating this face and construct the base transformations. As in the first example, each base transformation for this element is a permutation. These permutations are shown in Table 3. We note that the rotation shown in Figure 7 goes in the opposite direction to that shown in Figure 12. This is because we count how many times a face has been rotated to get from a cell to the reference cell, then use the base transformations to "undo" this rotation and map from the reference cell to the physical cell.

4.1.4 Example 4: Nédélec First Kind on a Tetrahedron. The reference element and DOF layout for a degree 2 Nédélec first kind element are shown in Figure 13. Once again, the effect of reversing each entity can be ascertained by looking at the layout of DOFs on the reference element. As in the second example, the directions of DOFs on an edge are reversed when that edge is reversed.
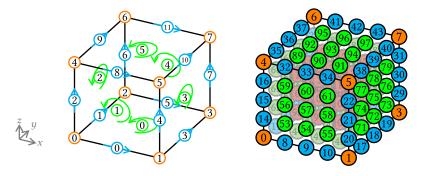
Fig. 12. Left: The numbering and orientation of the entities of the reference hexahedron. Right: The DOF arrangement for a degree 4 Lagrange space. Each DOF is a point evaluation.

Table 3. The Permutations Described by the Base Transformations for a Degree 4
Lagrange Space on a Hexahedral Cell

...	10	9	8	...						edge ⓪ reversed	(B_0)
...	13	12	11	...						edge ① reversed	(B_1)
...	16	15	14	...						edge ② reversed	(B_2)
...	19	18	17	...						edge ③ reversed	(B_3)
...	22	21	20	...						edge ④ reversed	(B_4)
...	25	24	23	...						edge ⑤ reversed	(B_5)
...	28	27	26	...						edge ⑥ reversed	(B_6)
...	31	30	29	...						edge ⑦ reversed	(B_7)
...	34	33	32	...						edge ⑧ reversed	(B_8)
...	37	36	35	...						edge ⑨ reversed	(B_9)
...	40	39	38	...						edge ⑩ reversed	(B_{10})
...	43	42	41	...						edge ⑪ reversed	(B_{11})
...	46	49	52	45	48	51	44	47	50	... face ⓪ rotated	(B_{12})
...	44	47	50	45	48	51	46	49	52	... face ⓪ reflected	(B_{13})
...	55	58	61	54	57	60	53	56	59	... face ① rotated	(B_{14})
...	53	56	59	54	57	60	55	58	61	... face ① reflected	(B_{15})
...	64	67	70	63	66	69	62	65	68	... face ② rotated	(B_{16})
...	62	65	68	63	66	69	64	67	70	... face ② reflected	(B_{17})
...	73	76	79	72	75	78	71	74	77	... face ③ rotated	(B_{18})
...	71	74	77	72	75	78	73	76	79	... face ③ reflected	(B_{19})
...	82	85	88	81	84	87	80	83	86	... face ④ rotated	(B_{20})
...	80	83	86	81	84	87	82	85	88	... face ④ reflected	(B_{21})
...	91	94	97	90	93	96	89	92	95	... face ⑤ rotated	(B_{22})
...	89	92	95	90	93	96	91	94	97	... face ⑤ reflected	(B_{23})

As the space has a high number of DOFs, only the DOFs on the entity are shown. The other
DOFs will be unaffected by the base transformation.

Therefore, the base transformations for each edge (B_0 to B_5) will be identity matrices with a 1 replaced with -1 for the DOFs on that edge. The DOFs on the faces of this element are not moved by permuting the face—as the two DOFs on each face are collocated at the same point—but their directions are affected. Referring back to Figure 8, we can calculate that the effect of rotating face
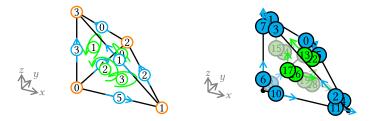
Fig. 13. Left: The numbering and orientation of the entities of the reference tetrahedron. Right: The DOF arrangement for a degree 2 Nédélec first kind curl-conforming space. DOFs 0 to 11 are integral moments with degree 1 Lagrange basis functions on the edges. DOFs 12 to 19 are integral moments of the tangential components on the faces.

0 is described by the transformation

$$B_6 := \begin{pmatrix} I_{12} & & & \\ & 0 & -1 & \\ & 1 & -1 & \\ & & & I_6 \end{pmatrix},$$

and the effect of reflecting face 0 is described by the transformation

$$B_7 := \begin{pmatrix} I_{12} & & & \\ & 0 & 1 & \\ & 1 & 0 & \\ & & & I_6 \end{pmatrix}.$$

These two matrices are the base transformations for face 0. The effect of any transformation of face 0 can be calculated by taking various products of these matrices. We can describe the effect of rotating and reflecting the other faces in the same way to obtain B_8 to B_{13}.

4.2 Application of the Base Transformations

We now look at how the base transformations computed in the previous section can be applied. Figure 14 shows an example mesh with four triangular cells on which we will compute the transformed basis functions of a degree 3 Lagrange space. The reference orientations of the cell sub-entities are shown in Figure 14(b): the edges marked with a star are those whose local orientation on that cell do not match the low-to-high orientations shown in Figure 14(a). These stars tell us which base transformations we should apply to the reference basis functions for that cell.

In cell 0, all three edges are reversed, so we should apply all three base transformations to the reference basis functions. Therefore, for cell 0, the transformation matrix is given by

$$M_0 := B_0 B_1 B_2 = \begin{pmatrix} I_3 & & & & & & & \\ & 0 & 1 & & & & & \\ & 1 & 0 & & & & & \\ & & & 0 & 1 & & & \\ & & & 1 & 0 & & & \\ & & & & & 0 & 1 & \\ & & & & & 1 & 0 & \\ & & & & & & & 1 \end{pmatrix}, \tag{9}$$
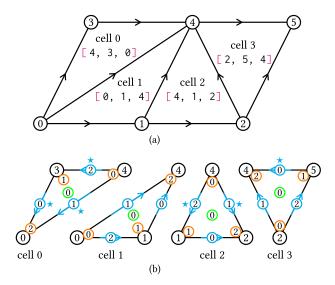
Fig. 14. (a) The mesh of triangles used in the first example, with low-to-high edge orientations. (b) The local numbering and orientations of the entities of the four cells in the first and fourth examples. The edges labelled with stars are those whose local orientation does not match their global orientation shown in Figure 14(a).

where B_0 to B_2 are the base transformations defined in (7). In cell 1, no edges are reversed, so M_1 is the identity matrix and no change is made to the basis functions. In cell 2, edges 1 and 2 are reversed, so M_2 is given by

$$
M_2 := B_1 B_2 = \begin{pmatrix} I_5 & & & & & \\ & 0 & 1 & & & \\ & 1 & 0 & & & \\ & & & 0 & 1 & \\ & & & 1 & 0 & \\ & & & & & 1 \end{pmatrix}. \tag{10}
$$

In cell 4, only edge 0 is reversed, so M_3 is equal to B_0 (the base transformation for edge 0).

As noted earlier, all of the base transformations for this element are permutations. Therefore, they can be applied directly when creating the DOF map. Figure 15(a) shows the global DOF numbers assigned to each cell following the application of these permutations. If the permutations are not applied, the DOF numbers shown in Figure 15(b) would be assigned instead: note that, in this case, the positions of the DOFs on the edges do not agree from the point of view of each cell, and so the global basis functions would not be continuous.

For the degree 2 Raviart–Thomas space on a quadrilateral, the transformation matrix for each cell can be computed in the same way by combining the base transformation defined in Equation (8) except, in this case, the transformations are not permutations, so they cannot be applied directly to the DOF map. For example, for the mesh and reference orientations shown in Figure 16, M_0 would be the identity for cell 0. For cell 1, $M_1 := B_0 B_2$ (the product of the base permutations for edges 0 and 2).

For three-dimensional cells, the situation is slightly more complex, as we have to apply base transformations for the faces as well as the edges. For each face, we need to ascertain how many times the face needs to be rotated and if it needs to be reflected to map from the reference orientation. A two-cell mesh for a degree 4 Lagrange space on a hexahedron is shown in Figure 17.
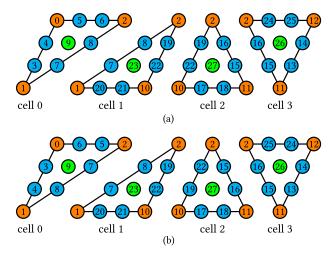
Fig. 15. (a) The DOF numbering on each cell in the mesh once the permutation has been applied to each cell. (b) The DOF numbering on each cell if no permutation is performed.
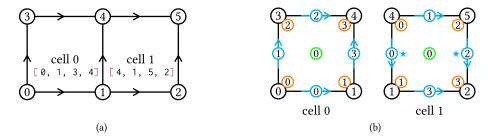

Fig. 16. (a) The mesh of quadrilaterals used in the second example, with low-to-high edge orientations. (b) The local numbering and orientations of the entities of the three cells in the second example. The edges labelled with stars are those whose local orientation does not match their global orientation shown in Figure 16(a).

Table 4. Do the Edges (Top) and Faces (Bottom) of Each Cell of the Mesh in Figure 17 Need Reflecting, and How Many Times Do the Faces Need Rotating?

	⓪	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩	⑪
cell 0	false	false	false	false	false	false	false	true	false	false	true	true
cell 1	false	true	true	true	false	false	false	false	false	false	true	false

	⓪	①	②	③	④	⑤
cell 0	true, 0	false, 0	false, 0	false, 0	false, 0	true, 2
cell 1	true, 2	true, 2	false, 1	true, 1	true, 0	true, 0

Table 4 summarises which base transformations should be applied to reference basis functions for each cell of this mesh. While the transformations due to reflections can be applied only once (as reflecting twice is the same as doing nothing), the transformations due to rotations can be applied more than once. For example, the transformation due to rotating face 0 must be applied twice to the reference basis functions for cell 0.
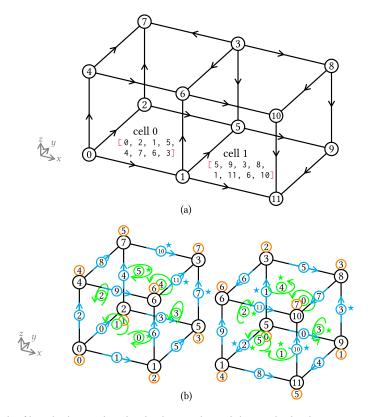
Fig. 17. The mesh of hexahedra used in the third example, with low-to-high edge orientations. (b) The local numbering and orientations of the entities of the three cells in the third example. The edges labelled with stars are those whose local orientation does not match their global orientation shown in Figure 17(a).

4.3 Extension to Arbitrary Cells and Mixed Meshes

The proposed method considers sub-entities of each cell, and the same transformation is applied to a sub-entity of a given type whatever the type of the cell. It is straightforward, therefore, to extend this method to arbitrary polygons and polyhedra and meshes with a mixture of cell types. Figure 18 shows three example cells in a mixed mesh. As in the previous examples, we could apply the base transformations for the edges labelled with a star.

On a mesh of hexagonal prisms, for example, each cell would have 18 edges and 8 faces: 1 base transformation would be needed for each edge and 2 base transformations would be needed for each face. As in the previous examples, these base transformations can then be combined to find the transformation for any cell.

5 IMPLEMENTATION

The DOF transformation method described in this article is implemented in the FEniCSx open-source finite element libraries [3, 21]. In Basix—the FEniCSx library for element definitions, basis function computation and interpolation—we compute the base transformations for each element and provide these alongside the other information about the finite element defined on the reference cell. FFCx—the FEniCSx form compiler—generates kernels (in C) from the Unified Form Language (UFL) [5] to evaluate finite element forms on a cell. Evaluation is on a common reference cell and the kernels are executed without any reference to cell orientations. DOLFINx is the main
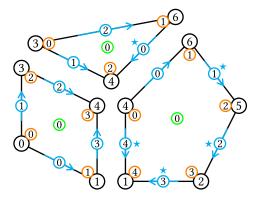
Fig. 18. Three cells in a mixed mesh. The edges and faces labelled with stars are those whose local orientation does not match their global orientation.

user interface of FEniCSx and executes cell kernels on a mesh. The transformation of cell basis functions, discussed in Sections 3.1 to 3.4, is managed by DOLFINx using Basix. After calling the kernel for a cell, DOLFINx calls Basix to apply, in place, the required transformations to the kernel output. DOLFINx passes the required cell orientation data to Basix. The data can be packed into a single integer, with one bit per entity used to signify whether the entity is reflected and 2 bits per face (for triangles and quadrilaterals; polygons with more edges require more bits) to signify how many times the face has been rotated. The information for each entity can then be retrieved using bit operations. For elements whose base transformations are simply permutations of the DOFs (e.g., Lagrange elements), the effects of the permutations are incorporated into the DOF map construction and no transformations are required during the assembly process.

The transformation matrices M for each cell are never explicitly computed. Instead, the necessary composition of base transformations is directly applied to the evaluations of the reference basis functions. As each base transformation affects only a small number of basis functions, we can restrict our transformation to affect the evaluations of the relevant functions only rather than repeatedly multiplying by n by n matrices.

5.1 Computing the Base Transformations in Basix

For Lagrange spaces, it is reasonably straightforward to implement functions that generate the necessary permutations of the DOF numbering for an arbitrary degree element. For spaces defined using integral moments, the effect of permuting a cell sub-entity is dependent on the space against which the integral moments are taken. As an example, we consider a degree 5 serendipity element on a hexahedron [9]. The functionals $l_0, \ldots, l_{73}$ that define this space are defined by:

- l_0 to l_7 are point evaluations at the vertices of the hexahedron.
- l_8 to l_{55} are integral moments against a degree 3 Lagrange space on the edges of the hexahedron. There are 4 of these functionals for each edge.
- l_{56} to l_{73} are integral moments against a degree 1 dPc space on the faces of the hexahedron. A dPc space on a quadrilateral contains the basis functions of a Lagrange space on a triangle [11]. The basis functions of a degree 1 dPc space are

$$\psi_0(t, s) = 1 - t - s,$$
$$\psi_1(t, s) = t,$$
$$\psi_2(t, s) = s.$$

There are 3 of these functionals for each face.

To illustrate how the base transformations can be computed, we consider the DOFs of this element on the 0th face of the cell (l_{56} to l_{58}). To compute the base transformation corresponding to a rotation of this face, we consider the transformation $T : (t, s) \mapsto (s, 1 - t)$. This represents a clockwise rotation of the face and can be used to undo anticlockwise rotations. If we apply the transformation T to the basis functions of the dPc space on the face, we obtain that

$$\psi_0(t, s) \mapsto \tilde{\psi}_0(t, s) = t - s,$$
$$\psi_1(t, s) \mapsto \tilde{\psi}_1(t, s) = s,$$
$$\psi_2(t, s) \mapsto \tilde{\psi}_2(t, s) = 1 - t.$$

These can be represented in terms of ψ_0 to ψ_2 as

$$\begin{pmatrix} \tilde{\psi}_0 \\ \tilde{\psi}_1 \\ \tilde{\psi}_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 & -1 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \end{pmatrix}. \tag{11}$$

The matrix here contains the coefficients of an interpolation of the functions $\tilde{\psi}_0$ to $\tilde{\psi}_2$ into the dPc space on the quadrilateral face. The same matrix can be use to write the functionals $\tilde{l}_{56}$ to $\tilde{l}_{58}$ (representing integral moments with the functions $\tilde{\psi}_0$ to $\tilde{\psi}_2$) in terms of l_{56} to l_{58}:

$$\begin{pmatrix} \tilde{l}_{56} \\ \tilde{l}_{57} \\ \tilde{l}_{58} \end{pmatrix} = \begin{pmatrix} 0 & 1 & -1 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} l_{56} \\ l_{57} \\ l_{58} \end{pmatrix}.$$

The basis functions ϕ_{56} to ϕ_{58} of the serendipity space, and the basis functions $\tilde{\phi}_{56}$ to $\tilde{\phi}_{58}$ corresponding to the functionals $\tilde{l}_{56}$ to $\tilde{l}_{58}$ satisfy

$$\begin{pmatrix} \tilde{l}_{56} \\ \tilde{l}_{57} \\ \tilde{l}_{58} \end{pmatrix} \begin{pmatrix} \tilde{\phi}_{56} \\ \tilde{\phi}_{57} \\ \tilde{\phi}_{58} \end{pmatrix}^t = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad \begin{pmatrix} l_{56} \\ l_{57} \\ l_{58} \end{pmatrix} \begin{pmatrix} \phi_{56} \\ \phi_{57} \\ \phi_{58} \end{pmatrix}^t = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The base transformation that we want to calculate is the matrix A such that

$$\begin{pmatrix} \phi_{56} \\ \phi_{57} \\ \phi_{58} \end{pmatrix} = A \begin{pmatrix} \tilde{\phi}_{56} \\ \tilde{\phi}_{57} \\ \tilde{\phi}_{58} \end{pmatrix}.$$

Combining these, we see that

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \tilde{l}_{56} \\ \tilde{l}_{57} \\ \tilde{l}_{58} \end{pmatrix} \begin{pmatrix} \tilde{\phi}_{56} \\ \tilde{\phi}_{57} \\ \tilde{\phi}_{58} \end{pmatrix}^t$$

$$= \begin{pmatrix} 0 & 1 & -1 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} l_{56} \\ l_{57} \\ l_{58} \end{pmatrix} \begin{pmatrix} \phi_{56} \\ \phi_{57} \\ \phi_{58} \end{pmatrix}^t A^{-t}$$

$$= \begin{pmatrix} 0 & 1 & -1 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix} A^{-t}.$$

Therefore, the base transformation representing a rotation of the 0th face is given by

$$A = \begin{pmatrix} 0 & 1 & -1 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix}^t = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ -1 & 1 & 1 \end{pmatrix}.$$

In this way, we can compute the base transformations for DOFs defined by integral moments by computing the interpolation matrix into the space we are taking integral moments against, as shown in Equation (11). In Basix, the information needed to calculate these interpolation coefficients is created when an element is initialised; this information can be reused to compute the base transformations.

The base transformations for the reflection of a face and for integral moments on other mesh sub-entities can be obtained by following a similar method.

6 CONCLUSIONS

We have presented a method for constructing Ciarlet finite element spaces of arbitrary degree for any cell type and for a range of degrees-of-freedom types and, hence, finite element types. The method does not depend on re-ordering the mesh, which removes the major limitation of other approaches. The method can be applied to meshes of triangles, tetrahedra, quadrilaterals, and hexahedra. It also extends to general polygonal and polyhedral cells, and mixed meshes.

For some elements, this approach involves applying a permutation to the DOF numbering on each cell. For other elements, the issues cannot be resolved using permutations, and linear transformations must be applied to the local element tensor for each cell.

The construction of the entity permutation data needed by this method needs only knowledge of the global numbering of the vertices in the mesh. This is an advantage over ordering the mesh, as the input ordering is preserved, no parallel communication is required, and the method is valid for all meshes of any cell type. The approach is particularly appealing in high-performance computing applications, as it allows the use of high-degree elements on meshes of hexahedral cells.

REFERENCES

[1] Rainer Agelek, Michael Anderson, Wolfgang Bangerth, and William L. Barth. 2017. On orienting edges of unstructured two- and three-dimensional meshes. *ACM Transactions on Mathematical Software* 44, 1, Article 5 (2017), 22 pages. https://doi.org/10.1145/3061708

[2] Mark Ainsworth and Joe Coyle. 2003. Hierarchic finite element bases on unstructured tetrahedral meshes. *International Journal for Numerical Engineering* 58, 14 (2003), 2103–2130. https://doi.org/10.1002/nme.847

[3] Martin S. Alnæs, Jan Blechta, Johan Hake, August Johansson, Benjamin Kehlet, Anders Logg, Chris Richardson, Johannes Ring, Marie E. Rognes, and Garth N. Wells. 2015. The FEniCS project version 1.5. *Archive of Numerical Software* 3, 100 (2015), 9–23. https://doi.org/10.11588/ans.2015.100.20553

[4] Martin S. Alnæs, Anders Logg, and Kent-Andre Mardal. 2012. UFC: A finite element code generation interface. In *Automated Solution of Differential Equations by the Finite Element Method*, Anders Logg, Kent-Andre Mardal, and Garth N. Wells (Eds.). *Lecture Notes in Computational Science and Engineering*, Vol. 84. Springer, Berlin, Chapter 16, 283–302. https://doi.org/10.1007/978-3-642-23099-8_16

[5] Martin S. Alnaes, Anders Logg, Kristian B. Ølgaard, Marie E. Rognes, and Garth N. Wells. 2014. Unified form language: A domain-specific language for weak formulations of partial differential equations. *ACM Transactions on Mathematical Software* 40, Article 9 (2014), 37 pages. https://doi.org/10.1145/2566630

[6] Robert Anderson, Julian Andrej, Andrew Barker, Jamie Bramwell, Jean-Sylvain Camier, Jakub Cerveny, Veselin Dobrev, Yohann Dudouit, Aaron Fisher, Tzanio Kolev, et al. 2021. MFEM: A modular finite element methods library. *Computers & Mathematics with Applications* 81 (2021), 42–74. https://doi.org/10.1016/j.camwa.2020.06.009

[7] Daniel Arndt, Wolfgang Bangerth, Thomas C. Clevenger, Denis Davydov, Marc Fehling, Daniel Garcia-Sanchez, Graham Harper, Timo Heister, Luca Heltai, Martin Kronbichler, Ross Macguire Kynch, Matthias Maier, Jean-Paul Pelteret, Bruno Turcksin, and David Wells. 2019. The deal.II library, version 9.1. *Journal of Numerical Mathematics* 27, 4 (2019), 203–213. https://doi.org/10.1515/jnma-2019-0064

[8] Doug A. Arnold, Daniele Boffi, and Richard S. Falk. 2005. Quadrilateral H(div) finite elements. *SIAM Journal on Numerical Analysis* 42 (2005), 2429–2451. https://doi.org/10.1137/S0036142903431924

[9] Douglas N. Arnold and Gerard Awanou. 2011. The serendipity family of finite elements. *Foundations of Computational Mathematics* 11, 3 (2011), 337–344. https://doi.org/10.1007/s10208-011-9087-3

[10] Douglas N. Arnold and Gerard Awanou. 2013. Finite element differential forms on cubical meshes. *Mathematics of Computation* 83, 288 (2013), 1551–1570. https://doi.org/10.1090/s0025-5718-2013-02783-4

[11] Douglas N. Arnold and Anders Logg. 2014. Periodic table of the finite elements. *SIAM News* 47, 9 (2014). https://www-users.math.umn.edu/~arnold/femtable.

[12] Lourenço Beirão da Veiga, Franco Brezzi, Andrea Cangiani, Gianmarco Manzini, L. Donatella Marini, and Alessandro Russo. 2013. Basic principles of virtual element methods. *Mathematical Models and Methods in Applied Sciences* 23, 1 (2013), 199–214. https://doi.org/10.1142/S0218202512500492

[13] Timo Betcke and Matthew W. Scroggs. 2021. Retrieved February 25, 2021. Simple FEM-BEM Coupling for the Helmholtz Equation with FEniCSx. https://nbviewer.jupyter.org/github/bempp/bempp-cl/blob/master/notebooks/helmholtz/simple_helmholtz_fem_bem_coupling_dolfinx.ipynb.

[14] Philippe G. Ciarlet. 1978. *The Finite Element Method for Elliptic Problems.* North-Holland, Amsterdam.

[15] Federico Fuentes, Brendan Keith, Leszek Demkowicz, and Sriram Nagaraj. 2015. Orientation embedded high order shape functions for the exact sequence elements of all shapes. *Computers & Mathematics with Applications* 70, 4 (2015), 353–458. https://doi.org/10.1016/j.camwa.2015.04.027

[16] Paolo Gatto and Leszek Demkowicz. 2010. Construction of H^1-conforming hierarchical shape functions for elements of all shapes and transfinite interpolation. *Finite Elements in Analysis and Design* 46, 6 (2010), 474–486. https://doi.org/10.1016/j.finel.2010.01.005

[17] Miklós Homolya and David A. Ham. 2016. A parallel edge orientation algorithm for quadrilateral meshes. *SIAM Journal on Scientific Computing* 38, 5 (2016), S48–S61. https://doi.org/10.1137/15M1021325

[18] Claes Johnson and Jean-Claude Nédélec. 1980. On the coupling of boundary integral and finite element methods. *Mathematics of Computation* 35, 152 (1980), 1063–1079. https://doi.org/10.2307/2006375

[19] Robert C. Kirby. 2018. A general approach to transforming finite elements. *The SMAI Journal of Computational Mathematics* 4 (2018), 197–224. https://doi.org/10.5802/smai-jcm.33

[20] Robert C. Kirby, Anders Logg, Marie E. Rognes, and Andy R. Terrel. 2012. Common and unusual finite elements. In *Automated Solution of Differential Equations by the Finite Element Method*, Anders Logg, Kent-Andre Mardal, and Garth N. Wells (Eds.). *Lecture Notes in Computational Science and Engineering*, Vol. 84. Springer, Berlin, Chapter 3, 95–119. https://doi.org/10.1007/978-3-642-23099-8_3

[21] Anders Logg, Kent-Andre Mardal, Garth N. Wells, et al. 2012. *Automated Solution of Differential Equations by the Finite Element Method. Lecture Notes in Computational Science and Engineering*, Vol. 84. Springer, Berlin. https://doi.org/10.1007/978-3-642-23099-8

[22] Anders Logg and Garth N. Wells. 2010. DOLFIN: Automated finite element computing. *ACM Transactions on Mathematical Software* 37, 2, Article 20 (2010), 28 pages. https://doi.org/10.1145/1731022.1731030

[23] Jean-Claude Nédélec. 1980. Mixed finite elements in $\mathbb{R}^3$. *Numerische Mathematik* 35, 3 (1980), 315–341. https://doi.org/10.1007/BF01396415

[24] Jean-Claude Nédélec. 1986. A new family of mixed finite elements in $\mathbb{R}^3$. *Numerische Mathematik* 50, 1 (1986), 57–81. https://doi.org/10.1007/BF01389668

[25] Florian Rathgeber, David A. Ham, Lawrence Mitchell, Michael Lange, Fabio Luporini, Andrew T. T. McRae, Gheorghe-Teodor Bercea, Graham R. Markall, and Paul H. J. Kelly. 2016. Firedrake: Automating the finite element method by composing abstractions. *ACM Transactions on Mathematical Software* 43, 3, Article 24 (2016), 27 pages. https://doi.org/10.1145/2998441

[26] Pierre-Arnaud Raviart and Jean-Marie Thomas. 1977. A mixed finite element method for 2nd order elliptic problems. In *Mathematical Aspects of Finite Element Methods*, Ilio Galligani and Enrico Magenes (Eds.). *Lecture Notes in Mathematics*, Vol. 606. Springer, Berlin, 292–315. https://doi.org/10.1007/BFb0064470

[27] Marie E. Rognes, Robert C. Kirby, and Anders Logg. 2009. Efficient assembly of H(div) and H(curl) conforming finite elements. *SIAM Journal on Scientific Computing* 31, 6 (2009), 4130–4151. https://doi.org/10.1137/08073901X

[28] Joachim Schöberl. 1997. NETGEN: An advancing front 2D/3D-mesh generator based on abstract rules. *Computing and Visualization in Science* 1 (1997), 41–52. Issue 1. https://doi.org/10.1007/s007910050004

[29] Joachim Schöberl. 2014. *C++11 Implementation of Finite Elements in NGSolve.* Technical Report. Institute for Analysis and Scientific Computing, TU Wein. https://www.asc.tuwien.ac.at/~schoeberl/wiki/publications/ngs-cpp11.pdf.

[30] Matthew W. Scroggs, Igor A. Baratta, Chris N. Richardson, and Garth N. Wells. 2022. Basix: A runtime finite element basis evaluation library. (2022). submitted to Journal of Open Source Software.

[31] Barna Szabó and Ivo Babuška. 1991. *Finite Element Analysis.* Wiley.

[32] The DefElement contributors. 2021. DefElement: An encyclopedia of finite element definitions. Retrieved January 31, 2021 from https://defelement.com.

[33] The On-Line Encyclopedia of Integer Sequences. 2020. Sequence A334304. Retrieved January 31, 2021 from https://oeis.org/A334304.

[34] Sabine Zaglmayr. 2006. *High Order Finite Element Methods for Electromagnetic Field Computation.* Ph.D. Dissertation. Johannes Kepler Universität, Linz, Austria.