SMAI-JCM
SMAI Journal of Computational Mathematics

A general approach to transforming finite elements

Robert C. Kirby

Volume 4 (2018), p. 197-224.
<http://smai-jcm.cedram.org/item?id=SMAI-JCM_2018__4__197_0>

SMAI Journal of Computational Mathematics
Vol. 4, 197-224 (2018)

A general approach to transforming finite elements

Robert C. Kirby [1]

[1] Baylor University; Department of Mathematics; One Bear Place #97328; Waco, TX 76798-97328; USA
E-mail address: robert_kirby@baylor.edu.

Abstract. The use of a *reference element* on which a finite element basis is constructed once and mapped to each cell in a mesh greatly expedites the structure and efficiency of finite element codes. However, many famous finite elements such as Hermite, Morley, Argyris, and Bell, do not possess the kind of equivalence needed to work with a reference element in the standard way. This paper gives a generalized approach to mapping bases for such finite elements by means of studying relationships between the finite element nodes under push-forward.

This approach, developed through a sequence of examples of increasing complexity, requires one to study the relationship between the function space and degrees of freedom, or nodes, on a generic cell and the transformation of the corresponding entities on a reference cell. When the function space is preserved under mapping, one must be able to express the pushed-forward finite element nodes as linear combinations of the reference element finite element nodes. The transpose of this linear transformation maps the pull-back of the reference element basis functions to the desired finite element basis functions. Generically, developing this transformation for elements such as Morley and Hermite involves *completing* the set of finite element nodes, although the process is simplified in concrete ways when the finite elements form affine- or affine-interpolation equivalent families such as Lagrange or Hermite. When the finite element function space is not preserved under the pull-back, such as in the case of the Bell element, one applies the theory to an enriched finite element with a larger (but preserved) function space and additional nodes.

2010 Mathematics Subject Classification. 65N30.

Keywords. Finite element method, basis function, pull-back.

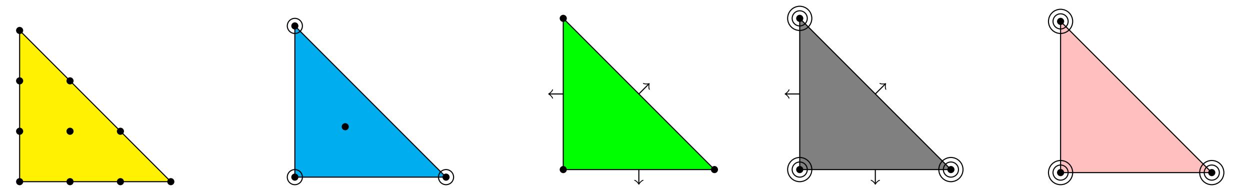
1. Introduction

At the heart of any finite element implementation lies the evaluation of basis functions and their derivatives on each cell in a mesh. These values are used to compute local integral contributions to stiffness matrices and load vectors, which are assembled into a sparse matrix and then passed on to an algebraic solver. While it is fairly easy to parametrize local integration routines over basis functions, one must also provide an implementation of those basis functions. Frequently, finite element codes use a *reference element*, on which a set of basis functions is constructed once and mapped via coordinate change to each cell in a mesh. This has significant advantages in terms of code reuse and modularity. For example, Jacobians can be computed centrally and shared among all finite element bases and integration loops.

Alternately, many finite element bases can be expressed in terms of barycentric coordinates, in which case one must simply convert between the physical and barycentric coordinates on each cell in order to evaluate basis functions. Although we refer the reader to recent results on *Bernstein polynomials* [1, 25] for interesting algorithms in the latter case, the prevelance of the reference element paradigm in modern high-level finite element software [4, 8, 27, 28, 32, 33] we shall restrict ourselves to the former.

The development of FIAT [24] has had a significant impact on finite element software, especially through its adoption in high-level software projects such as FEniCS [27] and Firedrake [33]. FIAT

This work was supported by NSF grant 1525697.



(a) Cubic Lagrange (b) Cubic Hermite (c) Morley (d) Quintic Argyris (e) Bell

Figure 1.1. Some famous triangular elements. Solid dots represent point value degrees of freedom, smaller circles represent gradients, and larger circles represent the collection of second derivatives. The arrows indicate directional derivatives evaluated at the tail of the arrow.

provides tools to describe and construct reference bases for arbitrary-order instances of many common and unusual finite elements. Composed with a domain-specific language for variational problems like UFL [2] and a form compiler mapping UFL into efficient code for element integrals [21, 26, 29] gives a powerful, user-friendly tool chain.

However, any code based on the reference element paradigm operates under the assumption that finite elements satisfy a certain kind of *equivalence*. Essentially, one must have a pull-back operation that puts basis functions on each cell into one-to-one correspondence with the reference basis functions. Hence, the original form of `ffc` [26] used only (arbitrary order) Lagrange finite elements, although this was generalized to $H(\text{div})$ and $H(\text{curl})$ elements using Piola transforms in [34]. Current technology captures the full simplicial discrete de Rham complex and certain other elements, but many famous elements are not included. Although it is possible to construct reference elements in FIAT or some other way, current form compilers or other high-level libraries do not provide correct code for mapping them. A major goal of this paper is to describe such mappings in order to understand how the capabilities of these high-level codes may be extended.

Elements such as Hermite [13], Argyris [3], Morley [31], and Bell [5], shown alongside the Lagrange element in Figure 1.1, do not satisfy the proper equivalence properties to give a simple relationship between the reference basis and nodal basis on a general cell. Typically, implementations of such elements require special-purpose code for constructing the basis functions separately on each element. Some older libraries, such as MODULEF [7] contain extensive sets of such elements. However, they seem to have fallen out of use in more recent software projects, perhaps as a result of the widespread use of the reference element paradigm. Although Bernadou gives a barycentric representation of the Argyris basis [6], Domínguez and Sayas [16] give a technique for mapping bases for the Argyris element and a separate computer implementation is available (`https://github.com/VT-ICAM/ArgyrisPack`), and Jardin [22] gives a per-element construction technique for the Bell element, these represent the exception rather than the rule. The literature contains no general approach for constructing and mapping finite element bases in the absence of affine equivalence or a suitable generalization thereof.

In this paper we provide such a general theory for transforming finite elements that supplements the theory on which FIAT is based for constructing those elements. This technique relies on constructing a matrix transforming the push-forward of the physical finite element nodes into the reference element nodes. The transpose of this matrix transforms the affine pull-back of the reference nodal basis into the physical nodal basis. For affine equivalent elements, this matrix is the identity, so no special treatment is required. For affine-interpolation equivalent families such as Hermite, there is a generally simple (block-diagonal) transformation depending on the the cell geometry. For elements such as Morley and Argyris that lack affine-interpolation equivalence, the transformation is more complicated. In this case, we suggest a three-part process. First, one extends the sets of physical and reference finite element nodes so that the spans of the extended reference nodes and push-forwards of the extended physical nodes coincide. Second, one must find a linear transformation relating the push-forward of the extended physical nodes to the extended reference nodes. Finally, one must express the restrictions of the extended physical nodes to the finite element space in terms of the original physical nodes by means of an interpolation-theoretic construction. This technique finds its maximal generality when extended to elements, such as Bell, for which the function space is not preserved under mapping. In this case, one applies the established mapping technique to a slightly enriched finite element for which a subset of the nodal basis functions turns out to be the correct nodal basis.

Our focus is on the case of scalar-valued elements in affine spaces, although we briefly discuss certain generalizations on both counts. We begin the rest of the paper by recalling definitions in §2. The bulk of the paper occurs in §3, where we show how to map finite element bases in the various situations when the function space is preserved under pull-back. We also sketch briefly how the theory is adapted to the case of more general pullbacks such as non-affine coordinate mappings or Piola transforms. In §4, we give the extension of this theory, with special attention to the Bell element, to the case when the function space is not preserved. Finally, in §5, we present some numerical results using these elements.

2. **Definitions and preliminaries**

Throughout, we let $\Omega \subset \mathbb{R}^d$ for $d = 2, 3$ be a bounded domain and also we let $C_b^k(\Omega)$ denote the space of functions with continuous and bounded derivatives up to and including order k over Ω, and $C_b^k(\Omega)'$ its topological dual.

Definition 2.1. A *finite element* is a triple (K, P, N) such that

- $K \subset \mathbb{R}^d$ is a bounded domain.
- $P \subset C_b^k(K)$ for some integer $k \geq 0$ is a finite-dimensional function space.
- $N = \{n_i\}_{i=1}^{\nu} \subset C_b^k(K)'$ is a collection of linearly independent linear functionals whose actions restricted to P form a basis for P'.

The nodes in N are taken as objects in the full infinite-dimensional dual, although sometimes we will only require their restrictions to members of P. For any $n \in C_b^k(K)'$, define $\pi n \in P'$ by restriction. That is, define $\pi n(p) = n(p)$ for any $p \in P$.

Further, with a slight abuse in notation, we will let $N = \begin{bmatrix} n_1 & n_2 & \dots & n_\nu \end{bmatrix}^T$ denote a functional on P^ν, or equivalently, a vector of ν members of the dual space.

As shorthand, we define these spaces consisting of vectors of functions or functionals by

$$\begin{aligned} X &\equiv (P)^\nu, \\ X^\dagger &\equiv \left(C_b^k(K)'\right)^\nu. \end{aligned} \tag{2.1}$$

We can "vectorize" the restriction operator π, so that for any $N \in X^\dagger$, $\pi N \in (P^\nu)'$ has $(\pi N)_i = \pi n_i$.

Galerkin methods work in terms of a basis for the approximating space, and these are typically built out of local bases for each element:

Definition 2.2. Let (K, P, N) be a finite element with $\dim P = \nu$. The *nodal basis* for P is the set $\{\psi_i\}_{i=1}^{\nu}$ such that $n_i(\psi_j) = \delta_{i,j}$ for each $1 \leq i, j \leq \nu$.

The nodal basis also can be written as $X \ni \Psi = \begin{bmatrix} \psi_1 & \psi_2 & \dots & \psi_\nu \end{bmatrix}$.

Traditionally, finite element codes construct the nodal basis for a *reference* finite element $\left(\hat{K}, \hat{P}, \hat{N}\right)$ and then map it into the basis for (K, P, N) for each K in the mesh. Let $F : K \to \hat{K}$ be the geometric mapping, as in Figure 2.1. We let J denote the Jacobian matrix of this transformation.

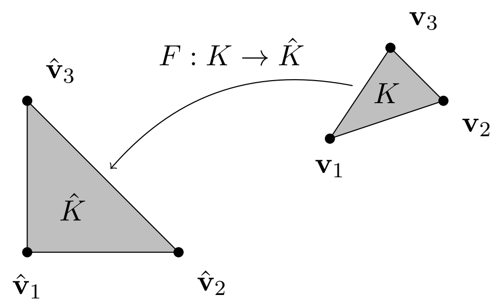


Figure 2.1. Affine mapping to a reference cell $\hat{K}$ from a typical cell K. Note that here F maps from the physical cell K to the reference cell $\hat{K}$ rather than the other way around. This leads to a more similar definition of the pull-back and push-forward.

Similarly to (2.1), we define the vector spaces relative to the reference cell:

$$\begin{aligned} \hat{X} &\equiv \left(\hat{P}\right)^{\nu}, \\ \hat{X}^{\dagger} &\equiv \left(C_b^k(\hat{K})'\right)^{\nu}. \end{aligned} \tag{2.2}$$

As with π, we define $\hat{\pi}\hat{n}$ as the restriction of $\hat{n}$ to $\hat{P}$, and can vectorize it over $\hat{X}^{\dagger}$ accordingly.

This geometric mapping induces a mapping between spaces of functions over K and $\hat{K}$ as well as between the dual spaces. These are called the pull-back, and push-forward operations, respectively:

Definition 2.3. The *pull-back* operation mapping $C_b^k(\hat{K}) \to C_b^k(K)$ is defined by

$$F^*\left(\hat{f}\right) = \hat{f} \circ F \tag{2.3}$$

for each $\hat{f} \in C_b^k(\hat{K})$.

Definition 2.4. The *push-forward* operation mapping the dual space $C_b^k(K)'$ into $C_b^k(\hat{K})'$ is defined by

$$F_*(n) = n \circ F^* \tag{2.4}$$

for each $n \in C_b^k(K)'$.

It is easy to verify that the pull-back and push-forward are linear operations preserving the vector space operations. Moreover, they are invertible iff F itself is. Therefore, we have

Proposition 2.5. *Given finite elements* (K, P, N) *and* $(\hat{K}, \hat{P}, \hat{N})$ *such that* $F(K) = \hat{K}$ *and* $F^*(\hat{P}) = P$, $F^* : \hat{P} \to P$ *and* $F_* : P' \to \hat{P}'$ *are isomorphisms.*

The pull-back and push-forward operations are also defined over the vector spaces X, $X^{\dagger}$, $\hat{X}$, and $\hat{X}^{\dagger}$. If N is a vector of functionals and Φ a vector of functions, then the vector push-forward and pull-back are, respectively

$$\begin{aligned} F_*(N) \in \hat{X}^{\dagger}, &\quad \left(F_*(N)\right)_i = F_*(n_i), \\ F^*(\hat{\Phi}) \in X, &\quad \left(F^*(\hat{\Phi})\right)_i = F^*(\hat{\phi}_i). \end{aligned} \tag{2.5}$$

It will also be useful to consider vectors of functionals acting on vectors of functions. We define this to produce a matrix as follows. If $N = \begin{bmatrix} n_1 & n_2 & \dots & n_k \end{bmatrix}^T$ is a collection of functionals and

$\Phi = \begin{bmatrix}\phi_1 & \phi_2 & \dots & \phi_\ell\end{bmatrix}^T$ a collection of functions, then we define the (outer) product $N(\Phi)$ to be the $k \times \ell$ matrix

$$(N(\Phi))_{ij} = n_i(\phi_j). \tag{2.6}$$

For example, if N is the vector of nodes of a finite element and Ψ contains the nodal basis functions, then the Kronecker delta property is expressed as $N(\Psi) = I$.

If M is a matrix of numbers of appropriate shape and $\Phi \in X$ members of a function space P, then $M\Phi$ is just defined by $(M\Phi)_i = \sum_{j=1}^{\nu} M_{ij}\Phi_j$, according to the usual rule for matrix-vector multiplication.

Lemma 2.6. *Let $N \in X^\dagger$ and $\Phi \in X$ and $M \in \mathbb{R}^{\nu\times\nu}$. Then*

$$N(M\Phi) = N(\Phi)M^T. \tag{2.7}$$

Proof. The proof is a simple calculation:

$$(N(M\Phi))_{ij} = n_i\left((M\Phi)_j\right) = n_i\left(\sum_{k=1}^{\nu} M_{jk}\phi_k\right) = \sum_{k=1}^{\nu} n_i M_{jk}(\phi_k) = \sum_{k=1}^{\nu} (N(\Phi))_{ik} M_{jk}.$$

■

The relationship between pull-back and push-forward also leads to the vectorized relation

Lemma 2.7. *Let $N \in X^\dagger$ and $\hat{\Phi} \in \hat{X}$. Then*

$$N(F^*(\hat{\Phi})) = F_*(N)(\hat{\Phi}) \tag{2.8}$$

Definition 2.8. Let (K, P, N) and $(\hat{K}, \hat{P}, \hat{N})$ be finite elements and F an affine mapping on K. Then (K, P, N) and $(\hat{K}, \hat{P}, \hat{N})$ are *affine equivalent* if

- $F(K) = \hat{K}$,
- The pullback maps $F^*(\hat{P}) = P$ (in the sense of equality of vector spaces),
- $F_*(N) = \hat{N}$ (in the sense of equality of finite sets).

Definition 2.9. Let (K, P, N) be a finite element of class C^k and $\Psi \in X$ its nodal basis. The *nodal interpolant* $\mathcal{I}_N : C_b^k(K) \to P$ is defined by

$$\mathcal{I}(f) = \sum_{i=1}^{\nu} n_i(f)\psi_i. \tag{2.9}$$

This interpolant plays a fundamental role in establishing approximation properties of finite elements via the Bramble-Hilbert Lemma [9, 17]. The homogeneity arguments in fact go through for the following generalized notion of element equivalence:

Definition 2.10. Two finite elements (K, P, N) and $(K, P, \tilde{N})$ are *interpolation equivalent* if $\mathcal{I}_N = \mathcal{I}_{\tilde{N}}$.

Definition 2.11. If $(K, P, \tilde{N})$ is affine equivalent to $(\hat{K}, \hat{P}, \hat{N})$ and interpolation equivalent to (K, P, N), then (K, P, N) and $(\hat{K}, \hat{P}, \hat{N})$ are *affine-interpolation equivalent.*

Brenner and Scott [10] give the following result, of which we shall make use:

Proposition 2.12. *Finite elements (K, P, N) and $(K, P, \tilde{N})$ are interpolation equivalent iff the spans of N and $\tilde{N}$, (viewed as subsets of $C_b^k(K)'$), are equal.*

For Lagrange and certain other finite elements, one simply has that $F^*(\hat{\Psi}) = \Psi$, which allows for the traditional use of reference elements used in FEniCS, Firedrake, and countless other codes. In fact, this property holds for Lagrange elements regardless of whether traditional equispaced points or some other more optimized point distribution [20] is used. It also holds for Bernstein polynomials, but for many other elements this is not the case. It is our goal in this paper to give a general approach that expresses Ψ as a linear transformation M applied to $F^*(\hat{\Psi})$, allowing the use of a reference element paradigm for a much broader class of elements.

Before proceeding, we note that approximation theory for Argyris and other families without affine-interpolation equivalence can proceed by means of establishing the *almost-affine* property [12]. Such proofs can involve embedding the inequivalent element family into an equivalent one with the requisite approximation properties. For example, the Argyris element is proved almost-affine by comparison to the "type (5)" quintic Hermite element. Although we see definite computational consequences of affine-equivalence, affine-interpolation equivalence, and neither among our element families, our approach to transforming inequivalent families does not make use of any almost-affine properties.

3. Transformation theory

For now, we assume that the pull-back operation (2.3) appropriately converts the reference element function space into the physical function space and discuss the construction of nodal bases based on relationships between the reference nodes $\hat{N}$ and the pushed-forward physical nodes $F_*(N)$.

We focus on the simplicial case, although generalizations do not have a major effect, as we note later. Throughout, we will use the following convention, developed in [34] for handling facet orientation in mixed methods but also useful in ordering higher-order Lagrange degrees of freedom. Since our examples are triangles (2-simplices), it is not necessary to expand on the entire convention. Given a triangle with vertices $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$, we define edge γ_i of the triangle to connect the vertices other than $\mathbf{v}_i$. The (unit) tangent vector $\mathbf{t}_i = \begin{bmatrix} t_i^{\mathbf{x}} & t_i^{\mathbf{y}} \end{bmatrix}^T$, points in the direction from the lower- to the higher-numbered vertex. When triangles share an edge, then, they agree on its orientation. The normal to an edge is defined by rotating the tangent by applying the matrix $R = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$ so that $\mathbf{n}_i = R\mathbf{t}_i = \begin{bmatrix} n_i^{\mathbf{x}} & n_i^{\mathbf{y}} \end{bmatrix}^T$ We also let $\mathbf{e}_i$ denote the midpoint of γ_i.

Now, we fix some notation for describing nodes. First, we define $\delta_{\mathbf{x}}$ acting on any continuous function by pointwise evaluation:

$$\delta_{\mathbf{x}}(p) = p(\mathbf{x}). \tag{3.1}$$

We let $\delta_{\mathbf{x}}^{\mathbf{s}}$ denote the directional derivative in direction some direction $\mathbf{s}$ at a point $\mathbf{x}$, so that

$$\delta_{\mathbf{x}}^{\mathbf{s}}(p) = \mathbf{s}^T \nabla p(\mathbf{x}). \tag{3.2}$$

We use repeated superscripts to indicate higher-order derivatives, so that $\delta_{\mathbf{x}}^{\mathbf{xx}}$ defines the second directional derivative along the x-axis at point $\mathbf{x}$, for example.

It will also be convenient to use block notation, with a single symbol representing two or more items. For example, the gradient notation

$$\nabla_{\mathbf{x}} = \begin{bmatrix} \delta_{\mathbf{x}}^{\mathbf{x}} & \delta_{\mathbf{x}}^{\mathbf{y}} \end{bmatrix}^T$$

gives the pair of functionals evaluating the Cartesian derivatives at a point $\mathbf{x}$. To denote a gradient in a different basis, we append the directions as superscripts so that

$$\nabla_{\mathbf{x}}^{\mathbf{nt}} = \begin{bmatrix} \delta_{\mathbf{x}}^{\mathbf{n}} & \delta_{\mathbf{x}}^{\mathbf{t}} \end{bmatrix}^T$$

contains the normal and tangential derivatives at a point $\mathbf{x}$.

Similarly, we let

$$\triangle_{\mathbf{v}} = \begin{bmatrix} \delta_{\mathbf{x}}^{\mathbf{xx}} & \delta_{\mathbf{x}}^{\mathbf{xy}} & \delta_{\mathbf{x}}^{\mathbf{yy}} \end{bmatrix}^T$$

denote the vector of three functionals evaluating the unique (supposing sufficient smoothness) second partials at $\mathbf{x}$.

Let $\Psi = \{\psi_i\}_{i=1}^{\nu}$ be the nodal basis for a finite element (K, P, N) and $\hat{\Psi} = \{\hat{\psi}_i\}_{i=1}^{\nu}$ that for a reference element $\left(\hat{K}, \hat{P}, \hat{N}\right)$. We also assume that $F(K) = \hat{K}$ and $F^*(\hat{P}) = P$. Because the pull-back is invertible, it maps linearly independent sets to linearly independent sets. Therefore, $F^*(\hat{\Psi})$ must also be a basis for P. There exists an invertible $\nu \times \nu$ matrix M such that

$$\Psi = MF^*(\hat{\Psi}), \tag{3.3}$$

or equivalently, that each nodal basis function is some linear combination of the pull-backs of the reference nodal basis functions.

Our theory for transforming the basis functions (i. e. computing the matrix M) will work via duality, relating the matrix M to how the nodes (or at least their restrictions to the finite-dimensional spaces) push forward.

It will be useful to define as an intermediate $\nu \times \nu$ matrix $B = F_*(N)(\hat{\Psi})$. Recall from (2.6) that its entries for $1 \leq i, j \leq \nu$ are

$$B_{ij} \equiv F_*(n_i)(\hat{\psi}_j) = n_i(F^*(\hat{\psi}_j)) = n_i(\hat{\psi}_j \circ F) \tag{3.4}$$

This matrix, having nodes only applied to members of P, is indifferent to restrictions and so $B = F_*(\pi N)(\hat{\Psi})$ as well.

Because of Proposition 2.5 and finite-dimensionality, the the nodal sets $\hat{\pi}\hat{N}$ and $F_*(\pi N)$ are both bases for $\hat{P}'$, and so there exists an invertible $\nu \times \nu$ matrix V such that

$$\hat{\pi}\hat{N} = VF_*(\pi N). \tag{3.5}$$

Frequently, it may be easier to express the pushed-forward nodes as a linear combination of the reference nodes. In this case, one obtains the matrix V^{-1}. At any rate, the matrices V and M are closely related.

Theorem 3.1. *For finite elements (K, P, N) and $(\hat{K}, \hat{P}, \hat{N})$ with $F(K) = \hat{K}$ and $F_*(\hat{P}) = P$, the matrices in* (3.3) *and* (3.5) *satisfy*

$$M = V^T. \tag{3.6}$$

Proof. We proceed by relating both matrices to B defined in (3.4) via the Kronecker property of nodal bases. First, we have

$$I = N(\Psi) = N(MF^*(\hat{\Psi})) = N(F^*(\hat{\Psi}))M^T = BM^T.$$

so that

$$M = B^{-T}. \tag{3.7}$$

Similarly,

$$I = (VF_*(N))\,(\hat{\Psi}) = VF_*(N)(\hat{\Psi}) = VB,$$

so that $V = B^{-1}$ and the result follows. ∎

To relate the pullback of the reference element basis functions to any element's basis functions, it is sufficient to determine the relationship between the nodes.

3.1. **Affine equivalence: The Lagrange element**

When elements form affine-equivalent families, the matrix M has a particularly simple form.

Theorem 3.2. *If (K, P, N) and $(\hat{K}, \hat{P}, \hat{N})$ are affine-equivalent finite elements then the transformation matrix M is the identity.*

Proof. Suppose the two elements are affine-equivalent, so that $F_*(N) = \hat{N}$. Then, a direct calculation gives

$$N(F^*(\hat{\Psi})) = F_*(N)(\hat{\Psi}) = \hat{N}(\hat{\Psi}) = I,$$

so that $M = I$. ■

The *Lagrange* elements are the most widely used finite elements and form the prototypical affine-equivalent family [10]. For a simplex K in dimension d and integer $r \geq 1$, one defines $P = P_r(K)$ to be the space of polynomials over K of total degree no greater than r, which has dimension $\binom{r+d}{d}$. The nodes are taken to be pointwise evaluation at a lattice of $\binom{r+d}{d}$ points. Classically, these are taken to be regular and equispaced, although options with superior interpolation and conditioning properties for large r are also known [20]. One must ensure that nodal locations are chosen at the boundary to enable C^0 continuity between adjacent elements. A cubic Lagrange triangle ($r = 3$ and $d = 2$) is shown earlier in Figure 1.1a.

The practical effect of Theorem 3.2 is that the reference element paradigm "works " – computer code contains a routine to evaluate the nodal basis $\hat{\Psi}$ and its derivatives for a reference element $(\hat{K}, \hat{P}, \hat{N})$. Then, this routine is called at a set of quadrature points in $\hat{K}$. One obtains values of the nodal basis at quadrature points on each cell K by pull-back, so no additional work is required. To obtain the gradients of each basis function at each quadrature point, one simply multiplies each basis gradient at each point by J^T.

On the other hand, when $M \neq I$, the usage of tabulated reference values is more complex. Given a table

$$\hat{\mathit{\Psi}}_{iq} = \hat{\psi}_i(\hat{\xi}_q) \tag{3.8}$$

of the reference basis at the reference quadrature points, one finds the nodal basis for (K, P, N) by constructing M for that element and then computing the matrix-vector product $M\hat{\mathit{\Psi}}$ so that

$$\psi_i(\xi_q) = \sum_{k=1}^{\nu} M_{i,k} \hat{\mathit{\Psi}}_{k,q} \tag{3.9}$$

Mapping gradients from the reference element requires both multiplication by M as well as application of J^T by the chain rule. We define $D\hat{\mathit{\Psi}} \in \mathbb{R}^{\nu \times |\xi| \times 2}$ by

$$D\hat{\mathit{\Psi}}_{i,q,:} = \hat{\nabla} \hat{\psi}_i(\hat{\xi})_q. \tag{3.10}$$

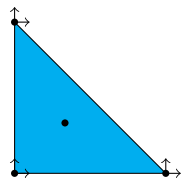
Then, the basis gradients requires contraction with M

$$D\mathit{\Psi}'_{i,q,:} := \sum_{k=1}^{\nu} M_{i,k} D\hat{\mathit{\Psi}}_{k,q,:}, \tag{3.11}$$

followed by the chain rule

$$D\mathit{\Psi}_{i,q,:} := J^T D\mathit{\Psi}'_{i,q,:}. \tag{3.12}$$

In fact, the application of M and J^T can be performed in either order. Note that applying M requires an $\nu \times \nu$ matrix-vector multiplication and in principle couples all basis functions together, while applying J^T works pointwise on each basis function separately. When M is quite sparse, one expects this to be a small additional cost compared to the other required arithmetic. We present further details for this in the case of triangular Hermite elements, to which we now turn.


(a) Reference Hermite element

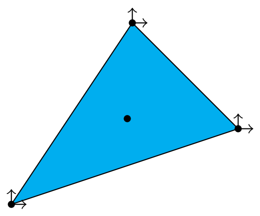

(b) Physical Hermite element

Figure 3.1. Reference and physical cubic Hermite elements with gradient degrees of freedom expressed in terms of local Cartesian directional derivatives.

3.2. The Hermite element: affine-interpolation equivalence

The Hermite triangle [13], shown in Figure 1.1b is based cubic polynomials, although higher-order instances can also be defined [10]. In contrast to the Lagrange element, its node set includes function values and derivatives at the nodes, as well as an interior function value. The resulting finite element spaces have C^0 continuity with C^1 continuity at vertices. They provide a classic example of elements that are not affine equivalent but instead give affine-interpolation equivalent families.

We will let (K, P, N) be a cubic Hermite triangle, specifying the gradient at each vertex in terms of the Cartesian derivatives – see Figure 3.1b. Let $\{\mathbf{v}_i\}_{i=1}^3$ be the three vertices of K and $\mathbf{v}_4$ its barycenter. We order the nodes N by

$$N = \begin{bmatrix}\delta_{\mathbf{v}_1} & \nabla^T_{\mathbf{v}_1} & \delta_{\mathbf{v}_2} & \nabla^T_{\mathbf{v}_2} & \delta_{\mathbf{v}_3} & \nabla^T_{\mathbf{v}_3} & \delta_{\mathbf{v}_4}\end{bmatrix}^T, \tag{3.13}$$

using block notation.

Now, we fix the reference element $(\hat{K}, \hat{P}, \hat{N})$ with $\hat{K}$ as the unit right triangle and express the gradient by the derivatives in the direction of the reference Cartesian coordinates, as in Figure 3.1a. Let $\{\hat{\mathbf{v}}_i\}_{i=1}^3$ be the three vertices of $\hat{K}$ and $\hat{\mathbf{v}}_4$ its barycenter. We define $\hat{N}$ analogously to N.

Consider the relationship between the nodal basis functions Ψ and the pulled-back $F^*(\hat{\Psi})$. For any $\hat{\psi} \in \hat{P}$, the chain rule leads to

$$\nabla(\hat{\psi} \circ F) = J^T \hat{\nabla}\hat{\psi} \circ F. \tag{3.14}$$

Now, suppose that $\hat{\psi}$ is a nodal basis function corresponding to evaluation at a vertex or the barycenter, so that $\delta_{\hat{\mathbf{v}}_i}\hat{\psi} = 1$ for some $1 \leq i \leq 4$, with the remaining reference nodes vanishing on $\hat{\psi}$. We compute that

$$\delta_{\mathbf{v}_i} F^*(\hat{\psi}) = (\hat{\psi} \circ F)(\mathbf{v}_i) = \hat{\psi}(\hat{v}_i) = 1,$$

while $\delta_{\mathbf{v}_j} F^*(\hat{\psi}) = 0$ for $1 \leq j \leq 4$ with $j \neq i$. Also, since the reference gradient of $\hat{\psi}$ vanishes at each vertex, (3.14) implies that the physical gradient of $F^*(\hat{\psi})$ must also vanish at each vertex. Pulling back $\hat{\psi}$ gives the corresponding nodal basis function for (K, P, N).

The situation changes for the derivative basis functions. Now take $\hat{\psi}$ to be the basis function with unit-valued derivative in, say, the $\hat{\mathbf{x}}$ direction at vertex $\hat{\mathbf{v}}_i$ and other degrees of freedom vanishing. Since it vanishes at each vertex and the barycenter of $\hat{K}$, $F^*(\hat{\psi})$ will vanish at each vertex and the barycenter of K. The reference gradient of $\hat{\psi}$ vanishes at the vertices other than i, so the physical gradient of its pullback must also vanish at the corresponding vertices of K. However, (3.14) shows that $\nabla(\hat{\psi} \circ F)$ will typically not yield $\begin{bmatrix}1 & 0\end{bmatrix}^T$ at $\mathbf{v}_i$. Consequently, the pull-backs of the reference derivative basis functions do not produce the physical basis functions.

Equivalently, we may express this failure in terms of the nodes – pushing forward N does not yield $\hat{N}$. We demonstrate this pictorially in Figure 3.2, showing the images of the derivative nodes under

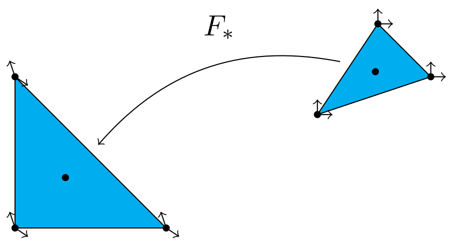


Figure 3.2. Pushing forward the Hermite derivative nodes in physical space does *not* produce the reference derivative nodes.

push-forward do not correspond to the reference derivative nodes. Taking this view allows us to address the issue using Theorem 3.1.

This discussion using the chain rule can be summarized by the matrix-valued equation

$$F_*(N) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & J^T & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & J^T & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & J^T & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \hat{N}, \tag{3.15}$$

noting that the second, fourth, and sixth rows and columns of this matrix are blocks of two, and each "0" is taken to be the zero matrix of appropriate size. This is exactly the inverse of V from Theorem 3.1.

In this case, the transformation V is quite local – that is, only the push-forward of nodes at a given point are used to construct the reference nodes at the image of that point. This seems to be generally true for interpolation-equivalent elements, although functionals with broader support (e.g. integral moments over the cell or a facet thereof) would require a slight adaptation. We will see presently for Morley and Argyris elements that the transformation neeed not be block diagonal for elements without interpolation equivalence. At any rate, the following elementary observation from linear algebra suggests the sparsity of V:

Proposition 3.3. *Let W be a vector space with sets of vectors $W_1 = \{w_i^1\}_{i=1}^m \subset W$ and $W_2 = \{w_i^2\}_{i=1}^n$. Suppose that* $\mathrm{span} W_1 \subset \mathrm{span W_2}$ *so that there exists a matrix $A \in \mathbb{R}^{m\times n}$ such that $w_i^1 = \sum_{k=1}^n A_{ik} w_k^2$. If we further have that some $w_i^1 \in \mathrm{span}\{w_j^2\}_{j\in\mathcal{J}}$ for some $\mathcal{J} \subset [1,n]$, then $A_{ij} = 0$ for all $j \notin \mathcal{J}$.*

Our theory applies equally to the general family of Hermite triangles of degree $k \geq 3$. In those cases, the nodes consist of gradients at vertices together with point-wise values at appropriate places. All higher-order cases generate C^0 families of elements with C^1-continuity at vertices. The V matrix remains analogous to the cubic case, with J^{-T} on the diagonal in three places corresponding to the vertex derivative nodes. No major differences appear for the tetrahedral Hermite elements, either.

As we saw earlier, Hermite and other elements for which $M \neq I$ incur an additional cost in mapping from the reference element, as one must compute basis function values and gradients via (3.9) and (3.12). The key driver of this additional cost is the application of M. Since M is very sparse for Hermite elements – just 12 nonzeros counting the 1's on the diagonal – evaluating (3.9) requires just 12 operations per column, so a 10-point quadrature rule requires 120 operations. Evaluating (3.11) requires twice this, or 240 operations. Applying J^T in (3.12) is required whether Hermite or Lagrange elements are used. It requires 4×10 times the number of quadrature points used – so a 10-point rule would require 400 operations. Hence, the chain rule costs more than the application of M in this situation. On the other hand, building an element stiffness matrix requires a double loop over these 10 basis functions nested with a loop over the, say, 10 quadrature points. Hence, the deepest part of the loop nest requires 1000 iterations, and with even a handful of operations will easily dominate the additional cost of multiplying by M.

3.3. **The Morley and Argyris elements**

The construction of C^1 finite elements, required for problems such as plate bending or the Cahn-Hilliard equations, is a long-standing difficulty. Although it is possible to work around this requirement by rewriting the fourth-order problem as a lower order system or by using C^0 elements in conjunction with variational form penalizing the jumps in derivatives [18, 35], this does not actually give a C^1 solution.

The quadratic Morley triangle [31], shown in Figure 1.1c, finds application in plate-bending problems and also provides a relatively simple motivation for and application of the theory developed here. The six degrees of freedom, vertex values and the normal derivatives on each edge midpoint, lead to an assembled finite element space that is neither C^0 nor C^1, but it is still suitable as a convergent nonconforming approximation for fourth-order problems.

The quintic Argyris triangle [3], shown in Figure 1.1d, with its 21 degrees, gives a proper C^1 finite element. Hence it can be used generically for fourth-order problems as well as second-order problems for which a continuously differentiable solution is desired. The Argyris elements use the values, gradients, and second derivatives at each triangle vertex plus the normal derivatives at edge midpoints as the twenty-one degrees of freedom.

It has been suggested that the Bell element [5] represents a simpler C^1 element than the Argyris element, on the account that it has fewer degrees of freedom. Shown in Figure 1.1e, we see that the edge normal derivatives have been removed from the Argyris element. However, this comes with a (smaller but) more complicated function space. Rather than full quintic polynomials, the Bell element uses quintic polynomials that have normal derivatives on each edge of only third degree. This constraint on the polynomial space turns out to complicate the transformation of Bell elements compared to Hermite or even Argyris. For the rest of this section, we focus on Morley and Argyris, returning to Bell later.

It can readily be seen that, like the Hermite element, the standard affine mapping will not preserve nodal bases. Unlike the Hermite element, however, the Morley and Argyris elements do not form affine-interpolation equivalent families – the spans of the nodes are not preserved under push-forward thanks to the edge normal derivatives – see Figure 3.3. As the Morley and Aryris nodal sets do not contain a full gradient at edge midpoints, the technique used for Hermite elements cannot be directly applied.

To address this, we introduce the following idea:

Definition 3.4. Let (K, P, N) and $(\hat{K}, \hat{P}, \hat{N})$ be finite elements of class C^k with affine mapping $F : K \to \hat{K}$ and associated pull-back and push-forward F^* and F_*. Suppose also that $F^*(\hat{P}) = P$. Let $N^c = \{n_i^c\}_{i=1}^{\mu} \subset C_b^k(K)'$ and $\hat{N}^c = \{\hat{n}_i^c\}_{i=1}^{\mu} \subset C^k(\hat{K})'$ be such that

- $N \subset N^c$ (taken as sets rather than vectors),
- $\hat{N} \subset \hat{N}^c$ (again as sets),
- $\operatorname{span}(F_*(N^c)) = \operatorname{span}(\hat{N}^c)$ in $C^k(\hat{K})'$.

Then N^c and $\hat{N}^c$ form a *compatible nodal completion* of N and $\hat{N}$.

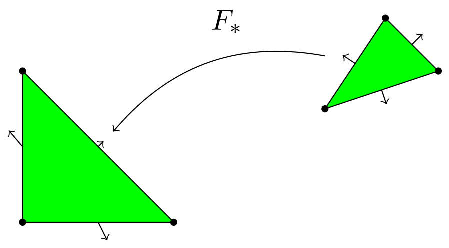


Figure 3.3. Pushing forward the Morley derivative nodes in physical space does *not* produce the reference derivative nodes.

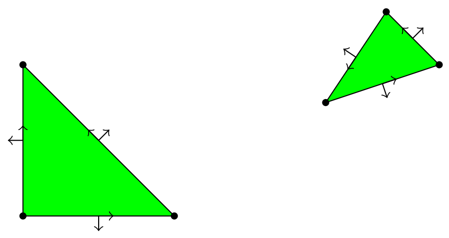


Figure 3.4. Nodal sets $\hat{N}^c$ and N^c giving the compatible nodal completion of N and $\hat{N}$ for a Morley element and reference element are formed by including tangential derivatives along with normal derivatives at each edge midpoint.

Example 3.5. Let (K, P, N) and $(\hat{K}, \hat{P}, \hat{N})$ be the Morley triangle and reference triangle. Take N^c to contain all the nodes of N together with the tangential derivatives at the midpoint of each edge of K and similarly for $\hat{N}^c$. In this case, $\mu = 9$. Then, both N^c and $\hat{N}^c$ contain complete gradients at each edge midpoint and function values at each vertex. The push-forward of N^c has the same span as $\hat{N}^c$ and so N^c and $\hat{N}^c$ form a compatible nodal completion of N and $\hat{N}$. This is shown pictorially in Figure 3.4.

A similar completion – supplementing the nodes with tangential derivatives at edge midpoints – exists for the Argyris nodes and reference nodes [16].

Now, since the spans of $\hat{N}^c$ and $F_*(N^c)$ agree (even in $C_b^k(\hat{K})'$), there exists a $\mu \times \mu$ matrix V^c, typically block diagonal, such that

$$\hat{N}^c = V^c F_*(N^c). \tag{3.16}$$

Let $E \in \mathbb{R}^{\nu \times \mu}$ be the Boolean matrix with $E_{ij} = 1$ iff $\hat{n}_i = \hat{n}_j^c$ so that

$$\hat{N} = E\hat{N}^c, \tag{3.17}$$

and it is clear that

$$\hat{N} = EV^c F_*(N^c). \tag{3.18}$$

In other words, the reference nodes are linear combinations of the pushed-forward nodes *and* the extended nodes, but we must have the linear combination in terms of the pushed-forward nodes alone.

Recall that building the nodal basis only requires the action of the nodes on the polynomial space. Because $\mu > \nu$, the set of nodes πN^c must be linearly dependent. We seek a matrix $D \in \mathbb{R}^{\mu \times \nu}$ such that

$$\pi N^c = D\pi N. \tag{3.19}$$

Since F_* is an isomorphism, such a D also gives

$$\hat{\pi} F_*(N^c) = D\hat{\pi} F_*(N). \tag{3.20}$$

Rows i of the matrix D such that $n_i^c = n_j$ for some j will just have $D_{ik} = \delta_{kj}$ for $1 \leq k \leq \nu$. The remaining rows must be constructed somehow via an interpolation argument, although the details will vary by element.

This discussion suggests a three-stage process, each encoded by matrix multiplication, for converting the push-forwards of the physical nodes to the reference nodes, hence giving a factored form of V in (3.5). Before working examples, we summarize this in the following theorem:

Theorem 3.6. *Let* (K, P, N) *and* $(\hat{K}, \hat{P}, \hat{N})$ *be finite elements with affine mapping* $F : K \to \hat{K}$ *and suppose that* $F^*(\hat{P}) = P$. *Let* N^c *and* $\hat{N}^c$ *be a compatible nodal completion of* N *and* $\hat{N}$. *Then given matrices* $E \in \mathbb{R}^{\nu\times\mu}$ *from* (3.17), $V^c \in \mathbb{R}^{\mu\times\mu}$ *from* (3.16) *and* $D \in \mathbb{R}^{\mu\times\nu}$ *from* (3.19) *that builds the (restrictions of) the extended nodes out of the given physical nodes, the nodal transformation matrix* V *satisfies*

$$V = EV^cD. \tag{3.21}$$

This gives the general outline for mapping finite elements whose function spaces are preserved under pull-back. One first extends the nodes to a set whose span is preserved under push-forward (the matrix D). Then, the extended nodes must be mapped by push-forward (V^c), and the nodes of the finite element extracted (E). The transpose of the resulting product then constructs the nodal basis functions from the pull-back of the reference element nodal basis functions. For affine-interpolation equivalent families, V simplifies to the single matrix V^c, which further simplifies to the identity for affine-equivalent families. To illustrate the more general setting, we now turn to the Morley and Argyris elements.

3.3.1. *The Morley element*

Following our earlier notation for the geometry and nodes, we order the nodes of a Morley triangle by

$$N = \begin{bmatrix}\delta_{\mathbf{v}_1} & \delta_{\mathbf{v}_2} & \delta_{\mathbf{v}_3} & \delta_{\mathbf{e}_1}^{\mathbf{n}_1} & \delta_{\mathbf{e}_2}^{\mathbf{n}_2} & \delta_{\mathbf{e}_3}^{\mathbf{n}_3}\end{bmatrix}^T \tag{3.22}$$

Nodes N^c will also include tangential derivatives at the edge midpoint. We put

$$N^c = \begin{bmatrix}\delta_{\mathbf{v}_1} & \delta_{\mathbf{v}_2} & \delta_{\mathbf{v}_3} & (\nabla_{\mathbf{e}_1}^{\mathbf{n}_1\mathbf{t}_1})^T & (\nabla_{\mathbf{e}_2}^{\mathbf{n}_2\mathbf{t}_2})^T & (\nabla_{\mathbf{e}_3}^{\mathbf{n}_3\mathbf{t}_3})^T\end{bmatrix}^T, \tag{3.23}$$

Again, this is a block vector the last three entries each consist of two values. We give the same ordering of reference element nodes $\hat{N}$ and $\hat{N}^c$.

The matrix E simply extracts the members of N^c that are also in N, so with $\eta = \begin{bmatrix}1 & 0\end{bmatrix}$, we have the block matrix

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \eta & 0 & 0 \\ 0 & 0 & 0 & 0 & \eta & 0 \\ 0 & 0 & 0 & 0 & 0 & \eta \end{bmatrix}. \tag{3.24}$$

Because the gradient nodes in N^c use normal and tangential coordinates, V^c will be slightly more more complicated than V for the Hermite element. For local edge γ_i, we define the (orthogonal) matrix

$$G_i = \begin{bmatrix}\mathbf{n}_i & \mathbf{t}_i\end{bmatrix}^T$$

with the normal and tangent vector in the rows. Similarly, we let

$$\hat{G}_i = \begin{bmatrix}\hat{\mathbf{n}}_i & \hat{\mathbf{t}}_i\end{bmatrix}^T$$

contain the unit normal and tangent to edge $\hat{\gamma}_i$ of the reference cell $\hat{K}$. It is clear that

$$F_*(\nabla^{\mathbf{n}_i\mathbf{t}_i}_{\mathbf{e}_i}) = F_*(G_i \nabla_{\mathbf{e}_i}) = G_i F_*(\nabla_{\mathbf{e}_i}) = G_i J^T \hat{\nabla}_{\mathbf{e}_i} = G_i J^T \hat{G}_i^T \hat{\nabla}^{\hat{\mathbf{n}}_i \hat{\mathbf{t}}_i}_{\hat{\mathbf{e}}_i}, \tag{3.25}$$

so, defining

$$B^i = (G_i J^T \hat{G}_i^T)^{-1} = \hat{G}_i J^{-T} G_i^T, \tag{3.26}$$

we have that

$$V^c = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & B^1 & 0 & 0 \\ 0 & 0 & 0 & 0 & B^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & B^3 \end{bmatrix}. \tag{3.27}$$

Now, we turn to the matrix $D \in \mathbb{R}^{9 \times 6}$, writing members of πN^c in terms of πN alone. The challenge is to express the tangential derivative nodes in terms of the remaining six nodes – vertex values and normal derivatives. In fact, only the vertex values are needed. Along any edge, any member of P is just a univariate quadratic polynomial, and so the tangential derivative is linear. Linear functions attain their average value over an interval at its midpoint. But the average value of the derivative over the edge is just the difference between vertex values divided by the edge length. The matrix D must be

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -\ell_1^{-1} & \ell_1^{-1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ -\ell_2^{-1} & 0 & \ell_2^{-1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -\ell_3^{-1} & \ell_3^{-1} & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.28}$$

We can also arrive at this formulation of D in another way, that sets up the discussion used for Argyris and later Bell elements. Consider the following (very elementary) univariate result

Proposition 3.7. *Let $p(x)$ any quadratic polynomial on $[-1, 1]$. Then*

$$p'(0) = \tfrac{1}{2}\left(p(1) - p(-1)\right) \tag{3.29}$$

Then, by a change of variables, this rule can be mapped to $\left[-\frac{\ell}{2}, \frac{\ell}{2}\right]$ so that

$$p'(0) = \tfrac{1}{\ell}\left(p(\tfrac{\ell}{2}) - p(-\tfrac{\ell}{2})\right).$$

Finally, one can apply this rule on the edge of a triangle running from $\mathbf{v}_a$ to $\mathbf{v}_b$ to find that

$$\pi\delta^{\mathbf{t}_i} = \tfrac{\ell}{2}\left(\pi\delta_{\mathbf{v}_b} - \pi\delta_{\mathbf{v}_a}\right).$$

It is interesting to explicitly compute the product $V = EV^cD$, as giving a single formula rather than product of matrices is more useful in practice. Multiplying through gives:

$$V = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & \frac{-B^1_{12}}{\ell_1} & \frac{B^1_{12}}{\ell_1} & B^1_{11} & 0 & 0 \\ \frac{-B^2_{12}}{\ell_2} & 0 & \frac{B^2_{12}}{\ell_2} & 0 & B^2_{11} & 0 \\ \frac{-B^3_{12}}{\ell_3} & \frac{B^3_{12}}{\ell_3} & 0 & 0 & 0 & B^3_{11} \end{bmatrix} \tag{3.30}$$

From the definition of B^i, it is possibly to explicitly calculate its entries in terms of the those of the Jacobian and the normal and tangent vectors for K and $\hat{K}$. Only the first row of each B^i is needed

$$\begin{aligned} B^i_{11} &= \hat{n}^{\mathbf{x}}_i \left(n^{\mathbf{x}}_i \tfrac{\partial x}{\partial \hat{x}} + t^{\mathbf{x}}_i \tfrac{\partial y}{\partial \hat{x}} \right) + \hat{t}^{\mathbf{x}}_i \left(n^{\mathbf{x}}_i \tfrac{\partial x}{\partial \hat{y}} + t^{\mathbf{x}}_i \tfrac{\partial y}{\partial \hat{y}} \right) \\ B^i_{12} &= \hat{n}^{\mathbf{x}}_i \left(n^{\mathbf{y}}_i \tfrac{\partial x}{\partial \hat{x}} + t^{\mathbf{y}}_i \tfrac{\partial y}{\partial \hat{x}} \right) + \hat{t}^{\mathbf{x}}_i \left(n^{\mathbf{y}}_i \tfrac{\partial x}{\partial \hat{y}} + t^{\mathbf{y}}_i \tfrac{\partial y}{\partial \hat{y}} \right) \end{aligned} \tag{3.31}$$

We can also recall that the normal and tangent vectors are related by $n^{\mathbf{x}} = t^{\mathbf{y}}$ and $n^{\mathbf{y}} = -t^{\mathbf{x}}$ to express these entries purely in terms of either the normal or tangent vectors. Each entry of the Jacobian and normal and tangent vectors of K and $\hat{K}$ enter into the transformation.

In this form, V has 12 nonzero entries, although the formation of those entries, which depend on normal and tangent vectors and the Jacobian, from the vertex coordinates requires an additional amount of arithmetic. The Jacobian will typically be computed anyway in a typical code, and the cost of working with $M = V^T$ will again be subdominant to the nested loops over basis functions and quadrature points required to form element matrices, much like Hermite.

3.3.2. *The Argyris element*

Because it is higher degree than Morley and contains second derivatives among the nodes, the Argyris transformation is more involved. However, it is a prime motivating example and also demonstrates that the general theory here reproduces the specific technique in [16]. The classical Argyris element has P as polynomials of degree 5 over a triangle K, a 21-dimensional space. The 21 associated nodes N are selected as the point values, gradients, and all three unique second derivatives at the vertices together with the normal derivatives evaluated at edge midpoints. These nodal choices lead to a proper C^1 element, and C^2 continuity is obtained at vertices.

Since the Argyris elements do not form an affine-interpolation equivalent family, we will need to embed the physical nodes into a larger set. Much as with Morley elements, the edge normal derivatives will be augmented by the tangential derivatives.

With this notation, N is a vector of 21 functionals and N^c a vector of 24 functions written as

$$\begin{aligned} N &= \left[\begin{array}{cccccccccccc} \delta_{\mathbf{v}_1} & \nabla_{\mathbf{v}_1} & \triangle_{\mathbf{v}_1} & \delta_{\mathbf{v}_2} & \nabla_{\mathbf{v}_2} & \triangle_{\mathbf{v}_2} & \delta_{\mathbf{v}_3} & \nabla_{\mathbf{v}_3} & \triangle_{\mathbf{v}_3} & \delta^{\mathbf{n}_1}_{\mathbf{e}_1} & \delta^{\mathbf{n}_2}_{\mathbf{e}_2} & \delta^{\mathbf{n}_3}_{\mathbf{e}_3} \end{array} \right]^T, \\ N^c &= \left[\begin{array}{cccccccccccc} \delta_{\mathbf{v}_1} & \nabla_{\mathbf{v}_1} & \triangle_{\mathbf{v}_1} & \delta_{\mathbf{v}_2} & \nabla_{\mathbf{v}_2} & \triangle_{\mathbf{v}_2} & \delta_{\mathbf{v}_3} & \nabla_{\mathbf{v}_3} & \triangle_{\mathbf{v}_3} & \nabla^{\mathbf{n}_1\mathbf{t}_1}_{\mathbf{v}_1} & \nabla^{\mathbf{n}_2\mathbf{t}_2}_{\mathbf{v}_2} & \nabla^{\mathbf{n}_3\mathbf{t}_3}_{\mathbf{v}_3} \end{array} \right]^T, \end{aligned} \tag{3.32}$$

with corresponding ordering of reference nodes $\hat{N}$ and $\hat{N}^c$. The 21×24 matrix E just selects out the items in N^c that are also in N, so that

$$E_{ij} = \begin{cases} 1, & \text{for } 1 \leq i = j \leq 19 \text{ or } (i,j) \in \{(20,21),(21,23)\} \\ 0, & \text{otherwise.} \end{cases}$$

The matrix V^c relating the push-forward of the extended nodes to the extended reference nodes is block diagonal and similar to our earlier examples. We use (3.14) to map the vertex gradient nodes as in the Hermite case. Mapping the three unique second derivatives by the chain rule requires the matrix:

$$\Theta = \begin{bmatrix} \left(\frac{\partial \hat{x}}{\partial x}\right)^2 & 2\frac{\partial \hat{x}}{\partial x}\frac{\partial \hat{y}}{\partial x} & \left(\frac{\partial \hat{y}}{\partial x}\right)^2 \\ \frac{\partial \hat{x}}{\partial y}\frac{\partial \hat{x}}{\partial x} & \frac{\partial \hat{x}}{\partial y}\frac{\partial \hat{y}}{\partial x} + \frac{\partial \hat{x}}{\partial x}\frac{\partial \hat{y}}{\partial y} & \frac{\partial \hat{y}}{\partial x}\frac{\partial \hat{y}}{\partial y} \\ \left(\frac{\partial \hat{x}}{\partial y}\right)^2 & 2\frac{\partial \hat{x}}{\partial y}\frac{\partial \hat{y}}{\partial y} & \left(\frac{\partial \hat{y}}{\partial y}\right)^2 \end{bmatrix} \tag{3.33}$$

The edge midpoint nodes transform by B just as in (3.26), so that the V^c is

$$V^c = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & J^{-T} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \Theta^{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & J^{-T} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \Theta^{-1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & J^{-T} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \Theta^{-1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & B^1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & B^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & B^3 \end{bmatrix}. \tag{3.34}$$

Constructing D, like for Morley, is slightly more delicate. The additional nodes acting on quintic polynomials – tangential derivatives at edge midpoints – must be written in terms of the remaining nodes. The first aspect of this involves a univariate interpolation-theoretic question. On the biunit interval $[-1, 1]$, we seek a rule of the form

$$f'(0) \approx a_1 f(-1) + a_2 f(1) + a_3 f'(-1) + a_4 f'(1) + a_5 f''(-1) + a_6 f''(1)$$

that is exact when f is a quintic polynomial. The coefficients may be determined to by writing a 6×6 linear system asserting correctness on the monomial basis. The answer, given in [16], is that

Proposition 3.8. *Any quintic polynomial p defined on $[-1, 1]$ satisfies*

$$p'(0) = \tfrac{15}{16}\left(p(1) - p(-1)\right) - \tfrac{7}{16}\left(p'(1) + p'(-1)\right) + \tfrac{1}{16}\left(p''(1) - p''(-1)\right). \tag{3.35}$$

This can be mapped to the interval $[-\frac{\ell}{2}, \frac{\ell}{2}]$ by a change of variables:

$$p'(0) = \tfrac{15}{8\ell}\left(p\left(\tfrac{\ell}{2}\right) - p\left(\tfrac{-\ell}{2}\right)\right) - \tfrac{7}{16}\left(p'\left(\tfrac{\ell}{2}\right) + p'\left(\tfrac{-\ell}{2}\right)\right) + \tfrac{\ell}{32}\left(p''\left(\tfrac{\ell}{2}\right) - p''\left(\tfrac{-\ell}{2}\right)\right). \tag{3.36}$$

Now, we can use this to compute the tangential derivative at an edge midpoint, expanding the tangential first and second derivatives in terms of the Cartesian derivatives. If $\mathbf{v}_a$ and $\mathbf{v}_b$ are the beginning and ending vertex of edge γ_i with midpoint $\mathbf{e}_i$ and length ℓ_i, we write the tangential derivative acting on quintics as

$$\begin{aligned} \pi\delta_{\mathbf{e}_i}^{\mathbf{t}_i} =& \tfrac{15}{8\ell_i}\left(\delta_{\mathbf{v}_b} - \delta_{\mathbf{v}_a}\right) - \tfrac{7}{16}\left(t_i^{\mathbf{x}}\left(\delta_{\mathbf{v}_b}^{\mathbf{x}} + \delta_{\mathbf{v}_a}^{\mathbf{x}}\right) + t_i^{\mathbf{y}}\left(\delta_{\mathbf{v}_b}^{\mathbf{y}} + \delta_{\mathbf{v}_a}^{\mathbf{y}}\right)\right) \\ &+ \tfrac{\ell_i}{32}\left((t_i^{\mathbf{x}})^2\left(\delta_{\mathbf{v}_b}^{\mathbf{xx}} - \delta_{\mathbf{v}_a}^{\mathbf{xx}}\right) + 2t_i^{\mathbf{x}}t_i^{\mathbf{y}}\left(\delta_{\mathbf{v}_b}^{\mathbf{xy}} - \delta_{\mathbf{v}_a}^{\mathbf{xy}}\right) + (t_i^{\mathbf{y}})^2\left(\delta_{\mathbf{v}_b}^{\mathbf{yy}} - \delta_{\mathbf{v}_a}^{\mathbf{yy}}\right)\right). \end{aligned} \tag{3.37}$$

For each edge γ_i, define the vector τ_i by

$$\tau_i = \left[(t_i^{\mathbf{x}})^2 \quad 2t_i^{\mathbf{x}}t_i^{\mathbf{y}} \quad (t_i^{\mathbf{y}})^2\right]^T.$$

The end result is that

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & I_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & I_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & I_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I_3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & I_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{-15}{8\ell_1} & \frac{7}{16}\mathbf{t}_1^T & \frac{-\ell}{32}\tau_1^T & \frac{15}{8\ell_1} & \frac{7}{16}\mathbf{t}_1^T & \frac{\ell}{32}\tau_1^T & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \frac{-15}{8\ell_2} & \frac{7}{16}\mathbf{t}_2^T & \frac{-\ell}{32}\tau_2^T & 0 & 0 & 0 & \frac{15}{8\ell_2} & \frac{7}{16}\mathbf{t}_2^T & \frac{\ell}{32}\tau_2^T & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{-15}{8\ell_3} & \frac{7}{16}\mathbf{t}_3^T & \frac{-\ell}{32}\tau_3^T & \frac{15}{8\ell_3} & \frac{7}{16}\mathbf{t}_3^T & \frac{\ell}{32}\tau_3^T & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3.38}$$

If this transformation is kept in factored form, D contains 57 nonzero entries and V^c contains 54 nonzero entries. E is just a Boolean matrix and its application requires copies. Application of M requires no more than 111 floating-point operations, besides the cost of forming the entries themselves. While this is about ten times the cost of the Hermite transformation, it is for about twice the number of basis functions and still well-amortized over the cost of integration loops. Additionally, one can multiply out the product EV^cD symbolically and find only 81 nonzero entries, which reduces the cost of multiplication accordingly.

3.4. **Generalizations**

3.4.1. *Non-affine mappings*

Non-affine geometric transformations, whether for simplicial or other element shapes, present no major complications to the theory. In this case, K and $\hat{K}$ are related by a non-affine map, and P is taken to be the image of $\hat{P}$ under pull-back

$$P = \left\{ F^*(\hat{p}) : \hat{p} \in \hat{P} \right\}, \tag{3.39}$$

although this space need not consist of polynomials for non-affine F. At any rate, one may define Hermite elements on curvilinear cells [12, 15]. In this case, the Jacobian matrix varies spatially so that each instance of J^T in (3.15) must be replaced by the particular value of J^T at each vertex.

3.4.2. *Generalized pullbacks*

Many vector-valued finite element spaces make use of pull-backs other than composition with affine maps. For example, the Raviart-Thomas and Nédélec elements use contravariant and covariant Piola maps, respectively. Because these preserve either normal or tangential components, one can put the nodal basis functions of a given element (K, P, N) and reference element $(\hat{K}, \hat{P}, \hat{N})$ into one-to-one correspondence by means of the Piola transform, a fact used heavily in [34]. It would be straightforward to give a generalization of affine equivalence to equivalence under an arbitrary pull-back F^*, with push-forward defined in terms of F^*. In this case, the major structure of §3.1 would be unchanged.

However, not all $H(\text{div})$ elements form equivalent families under the contravariant Piola transform. For example, Mardal, Tai, and Winther [30] give an element that can be paired with discontinuous polynomials to give uniform inf-sup stability on a scale of spaces between $H(\text{div})$ and $(H^1)^2$, although it is H^1-nonconforming. The degrees of freedom include constant and linear moments of normal components on edges, which are preserved under Piola mapping. However, the nodes also include the constant moments of the tangential component on edges, which are *not* preserved under Piola transform. One could push-forward both the normal and tangential constant moments, then express them as a linear combination of the normal and tangential moments on the reference cell in a manner like (3.15). One could see the Mardal–Tai–Winther element as satisfying a kind of "Piola-interpolation equivalence" and readily adapt the techniques for Hermite elements.

3.5. **A further note on computation**

We have commented on the added cost of multiplying the set of basis functions by M during local integration. It also also possible to apply the transformation in a different way that perhaps more fully leverages pre-existing computer routines. With this approach, M can also be included in local matrix assembly by means means of a congruence transform acting on the "wrong" element matrix as follows.

Given a finite element (K, P, N) with nodal basis $\Psi = \{\psi_i\}_{i=1}^{\nu}$ and bilinear form $a_K(\cdot,\cdot)$ over the domain K, we want to compute the matrix

$$A^K_{ij} = a_K(\psi_j, \psi_i). \tag{3.40}$$

Suppose that a computer routine existed for evaluating A^K via a reference mapping for affine-equivalent elements. Given the mapping $F : \hat{K} \to K$, this routine maps all integration to the reference domain $\hat{K}$ assuming that the integrand over K is just the affine pull-back of something on $\hat{K}$. Consider the following computation:

$$\begin{aligned} A^K_{ij} &= a_K(\psi_j, \psi_i) \\ &= a_K\left(\sum_{\ell_2=1}^{\nu} M_{j\ell_2} F^*(\hat{\psi}_{\ell_2}), \sum_{\ell_1=1}^{\nu} M_{i\ell_1} F^*(\hat{\psi}_{\ell_1})\right) \\ &= \sum_{\ell_1,\ell_2=1}^{\nu} M_{j\ell_2} M_{i\ell_1} a_K(F^*(\hat{\psi}_{\ell_2}), F^*(\hat{\psi}_{\ell_1})). \end{aligned} \tag{3.41}$$

Now, this is just expressed in terms of the affine pullback of reference-element integrands and so could use the hypothesized computer routine. We then have

$$A^K_{ij} = \sum_{\ell_1,\ell_2=1}^{\nu} M_{j\ell_2} M_{i\ell_1} a_{\hat{K}}(\hat{\psi}_{\ell_2}, \hat{\psi}_{\ell_1}) = \sum_{\ell_1,\ell_2=1}^{\nu} M_{j\ell_1} M_{i\ell_2} \hat{A}^K_{\ell_1\ell_2}, \tag{3.42}$$

or, more compactly,

$$A^K = M\tilde{A}^K M^T,$$

where $\tilde{A}^K$ is the matrix one would obtain by using the pull-back of the reference element nodal basis functions instead of the actual nodal basis for (K, P, N). Hence, rather than applying M invasively at each quadrature point, one may use existing code for local integration and pre- and post-multiply the resulting matrix by the basis transformation. In the case of Hermite, for example, applying M to a vector costs 12 operations, so applying M to all 10 columns of $\tilde{A}^K$ costs 120 operations, plus another 120 for the transpose. This adds 240 extra operations to the cost of building $\tilde{A}^K$, or just 2.4 extra FLOPs per entry of the matrix.

One may also apply this idea in a "matrix-free" context. Given a routine for applying $\tilde{A}^K$ to a vector, one may simply apply M^T to the input vector, apply $\tilde{A}^K$ to the result, and post-multiply by M. Hence, one has the cost of muliplying by $\tilde{A}^K$ plus the cost of applying M and its transpose to a single vector. In the case of Hermite, one has the cost of computing the "wrong" local matrix-vector product via an existing kernel plus 24 additional operations.

Finally, we comment on evaluating discrete functions over elements requiring such transforms. Discrete function evaluation is frequently required in matrix-free computation, nonlinear residual evaluation, and in bilinear form evaluation when a coefficient is expressed in a finite element space. Suppose one has on a local element K a function expressed by

$$u = \sum_{j=1}^{\nu} c_j \psi_j,$$

where $c \in \mathbb{R}^\nu$ is the vector of coefficients and $\{\psi_j\}$ is the nodal basis for (K, P, N). In terms of pulled-back reference basis functions, u is given by

$$u = \sum_{j=1}^{\nu} c_j \left(\sum_{k=1}^{\nu} M_{jk} F^*(\hat{\psi}_k) \right) = \sum_{j,k=1}^{\nu} M_{jk} c_j F^*(\hat{\psi}_k),$$

which can also be written as

$$u = \sum_{k=1}^{\nu} (M^T c)_k F^*(\hat{\psi}_k) = \sum_{k=1}^{\nu} (Vc)_k F^*(\hat{\psi}_k). \tag{3.43}$$

Just as one can build element matrices by means of the "wrong" basis functions and a patch-up operation, one can also evaluate functions by transforming the coefficients and then using the standard pullback of the reference basis functions. Such observations may make incorporating nonstandard element transformations into existing code more practical.

4. When the function space is not preserved under pull-back

The theory so far has been predicated on F^* providing an isomorphism between the reference and physical function spaces. In certain cases, however, this assumption fails. Now, we bring our theory to its fullest expression by introducing an *enriched* finite element whose function space is in fact preserved. This new element is then transformed by the already-developed techniques, and a nodal basis includes one for the original, unenriched element as a proper subset. In some sense, this is merely applying the same principle used for the nodes to the function space.

Our main motivation here is to transform the Bell element, a near-relative of the quintic Argyris element. In this case, one takes P to be the subspace of P_5 that has cubic normal derivatives on edges rather than the typical quartic values. This reduction of P by three dimensions is accompanied by removing the three edge normal derivatives at midpoints from N. In general, however, the pull-back $F^*(\hat{P})$ does not coincide with P. Instead of cubic normal derivatives on edges, $F^*(\hat{P})$ has reduced degree in some other direction corresponding to the image of the normal under affine mapping.

4.1. General theory: extending the finite element

Abstractly, one may view the Bell element or other spaces built by constraint as the intersection of the null spaces of a collection of functionals acting on some larger space as follows. Let (K, P, N) be a finite element. Suppose that $P \subset \tilde{P}$ and that $\{\lambda_i\}_{i=1}^{\kappa} \subset \left(C_b^k\right)'$ are linearly independent functionals that when acting on $\tilde{P}$ satisfy

$$P = \cap_{i=1}^{\kappa} \text{null}(\lambda_i). \tag{4.1}$$

The following result is not difficult to prove:

Proposition 4.1. *Let (K, P, N) be a finite element with $\cap_{i=1}^{\kappa}\text{null}(\lambda_i) = P \subset \tilde{P}$ as per* (4.1)*. Similarly, let $(\hat{K}, \hat{P}, \hat{N})$ be a reference element with $\cap_{i=1}^{\kappa}\text{null}(\hat{\lambda}_i) = \hat{P} \subset \tilde{\hat{P}}$. Suppose that $\tilde{P} = F^*(\tilde{\hat{P}})$. Then $P = F^*(\hat{P})$ iff*

$$\text{span}\{F_*(\lambda_i)\}_{i=1}^{\kappa} = \text{span}\{\hat{\lambda}_i\}_{i=1}^{\kappa}. \tag{4.2}$$

In the case of the Bell element, the span condition (4.2) fails and so that the function space is not preserved under affine mapping. Consequently, the theory of the previous section predicated on this preservation does not directly apply. Instead, we proceed by making the following observation.

Proposition 4.2. *Let (K, P, N) be a finite element with $P \subset \tilde{P}$ satisfying $P = \cap_{i=1}^{\kappa}\text{null}(\lambda_i)$ for linearly independent functionals $\{\lambda_i\}_{i=1}^{\kappa}$. Define*

$$\tilde{N} = \begin{bmatrix} N \\ L \end{bmatrix}$$

to include the nodes of N together with $L = \begin{bmatrix}\lambda_1 & \lambda_2 & \dots & \lambda_\kappa\end{bmatrix}^T$. Then $(K, \tilde{P}, \tilde{N})$ is a finite element.

Proof. Since we have a finite-dimensional function space, it remains to show that $\tilde{N}$ is linearly independent and hence spans $\tilde{P}'$. Consider a linear combination in $\tilde{P}'$

$$\sum_{i=1}^{\nu} c_i n_i + \sum_{i=1}^{\kappa} d_i \lambda_i = 0.$$

Apply this linear combination to any $p \in P$ to find

$$\sum_{i=1}^{\nu} c_i n_i(p) = 0$$

since $\lambda_i(p) = 0$ for $p \in P$. Because (K, P, N) is a finite element, the n_i are linearly independent in P' so $c_i = 0$ for $1 \le i \le \nu$. Applying the same linear combination to any $p \in \tilde{P}\backslash P$ then gives that $d_i = 0$ since the constraint functionals are also linearly independent. ■

In principle, this characterization can always be made. However, it will be most natural in cases, like the Bell element, where the function space P is already built by constraints.

Continuing with the development, it is easy, given a nodal basis for $(K, \tilde{P}, \tilde{N})$, to obtain one for (K, P, N).

Proposition 4.3. *Let (K, P, N), $\{\lambda_i\}_{i=1}^{\kappa}$, and $(K, \tilde{P}, \tilde{N})$ be as in Proposition 4.2. Order the nodes in $\tilde{N}$ by $\tilde{N} = \begin{bmatrix} N \\ L \end{bmatrix}$ with $L_i = \lambda_i$ for $1 \le i \le \kappa$. Let $\{\tilde{\psi}_i\}_{i=1}^{\nu+\kappa}$ be the nodal basis for $(K, \tilde{P}, \tilde{N})$. Then $\{\tilde{\psi}_i\}_{i=1}^{\nu}$ is the nodal basis for (K, P, N).*

Proof. Clearly, $n_i(\tilde{\psi}_j) = \delta_{ij}$ for $1 \le i, j \le \nu$ by the ordering of the nodes in $\tilde{N}$. Moreover, $\{\tilde{\psi}_i\}_{i=1}^{\nu} \subset P$ because $\lambda_i(\tilde{\psi}_j) = 0$ for each $1 \le i \le \kappa$. ■

4.2. **The Bell element**

We can obtain a nodal basis for the Bell element or others with similarly constrained function spaces by mapping the nodal basis for a slightly larger finite element and extracting a subset of the basis functions. Let (K, P, N) and $(\hat{K}, \hat{P}, \hat{N})$ be the Bell elements over K and reference cell $\hat{K}$.

Recall that the Legendre polynomial of degree n is orthogonal to polynomials of degree $n-1$ or less. Let $\mathcal{L}^n$ be the Legendre polynomial of degree n mapped from the biunit interval to edge γ_i of K.

Define a functional

$$\lambda_i(p) = \int_{\gamma_i} \mathcal{L}^4(s)\,(\mathbf{n}_i \cdot \nabla p)\,ds. \tag{4.3}$$

For any $p \in P_5(K)$, its normal derivative on edge i is cubic iff $\lambda_i(p) = 0$. The constraint functionals are given in $L = \begin{bmatrix}\lambda_1 & \lambda_2 & \lambda_3\end{bmatrix}^T$ and $\tilde{N} = \begin{bmatrix} N \\ L \end{bmatrix}$ as in Proposition 4.2. We define

$$\hat{\lambda}_i(p) = \int_{\hat{\gamma}_i} \mathcal{L}^4(s)\,(\hat{\mathbf{n}}_i \cdot \nabla p)\,ds \tag{4.4}$$

and hence $(\hat{K}, \hat{P}, \hat{N})$ as well as $\hat{L}$ and $\tilde{\hat{N}}$ in a similar way.

The constrained spaces are P and $\hat{P}$ – quintic polynomials with cubic normal derivatives on edges, while $\tilde{P}$ and $\tilde{\hat{P}}$ are the spaces of full quintic polynomials over K and $\hat{K}$, respectively. We must construct a nodal basis for $(\hat{K}, \tilde{\hat{P}}, \tilde{\hat{N}})$, map it to a nodal basis for $(K, \tilde{P}, \tilde{N})$ by the techniques in Section 3, and then take the subset of basis functions corresponding to the Bell basis.

This is accomplished by specifying a compatible nodal extension of $\tilde{N}$ and $\tilde{\hat{N}}$ by including the edge moments of *tangential* derivatives against $\mathcal{L}^4$ with those of $\tilde{N}$ and $\tilde{\hat{N}}$. We define

$$\begin{aligned} \lambda_i'(p) &= \int_{\gamma_i} \mathcal{L}^4(s)\,(\mathbf{t}_i \cdot \nabla p)\,ds, \\ \hat{\lambda}_i'(p) &= \int_{\hat{\gamma}_i} \mathcal{L}^4(s)\left(\hat{\mathbf{t}}_i \cdot \nabla p\right) ds. \end{aligned} \tag{4.5}$$

We must specify the E, V^c, and D matrices for this extended set of finite element nodes. We focus first on D, needing to compute each λ_i' in terms of the remaining functionals. As with Morley and Argyris, we begin with univariate results.

The following is readily confirmed, for example, by noting the right-hand side is a quintic polynomial and computing values and first and second derivatives at ± 1:

Proposition 4.4. *Let p be any quintic polynomial on $[-1, 1]$. Then*

$$\begin{aligned} 16p(x) = &-(x-1)^3\left(p''(-1)(x+1)^2 + p'(-1)(x+1)(3x+5) + p(-1)\left(3x^2+9x+8\right)\right) \\ &+ (x+1)^3\left(p''(1)(x-1)^2 - p'(1)(x-1)(3x-5) + \ (1)\left(3x^2-9x+8\right)\right). \end{aligned} \tag{4.6}$$

The formula (4.6) can be differentiated and then integrated against $\mathcal{L}^4$ to show that

$$\int_{-1}^{1} p'(x)\mathcal{L}^4(x)dx = \tfrac{1}{21}\left[\ (1) - p(-1) - p'(1) - p'(-1) + \tfrac{1}{3}\left(p''(1) - p''(-1)\right)\right]. \tag{4.7}$$

Then, this can be mapped to a general interval $[\frac{-\ell}{2}, \frac{\ell}{2}]$ by a simple change of variables:

$$\int_{-\frac{\ell}{2}}^{\frac{\ell}{2}} p'(x)\mathcal{L}^4(x)dx = \tfrac{1}{21}\left[p\left(\tfrac{\ell}{2}\right) - p\left(-\tfrac{\ell}{2}\right) - \tfrac{\ell}{2}\left(p'\left(\tfrac{\ell}{2}\right) + p'\left(-\tfrac{\ell}{2}\right)\right) + \tfrac{\ell^2}{12}\left(p''\left(\tfrac{\ell}{2}\right) - p''\left(-\tfrac{\ell}{2}\right)\right)\right]. \tag{4.8}$$

Now, we can use this to express the functionals λ_i' from (4.5) as linear combinations of the Bell nodes:

Proposition 4.5. *Let K be a triangle and $\mathbf{v}_a$ and $\mathbf{v}_b$ are the beginning and ending vertices of edge γ_i with length ℓ_i. Let p be any bivariate quintic polynomial over K and λ_i' defined in* (4.5)*. Then the restriction of λ_i' to bivariate quintic polynomials satisfies*

$$\pi\lambda_i' = \tfrac{1}{21}\left[\pi\delta_{\mathbf{v}_b} - \pi\delta_{\mathbf{v}_a} - \tfrac{\ell_i}{2}\left(\pi\delta_{\mathbf{v}_b}^{\mathbf{t}_i} + \pi\delta_{\mathbf{v}_b}^{\mathbf{t}_i}\right) + \tfrac{\ell_i^2}{12}\left(\pi\delta_{\mathbf{v}_b}^{\mathbf{t}_i\mathbf{t}_i} - \pi\delta_{\mathbf{v}_b}^{\mathbf{t}_i\mathbf{t}_i}\right)\right], \tag{4.9}$$

and hence

$$
\begin{aligned}
\pi\lambda_i' =& \tfrac{1}{21}\left[\pi\delta_{\mathbf{v}_b} - \pi\delta_{\mathbf{v}_a}\right] \\
&- \tfrac{\ell_i}{42}\left[t_i^{\mathbf{x}}\left(\pi\delta_{\mathbf{v}_b}^{\mathbf{x}} + \pi\delta_{\mathbf{v}_a}^{\mathbf{x}}\right) + t_i^{\mathbf{y}}\left(\pi\delta_{\mathbf{v}_b}^{\mathbf{y}} + \pi\delta_{\mathbf{v}_a}^{\mathbf{y}}\right)\right] \\
&+ \tfrac{\ell_i^2}{252}\left((t_i^{\mathbf{x}})^2\left(\pi\delta_{\mathbf{v}_b}^{\mathbf{xx}} - \pi\delta_{\mathbf{v}_a}^{\mathbf{xx}}\right) + 2t_i^{\mathbf{x}}t_i^{\mathbf{y}}\left(\pi\delta_{\mathbf{v}_b}^{\mathbf{xy}} - \pi\delta_{\mathbf{v}_a}^{\mathbf{xy}}\right) + (t_i^{\mathbf{y}})^2\left(\pi\delta_{\mathbf{v}_b}^{\mathbf{yy}} - \pi\delta_{\mathbf{v}_a}^{\mathbf{yy}}\right)\right).
\end{aligned}
\tag{4.10}
$$

Now, V^c is quite similar to that for the Argyris element. There is a slight difference in the handling the edge nodes, for we have an integral moment instead of a point value and must account for the edge length accordingly. By converting between normal/tangent and Cartesian coordinates via the matrix G_i and mapping to the reference element, we find that for any p,

$$
\begin{aligned}
\begin{bmatrix}\lambda_i(p)\\ \lambda_i'(p)\end{bmatrix} &= \int_{\gamma_i}\mathcal{L}^4(s)\left(G_i\nabla p\right)ds \\
&= \int_{\hat{\gamma}_i}\left|\tfrac{d\hat{s}}{ds}\right|\mathcal{L}^4(\hat{s})\left(G_iJ^T\hat{G}_i^T\hat{\nabla}^{\hat{\mathbf{n}}_i\hat{\mathbf{t}}_i}\hat{p}\right)d\hat{s} \\
&= \left|\tfrac{d\hat{s}}{ds}\right|G_iJ^T\hat{G}_i^T\begin{bmatrix}\hat{\lambda}_i(p)\\ \hat{\lambda}_i'(p)\end{bmatrix}.
\end{aligned}
\tag{4.11}
$$

This calculation shows that V^c for the Bell element is identical to (3.34) for Argyris, except with a geometric scaling of the B matrices.

The extraction matrix E for the extended Bell elements consisting of full quintics now is identical to that for Argyris. Then, when evaluating basis functions, one multiplies the affinely mapped set of basis values by V^T and then takes only the first 18 entries to obtain the local Bell basis.

4.3. **A remark on the Brezzi-Douglas-Fortin-Marini element**

In [24], we describe a two-part process for computing the triangular Brezzi-Douglas-Fortin-Marini (BDFM) element [19], an $H(\text{div})$ conforming finite element based on polynomials of degree k with normal components constrained to have degree $k-1$. This is a reduction of the Brezzi-Douglas-Marini element [11] somewhat as Bell is of Argyris. However, as both elements form Piola-equivalent families, the transformation techniques developed here are not needed.

Like the Bell element, one can define constraint functionals (integral moments of normal components against the degree k Legendre polynomial) for BDFM. In [24], we formed a basis for the intersection of the null spaces of these functionals by means of a singular value decomposition. A nodal basis for the BDFM space then followed by building and inverting a generalized Vandermonde matrix on the basis for this constrained space.

In light of Propositions 4.2 and 4.3, however, this process was rather inefficient. Instead, we could have merely extended the BDFM nodes by the constraint functionals, building and inverting a single Vandermonde-like matrix. If one takes the BDM edge degrees of freedom as moments of normal components against Legendre polynomials up to degree $(k-1)$ instead of pointwise normal values, then this technique shows that one can even build a basis for BDM that includes a basis for BDFM as a proper subset.

5. **Numerical results**

Incorporation of these techniques into high-level software tools such as Firedrake is the subject of ongoing investigation. In the meantime, we provide some basic examples written in Python, with sparse matrix assembly and solvers using petsc4py [14]. We begin with the unit square divided into a 4×4 mesh, then each square divided into two right triangles. To confirm that our techniques work

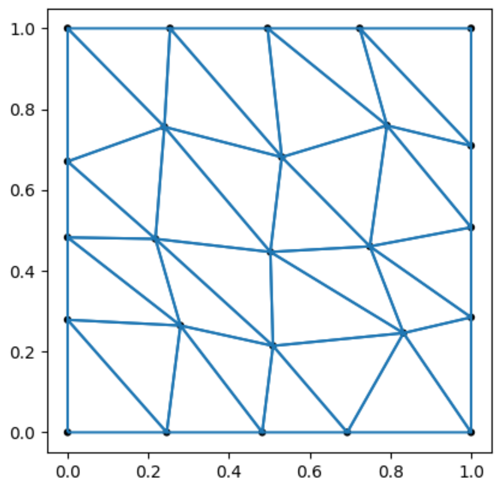


Figure 5.1. Initial mesh for our computations.

in irregular geometry, we then perturb the interior vertices of the triangulation, giving the mesh in Figure 5.1. All our results then follow on a sequence of uniform refinements of this mesh.

5.1. **Scaling degrees of freedom**

Before considering the accuracy of the L^2 projection, achieved via the global mass matrix, we comment on the conditioning of the mass and other matrices when both derivative and point value degrees of freedom appear. The Hermite element is illustrative of the situation.

On a cell of typical diameter h, consider a basis function corresponding to the point value at a given vertex. Since the vertex basis function has a size of $\mathcal{O}(1)$ on a triangle of size $\mathcal{O}(h^2)$, its L^2 norm should be $\mathcal{O}(h)$. Now, consider a basis function corresponding to a vertex derivative. Its *derivative* is now $\mathcal{O}(1)$ on the cell, so that the H^1 seminorm is $\mathcal{O}(h)$. Inverse inequalities suggest that the L^2 norm could then be as large as $\mathcal{O}(1)$. The different kinds of nodes introduce multiple scales of basis function sizes under transformation, which manifests in ill-conditioning. Where one expects a mass matrix to have an $\mathcal{O}(1)$ condition number, one now obtains an $\mathcal{O}(h^{-2})$ condition number. This is observed even on a unit square mesh, in Figure 5.2. All condition numbers are computed by converting the PETSc mass matrix to a dense matrix and using LAPACK via scipy [23].

However, there is a simple solution. For the Hermite element, one can scale the derivative degrees of freedom locally by an "effective h". All cells sharing a given vertex must agree on that h, which could be the average cell diameter among cells sharing a vertex. Scaling the nodes/basis functions (which amounts to multiplying V on the right by a diagonal matrix with 1's or h's) removes the scale separation among basis functions and leads again to an $\mathcal{O}(1)$ condition number for mass matrices, also seen in Figure 5.2. From here, we will assume that all degrees of freedom are appropriately scaled to give $\mathcal{O}(1)$ conditioning for the mass matrix.

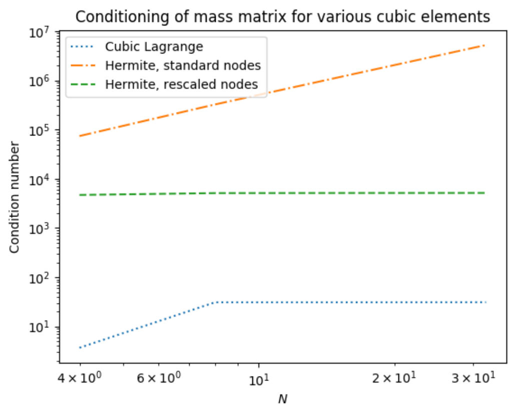


Figure 5.2. Condition numbers for cubic Lagrange and Hermite mass matrices on a sequence of uniform refinements of the mesh in Figure 5.1. This demonsrates an $\mathcal{O}(h^{-2})$ scaling when the "original" Hermite degrees of freedom are used, but $\mathcal{O}(1)$ condition number when the derivative degrees of freedom are scaled by h. Rescaling the Hermite nodes still gives a considerably larger condition number than for standard Lagrange elements.

5.2. Accuracy of L^2 projection

Now, we demonstrate that optimal-order accuracy is obtained by performing L^2 projection of smooth functions into the Lagrange, Hermite, Morley, Argyris, and Bell finite element spaces on uniform refinements of our initial mesh. Defining $u(x, y) = \sin(\pi x)\sin(2\pi y)$ on $[0, 1]^2$, we seek u_h such that

$$(u_h, v_h) = (u, v_h) \tag{5.1}$$

for each $v_h \in V_h$, where V_h is one of the finite element spaces. Predicted asymptotic convergence rates – third for Morley, fourth for Hermite and Lagrange, fifth for Bell, and sixth for Argyris, are observed in Figure 5.3 as we uniformly refine the initial mesh in Figure 5.1.

Note that the Hermite and Lagrange elements deliver the same order of approximation, but the Lagrange element delivers a slightly lower error. This is to be expected, as the space spanned by cubic Hermite triangles is a proper subset of that spanned by Lagrange.

5.3. The Laplace operator

As a simple second-order elliptic operator, we consider the Dirichlet problem for the Laplace operator on the unit square Ω:

$$-\Delta u = f, \tag{5.2}$$

equipped with homogeneous Dirichlet boundary conditions $u = 0$ on $\partial\Omega$.

We again consider tesselating Ω into uniform refinements of the mesh in Figure 5.1 and let V_h be one of the Lagrange, Hermite, Argyris, or Bell finite element spaces, all of which are H^1-conforming. The Morley element is not a suitable H^1 nonconforming element, so we do not use it here. We then seek $u_h \in V_h$ such that

$$(\nabla u_h, \nabla v_h) = (f, v_h) \tag{5.3}$$

for all $v_h \in V_h$.

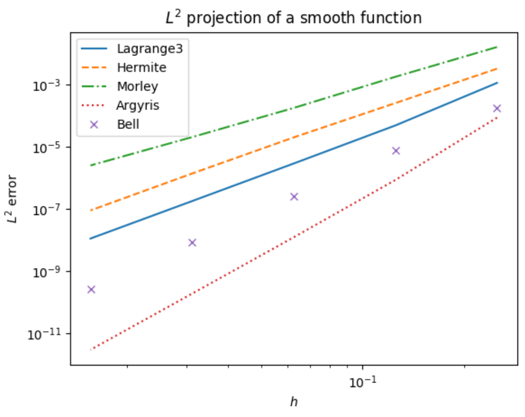


Figure 5.3. Accuracy of L^2 projection using cubic Lagrange, Hermite, Morley, Argyris, and Bell elements. All approach theoretically optimal rates.

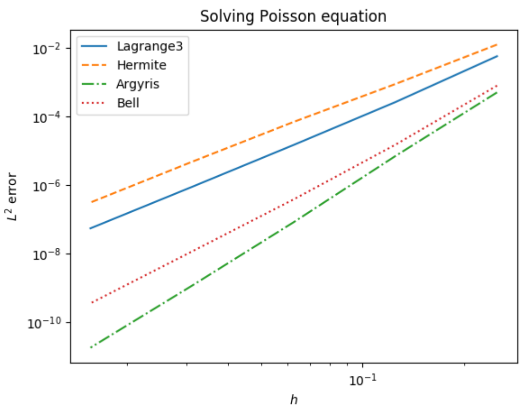


Figure 5.4. Convergence study of various elements for second-order elliptic equation (5.2). As the mesh is refined, all elements approach their predicted optimal rates of convergence.

Enforcing strong boundary conditions on elements with derivative degrees of freedom is delicate in general. However, with grid-aligned boundaries, it is less difficult. To force a function to be zero on a given boundary segment, we simply require the vertex values and all derivatives tangent to the edge vanish. This amounts to setting the x-derivatives on the top and bottom edges of the box and y-derivative on the left and right for Hermite, Argyris, and Bell elements. Dirichlet conditions for Lagrange are enforced in the standard way.

By the method of manufactured solutions, we select $f(x, y) = 8\pi^2 \sin(2\pi x)\sin(2\pi y)$ so that $u(x, y) = \sin(2\pi x)\sin(2\pi y)$. In Figure 5.4, we show the L^2 error in the computed solution for both element families. As the mesh is refined, both curves approach the expected order of convergence – fourth for Hermite and Lagrange, fifth for Bell, and sixth for Argyris. Again, the error for Lagrange is slightly smaller than for Hermite, albeit with more global degrees of freedom.

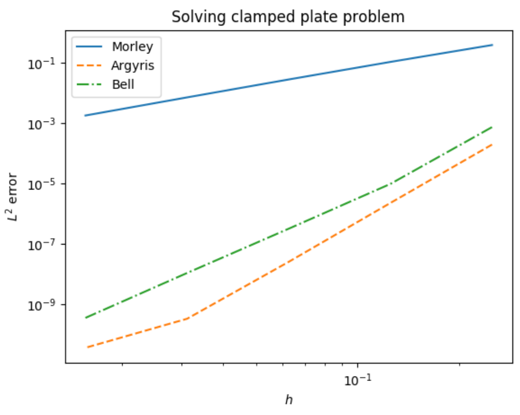


Figure 5.5. Convergence study of various elements for clamped plate biharmonic problem (5.5). As the mesh is refined, Bell, and Argyris elements converge in L^2 at fifth and sixth order, respectively. The nonconforming Morley element only converges at second order, which is known to be sharp.

5.4. **The clamped plate problem**

We now turn to a fourth-order problem for which the Argyris and Bell elements provide conforming H^2 discretizations and Morley a suitable nonconforming one. Following [10], we take the bilinear form defined on $H^2(\Omega)$ to be

$$a(u, v) = \int_\Omega \Delta u \Delta v - (1 - \nu)\left(2u_{xx}v_{yy} + 2u_{yy}v_{xx} - 4u_{xy}v_{xy}\right) dx, \tag{5.4}$$

where $0 < \nu < 1$ yields a coercive bilinear form for any closed subspace of H^2 that does not contain nontrivial linear polynomials. We fix $\nu = 0.5$.

Then, we consider the variational problem

$$a(u, v) = F(v) = \int_\Omega fv\, dx, \tag{5.5}$$

posed over suitable subspaces of H^2. It is known [10] that solutions of (5.5) that lie in $H^4(\Omega)$ satisfy the biharmonic equation $\Delta^2 u = f$ in an L^2 sense.

We consider the clamped plate problem, in which both the function value and outward normal derivative are set to vanish, which removes nontrivial linear polynomials from the space. Again, we use the method of manufactured solutions on the unit square to select $f(x, y)$ such that $u(x, y) = (x(1 - x)y(1 - y))^2$, which satifies clamped boundary conditions. We solve this problem with Argyris and Bell elements, and then also use the nonconforming Morley element in the bilinear form. Again, expected orders of convergence are observed in Figure 5.5.

6. **Conclusions**

Many users have wondered why FEniCS, Firedrake, and most other high-level finite element tools lack the full array of triangular elements, including Argyris and Hermite. One answer is that fundamental mathematical aspects of mapping such elements have remained relatively poorly understood. This work demonstrates the challenges involved with mapping such elements from a reference cell, but also proposes a general paradigm for overcoming those challenges by embedding the nodes into a larger set that transforms more cleanly and using interpolation techniques to relate the additional nodes back to original ones. In the future, we hope to incorporate these techniques in FInAT (`https://github.com/FInAT/FInAT`), a successor project to FIAT that produces abstract syntax for finite element evaluation rather than flat tables of numerical values. TSFC [21] already uses FInAT for its basis functions. If FInAT can provide rules for evaluating the matrix M in terms of local geometry on a per-finite element basis, then TSFC and other form compilers should be able to seamlessly (from the end-users' perspective) generate code for many new kinds of finite elements.

References

[1] M. Ainsworth, G. Andriamaro, and O. Davydov. Bernstein-Bézier finite elements of arbitrary order and optimal assembly procedures. *SIAM Journal on Scientific Computing*, 33(6):3087–3109, 2011.

[2] Martin S. Alnæs, Anders Logg, Kristian B. Ølgaard, Marie E. Rognes, and Garth N. Wells. Unified form language: A domain-specific language for weak formulations of partial differential equations. *ACM Transactions on Mathematical Software*, 40(2):9, 2014.

[3] J. H. Argyris, I. Fried, and D. W. Scharpf. The TUBA family of plate elements for the matrix displacement method. *Aeronautical Journal*, 72:701–709, 1968.

[4] W. Bangerth, R. Hartmann, and G. Kanschat. deal.II — a general purpose object oriented finite element library. *ACM Trans. Math. Softw.*, 33(4), 2007.

[5] K. Bell. A refined triangular plate bending finite element. *International Journal for Numerical Methods in Engineering*, 1(1):101–122, 1969.

[6] M. Bernadou. *Finite Element Methods for Thin Shell Problems.* Wiley, 1996.

[7] M. Bernadou and P.-L. George. *MODULEF: une bibliothèque modulaire d'éléments finis.* France. Inst. Nat. Rech. Inform. Autom., 1985.

[8] P.B. Bochev, H. Carter Edwards, Robert C. Kirby, Kara Peterson, and Denis Ridzal. Solving PDEs with Intrepid. *Scientific Programming*, 20(2):151–180, 2012.

[9] James H. Bramble and S. R. Hilbert. Bounds for a class of linear functionals with applications to Hermite interpolation. *Numerische Mathematik*, 16(4):362–369, 1971.

[10] Susanne C. Brenner and L. Ridgway Scott. *The mathematical theory of finite element methods*, volume 15 of *Texts in Applied Mathematics.* Springer, New York, third edition, 2008.

[11] Franco Brezzi, Jim Douglas Jr., and L. Donatella Marini. Two families of mixed finite elements for second order elliptic problems. *Numerische Mathematik*, 47(2):217–235, 1985.

[12] Philippe G. Ciarlet. *The finite element method for elliptic problems.* North-Holland, 1978.

[13] Philippe G. Ciarlet and P. A. Raviart. General Lagrange and Hermite interpolation in $\mathbb{R}^n$ with applications to finite element methods. *Archive for Rational Mechanics and Analysis*, 46(3):177–199, 1972.

[14] Lisandro D. Dalcin, Rodrigo R. Paz, Pablo A. Kler, and Alejandro Cosimo. Parallel distributed computing using Python. *Advances in Water Resources*, 34(9):1124–1139, 2011. New Computational Methods and Software Tools.

[15] Robert Dautray and Jacques-Louis Lions. *Mathematical Analysis and Numerical Methods for Science and Technology*, volume 4: Integral Equations and Numerical Methods. Springer-Verlag, 2012.

[16] Victor Domínguez and Francisco-Javier Sayas. Algorithm 884: A simple Matlab implementation of the Argyris element. *ACM Transactions on Mathematical Software (TOMS)*, 35(2):16, 2008.

[17] Todd F. Dupont and L. Ridgway Scott. Polynomial approximation of functions in Sobolev spaces. *Mathematics of Computation*, 34:441–463, 1980.

[18] G. Engel, K. Garikipati, T. J. R. Hughes, M. G. Larson, L. Mazzei, and R. L. Taylor. Continuous/discontinuous finite element approximations of fourth-order elliptic problems in structural and continuum mechanics with applications to thin beams and plates, and strain gradient elasticity. *Computer Methods in Applied Mechanics and Engineering*, 191(34):3669–3750, 2002.

[19] Michel Fortin and Franco Brezzi. *Mixed and hybrid finite element methods*. Springer, 1991.

[20] Jan S. Hesthaven and Timothy Warburton. *Nodal discontinuous Galerkin methods: Algorithms, analysis and applications*, volume 54 of *Springer Texts in Applied Mathematics*. Springer-Verlag, 2008.

[21] Miklós Homolya, Lawrence Mitchell, Fabio Luporini, and David A. Ham. TSFC: a structure-preserving form compiler. `https://arxiv.org/abs/1705.03667`, 2017.

[22] Stephen C. Jardin. A triangular finite element with first-derivative continuity applied to fusion MHD applications. *Journal of Computational Physics*, 200(1):133–152, 2004.

[23] Eric Jones, Travis Oliphant, and Pearu Peterson. Scipy: Open source scientific tools for python. `http://www.scipy.org/`, 2001.

[24] Robert C. Kirby. FIAT: A new paradigm for computing finite element basis functions. *ACM Trans. Math. Software*, 30:502–516, 2004.

[25] Robert C. Kirby. Fast simplicial finite element algorithms using Bernstein polynomials. *Numerische Mathematik*, 117(4):631–652, 2011.

[26] Robert C. Kirby and Anders Logg. A compiler for variational forms. *ACM Transactions on Mathematical Software*, 32(3), 2006.

[27] Anders Logg, Kent-Andre Mardal, Garth N. Wells, et al. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012.

[28] Kevin R. Long, Robert C. Kirby, and Bart van Bloemen Waanders. Unified embedded parallel finite element computations via software-based Fréchet differentiation. *SIAM Journal on Scientific Computing*, 32(6):3323–3351, 2010.

[29] Fabio Luporini, Ana Lucia Varbanescu, Florian Rathgeber, Gheorghe-Teodor Bercea, J. Ramanujam, David A. Ham, and Paul H. J. Kelly. COFFEE: an optimizing compiler for finite element local assembly. `https://arxiv.org/abs/1407.0904`, 2014.

[30] Kent-Andre Mardal, Xue-Cheng Tai, and Ragnar Winther. A robust finite element method for Darcy–Stokes flow. *SIAM Journal on Numerical Analysis*, 40(5):1605–1631, 2002.

[31] L. S. D. Morley. The constant-moment plate-bending element. *The Journal of Strain Analysis for Engineering Design*, 6(1):20–24, 1971.

[32] Christophe Prud'homme, Vincent Chabannes, Vincent Doyeux, Mourad Ismail, Abdoulaye Samake, and Gonçalo Pena. Feel++: A computational framework for Galerkin methods and advanced numerical methods. In *ESAIM: Proceedings*, volume 38, pages 429–455. EDP Sciences, 2012.

[33] 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. Firedrake: automating the finite element method by composing abstractions. *ACM Transactions on Mathematical Software (TOMS)*, 43(3):24, 2016.

[34] Marie E. Rognes, Robert C. Kirby, and Anders Logg. Efficient assembly of $H(\mathrm{div})$ and $H(\mathrm{curl})$ conforming finite elements. *SIAM Journal on Scientific Computing*, 31(6):4130–4151, 2009.

[35] Garth N. Wells and Nguyen Tien Dung. A C^0 discontinuous Galerkin formulation for Kirchhoff plates. *Computer Methods in Applied Mechanics and Engineering*, 196(35):3370–3380, 2007.