3D Partitioned Element Method: Shape-Function Formulation

Mark Rashid

23 April, 2015

1 Geometric Preliminaries

These notes cover the formulation of the shape functions on a single polyhedral finite element via the "partitioned element method" (PEM). A typical polyhedral finite element (henceforth, simply an "element") is shown in Figure 1. It is important to study this figure in order to understand the geometry-related notation and terminology that will be used throughout these notes.

Preparatory to the computation of the element's shape-function data, the element is first partitioned into *cells* ω_a, which are themselves polyhedra. (Throughout, lower-case Latin subscripts from the beginning of the alphabet, e.g. $a,\ b,\ c$, are used to index topes such as polyhedral cells, polygonal facets, etc.) The boundaries of the cells consist of polygonal facets σ_a, whose boundaries are, in turn, composed of line segments γ_a. Finally, each segment is bounded by a pair of vertices v_a. Some representative facets (σ's), segments (γ's), and vertices (v's) are labeled in Figure 1. Only the topes that lie on the boundary of the element can be seen in the figure; of course there are others in the interior, associated with the partition of the element into cells. The important point at this stage is that, geometrically, the element consists of inventories of cells $\{\omega_a\}$, facets $\{\sigma_a\}$, segments $\{\gamma_a\}$, and vertices $\{v_a\}$ that are logically connected via nested boundary definitions. I.e. each cell is defined by the collection of facets that define its boundary, each facet is defined by the set of segments on its boundary, and each segment is defined by the pair of vertices that constitute its endpoints.

The element's topes will be associated into other sets as well, for purposes that will be discussed shortly. By way of preview, it will prove useful, for example, to organize the facets into disjoint sets that define *element faces*. We shall also have a need for explicit "reverse connectivity" data, such as the cells that contain a given facet, and the facets that contain a given segment. These and other set associations, along with the notation used to describe them, are explained subsequently.

Vertices are indicated in Figure 1 with black dots. Some vertices have red circles around them; these circled vertices are also *nodes*, which are labeled p_a. A node is a point with which there is associated both a shape function, and a value of the function that is being interpolated. An *interpolant* on the element is simply a linear combination of nodal values times their respective shape functions. Not all vertices are nodes; indeed, nodes only occur

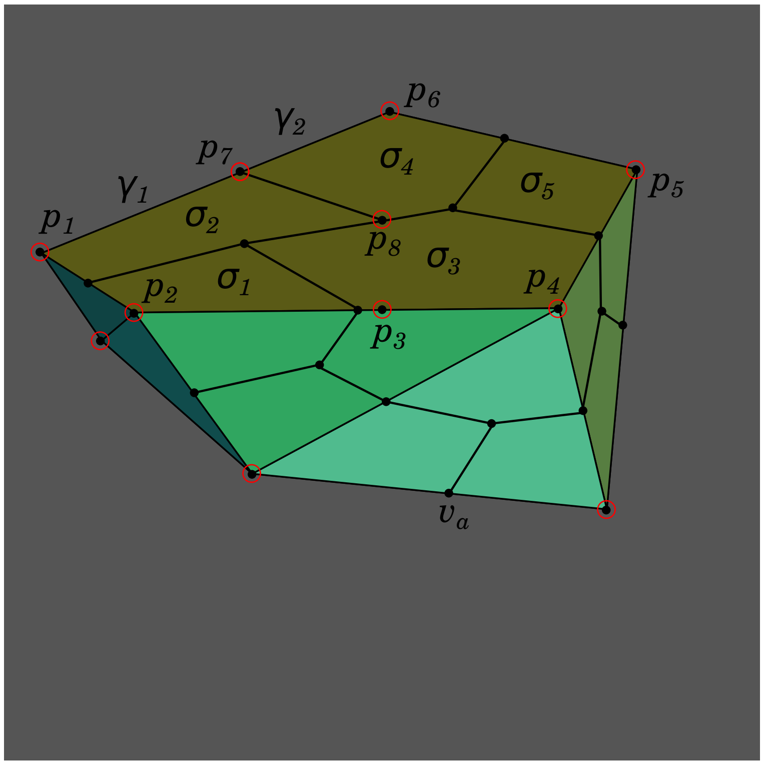

Figure 1: Element geometry.

on the boundary of the element. Even among the boundary vertices, some are nodes while others are not.

Before the element is subdivided into cells, the element geometry is defined by what will be called **source facets**. The *facets* that lie on the element's boundary are the result of subdivision of these source facets, pursuant to the partitioning of the element into cells. (Throughout, the modifier **source** will indicate that the tope in question belongs to the **source mesh** – i.e. the finite element mesh itself, prior to partitioning of any of its elements into cells.)

An element's source facets define the boundary of the element. The source facets are organized into **source faces**, as follows. A single source face consists of a contiguous collection of source facets that are shared with exactly one other element in the mesh, or with void. Note that "contiguous collection" here means that each source facet in the collection shares at least one source segment with another source facet in the collection. Two source facets that touch only at a source vertex would not be considered contiguous.

Next, **source edges** are defined as collections of **source segments**, as follows. Each source edge consists of a contiguous collection of source segments that belong to two source-face boundaries. Source edges usually consist of single, open chains of source segments that are linked together head-to-tail. It will also usually be the case that the endpoints of a source edge are nodes. Nodes may occur at other vertices along the source edge as well.

The association of source facets into source faces, and source segments into source edges, is inherited by the element's facets and segments after the element is partitioned into cells.

In other words, the facets that come from any of the source facets that belong to a single source face are collected together into a **face**. Similarly, the segments that come from any of the source segments that belong to a single source edge are collected together into an **edge**. So, **faces** and **edges** consist of sets of **facets** and **segments**, respectively, of the post-subdivision element, with the set memberships derived from the source-face and source-edge memberships.

2 Data Constructs

At the heart the PEM shape-function formulation is the idea that *some of the segments and facets of the element, and all of the cells, support their own local interpolants.* These local interpolants take the form of complete, low-order polynomials in the element-global Cartesian coordinates. ("Element-global Cartesian coordinates" refers to a fixed Cartesian frame that is aligned with the global frame for the whole problem, but with origin translated to the first node of the element.) In the "default" element formulation, these "tope-specific," or local, interpolants are simply linear polynomials, and thus require 4 coefficients to specify on each tope. However, the development here will accommodate quadratic interpolants as well (10 coefficients each). Local interpolants of even higher order are possible, but are unlikely to exhibit favorable cost-benefit economics. It is important to appreciate that these local interpolants are not strictly "compatible," e.g. the local interpolants on the collection of cells does not fit together into a *continuous* function on the element. However, the local interpolants *will* be chosen so that the discontinuities at inter-tope interfaces is, in an appropriate sense, minimized. These ideas will be elaborated and made precise in the next section.

A further central idea in the PEM shape-function formulation is that the local interpolants described above are *linear* functions of some or all of the element's nodal values. Specifically, the polynomial coefficients of each cell's local interpolant are linearly related to *all* nodal values, while the dependencies for facet and segment local interpolants are confined to particular subsets of nodes. *Which* nodal values influence which segment- and facet-interpolants is intimately related to the concepts of *element faces* and *element edges*, which are explained next.

Clearly not all of the element's facets and segments belong to a face or an edge: only the element-boundary facets, and only *some* of the element-boundary segments, have this status. However, importantly, only the facets and segments that *do* belong to a face or, respectively, to an edge support their own local interpolants. Further, a main purpose of organizing facets and segments into faces and edges is to delineate the sets of nodes on which each local interpolant depends. Specifically:

- If a segment belongs to a particular edge, then the segment has a local interpolant, and the coefficients of this interpolant depend on the nodal values at the nodes contained by the edge.

- If a facet belongs to a particular face, then the facet has a local interpolant, and the coefficients of this interpolant depend on the nodal values at the nodes contained by the face.

- All cells belong to the element, of course; this is the logical extension of facet/face and segment/edge associations. Each cell has a local interpolant, the coefficients of which depend on all nodal values of the element.

In Figure 1, segments γ_1 and γ_2 together make up a single edge. Nodes p_1, p_6, and p_7 are, correspondingly, assigned to γ_1 and γ_2. Turning to the facets, σ_1, σ_2, σ_3, σ_4, and σ_5 make a single element face. Each of these 5 facets is assigned nodes p_{1-8}. Geometrically, element faces need not be planar. However, if a face's bounding edges lie in a plane, but the facets of the face do not, then there will be one or more nodes at interior vertices of the face. In Figure 1, node p_8 is such a face-interior node. As mentioned previously, for the present purposes, node definitions, as well as the association of facets into faces and segments into edges, are assumed given.

The polynomial-coefficient/nodal-value dependencies referred to above are linear. The matrices that define these dependencies are the main objectives of the calculations described in these notes. The notation that will be used for these main results is as follows.

- $\mathbf{P}_a$: The $4 \times n$ (linear local interpolant) or $10 \times n$ (quadratic) matrix that transfers segment γ_a's n nodal values into the polynomial coefficients of its local interpolant.

- $\mathbf{Q}_a$: Similarly, for facet σ_a.

- $\mathbf{R}_a$: The transfer matrix for cell ω_a.

We will also need, for *all* segments and facets, a mapping from a local list of polynomial coefficients, to the coefficients of the corresponding polynomial in the element-global Cartesian coordinates. Consider first the case of a segment. Any complete quadratic polynomial g in the element-global Cartesian coordinates has the form

$$g = C_1 + C_2x + C_3y + C_4z + C_5x^2 + C_6xy + C_7xz + C_8y^2 + C_9yz + C_{10}z^2. \tag{1}$$

When restricted to a line segment, however, this g can be expressed by

$$g = c_0 + c_1\xi \ + \ c_2\xi^2, \tag{2}$$

where ξ is a segment coordinate that measures, e.g., distance along the segment from end-point 1. We will need, for each segment γ_a, the matrix $\mathbf{L}_a$ that relates the vector $\mathbf{C}$ of global polynomial coefficients to the corresponding vector $\mathbf{c}$ of segment polynomial coefficients, via

$$\mathbf{C} = \mathbf{L}_a\mathbf{c}. \tag{3}$$

The $\mathbf{L}_a$ matrices have dimension 10×3 for a mapping of quadratic polynomials. The corresponding matrices for the facets will be denoted $\mathbf{M}_a$, each of which has dimension 10×6 for a mapping of quadratic polynomials. For the present purposes, all of these matrices are to be regarded as given. In any case, they are easy to derive, and depend only on the direction of the line (for segments) or normal to the plane (for facets) on which the subject tope lies.

We are now in a position to articulate the data that should be stored for each segment, facet, and cell.

- For each **segment** that lies on the boundary of the element:

   1. Indices of the two facets that share the segment, followed by the index of the edge on which the segment falls, if the segment belongs to an edge (zero otherwise).
   2. Nodal-value-to-polynomial-coefficient matrix $\mathbf{P}_a$. This is only needed for segments that lie on an edge. It should be an allocatable array, and left unallocated for segments that lie on the interiors of faces. It is computed based on the developments given in the next section.
   3. Polynomial-coefficient map $\mathbf{L}_a$.
   4. Vector $\mathbf{m}_a$ of monomial integrals on the segment. These are integrals of monomials in the segment coordinate ξ (see eqn. 2 above). Integrals through a certain monomial order will be needed; the specific order is discussed later.

  All of this data should be pre-calculated and stored, with the exception of $\mathbf{P}_a$, whose calculation is explained in the next section. A derived type containing the above-described components should be defined, with one object of this type defined for each segment that lies on the boundary of the element.

- For each **facet** in the element's inventory of facets:

   1. Indices of the two cells that share the facet, followed by the index of the face on which the facet falls, if the facet belongs to a face (zero otherwise). If the facet belongs to a face, then the second (cell) index should be zero.
   2. Nodal-value-to-polynomial-coefficient matrix $\mathbf{Q}_a$. This is only needed for facets that lie on a face. It should be an allocatable array, and left unallocated for facets that lie in the element's interior. Its computation is the subject of the next section.
   3. Polynomial-coefficient map $\mathbf{M}_a$.
   4. Vector $\mathbf{m}_a$ of monomial integrals on the segment. These are integrals of monomials in the facet coordinates ξ, η through a certain order, which is discussed later.

  A derived type should be defined with these components, with one object if this type defined for each facet in the element's inventory of facets.

- For each **cell** in the element's inventory of cells:

   1. Nodal-value-to-polynomial-coefficient matrix $\mathbf{R}_a$. This is computed as explained in the next section.
   2. Vector $\mathbf{m}_a$ of monomial integrals on the facet. These are integrals of monomials in the element-global Cartesian coordinates, which are parallel to the mesh-global Cartesian coordinates, but offset by the first node of the element.

In addition to this tope-centric data, we will also need some data that pertains to edges and faces, as follows:

- $\mathcal{P}_a$, $\mathcal{S}_a$, $\bar{\mathbf{P}}_a$: The first of these is the set of nodes associated with edge a (and, therefore, the set of nodes associated with each segment that belongs to edge a). The second is the set of segments that compose edge a. The final item is a real matrix that gives the element-global polynomial coefficients for the linear (4 coefficients) or quadratic (10 coefficients) polynomial that is the least-squares best fit of the edge's nodal data. I.e., $\bar{\mathbf{P}}_a \times$ (vector of nodal values) = (polynomial coefficients).

- $\mathcal{Q}_a$, $\mathcal{F}_a$, $\bar{\mathbf{Q}}_a$: The first of these is the set of nodes associated with face a (and, therefore, the set of nodes associated with each facet that belongs to face a). The second is the set of facets that make up face a. The third item is a real matrix that gives the element-global polynomial coefficients for the linear (4 coefficients) or quadratic (10 coefficients) polynomial that is the least-squares best fit of the face's nodal data. I.e., $\bar{\mathbf{Q}}_a \times$ (vector of nodal values) = (polynomial coefficients).

The "set" components above could consist of allocatable arrays of integers, or of linked lists that point to the actual objects themselves. It is not clear at this stage which would be better. The best-fit matrices for edges and faces are needed for subtle reasons, which are explained in the next section. In any case, these matrices are easily formed once faces and edges have been defined and nodes assigned to them. As such, the above edge- and face-specific data should be regarded as known ahead of the calculations described in the next section.

3 Calculation

For the present purposes, we shall assume that the element's partitioned geometry is known; i.e. we have in hand the complete inventory of vertices, segments, facets, and cells. We shall also assume that all of the above-described data objects are fully pre-defined, with the exception of the $\mathbf{P}_a$, $\mathbf{Q}_a$, and $\mathbf{R}_a$ matrices. Computation of these tope-specific components is the subject of this section.

The basic pattern of the shape-function calculation is as follows. First, we consider each edge in turn. The $\mathbf{P}_a$ matrices for the segments that make up a given edge are the result of a minimization problem on the edge. After treating all edges, we will have all $\mathbf{P}_a$ matrices for the segments for which we need these matrices – i.e. all segments that belong to an edge. Next, we consider the faces, one after another. The $\mathbf{Q}_a$ matrices for the facets of a given face emerge from a minimization problem on the face. The $\mathbf{P}_a$ matrices for the segments that bound the edge also appear in this minimization problem. Solving these minimization problems gives us a $\mathbf{Q}_a$ for each facet for which we need one, namely, the facets that belong to faces. Finally, we come to the calculation of the $\mathbf{R}_a$ matrices for the element's cells, which are the outcomes of a minimization problem on the element. This minimization problem involves the $\mathbf{Q}_a$ matrices from the element's boundary facets. The calculation concludes with any cell-averaging that must occur in order to form the cell averages of the shape-function gradients.

3.1 edges and their segments

And now the details, taking the edge case first. Figure 2 shows a typical edge that contains 3 nodes, 5 vertices, and 4 segments. The nodal values φ_a of the interpolant are assumed given,

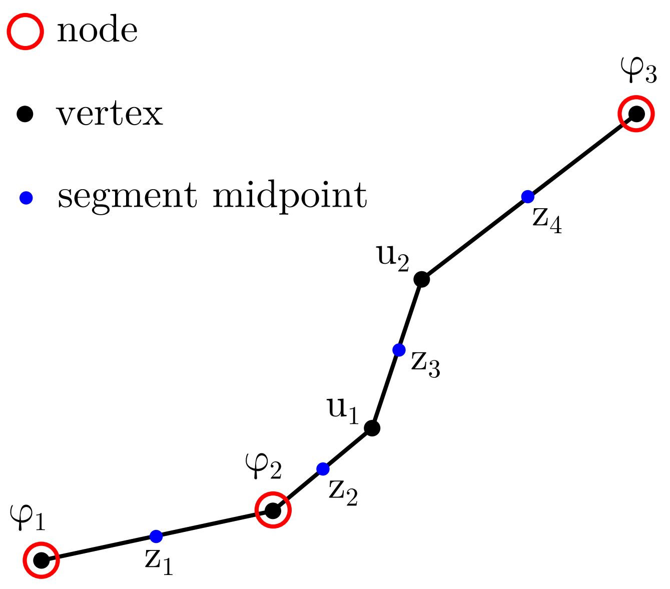

Figure 2: An element edge consisting of 4 segments.

while the non-node-vertex values u_a and the mid-segment values z_a are to be determined based on minimization of a suitable quadratic function. The mid-segment values are only present when quadratic local interpolants are used on the segments. If linear local interpolants are being used, then no mid-segment values appear, and the quadratic function to be minimized reduces to a function of the u_a.

The quadratic objective function is taken to be a combination of an unweighted sum of the squares of the jumps in the derivatives at the interior vertices, and the mean-square slope of the interpolant over the edge. Letting $\Delta_a,\ 2,\dots,n-1$ be the jump in tangential derivative at vertex a, this objective function is

$$(1-\beta_1)\sum_{a=2}^{n-1}\frac{1}{2}\Big[|\gamma_{a-1}|+|\gamma_a|\Big]\,\Delta_a^2\;+\;\beta_1\sum_a\int_{\gamma_a}(\varphi')^2 d\xi, \tag{4}$$

where n is the total number of vertices in the edge, and φ' is the derivative of the segment-local interpolant along segment γ_a. $\beta_1 \in (0,1)$ is a numerical parameter that controls the relative importance of the slope jumps at vertices vs. the slopes on the interiors of the segments. The above function constitutes a measure of smoothness of the interpolant on the edge, which is taken to be *continuous* and either piecewise linear or piecewise quadratic. Consequently, the edge interpolant, and therefore the vertex slope-jumps and segment-interior slopes, are completely determined by the node, vertex, and mid-segment values. Minimizing (4) with respect to the unknowns u_a and z_a leads to the linear relation

$$\mathbf{u} = \mathbf{D}\mathbf{p}, \tag{5}$$

where $\mathbf{u}$ is a vector containing all u_a and z_a unknowns, and $\mathbf{p}$ is a vector of nodal values φ_a.

With the matrix $\mathbf{D}$ in hand for the edge, it is a straightforward matter to form the segment-coordinate polynomial coefficients on each segment of the edge. These coefficients, of course, will be linear functions of the edge-nodal-value vector $\mathbf{p}$. From there, the polynomial-coefficient map $\mathbf{L}_a$ on each segment is used to form the final result $\mathbf{P}_a$, which maps the vector $\mathbf{p}$ to the polynomial coefficients of the segment-local interpolant.

A subtle but serious problem arises whenever the edge is not straight. This problem, and its fix, requires a bit of effort to explain and to understand. The essence of the issue is the requirement that the eventual element interpolant exhibit *linear completeness*. Linear completeness calls for an element interpolant that exactly reproduces a linear function, whenever the element's nodal values are set consistent with the linear function. In order for the final shape functions to exhibit linear completeness, it is necessary that the facet-local interpolants and, in their turn, the segment-local interpolants do so as well. In other words, e.g., whenever the vector $\mathbf{p}$ of nodal values for a particular edge are consistent with the linear function $b_0 + b_1x + b_2y + b_3z$, then the polynomial coefficients for each segment of the edge, namely $\mathbf{P}_a\mathbf{p}$, should be consistent with this same linear function. It is tempting to interpret this requirement as implying simply that $\mathbf{P}_a\mathbf{p}$ should be the vector consisting of the original linear-function coefficients b_0, b_1, b_2, b_3. However, this is not possible in general, as can be seen by taking a simple example. Suppose the edge is straight, and parallel to the element-global x-axis. In this case, the nodal values on the edge are insensitive to b_2 and b_3, so there is no way these two coefficients can be determined on the basis of the edge's nodal values.

This insensitivity in the case of a *straight* edge is not "the problem," though it does hint at it. (After all, linear completeness requires only that the interpolant on each segment of an edge be exactly consistent with any linear function which is used to set the edge's nodal values. In the event that multiple global linear functions produce the same pattern of nodal values *and* a consistent linear variation on the edge's segments, then linear completeness is still satisfied – the critical thing is the *consistency* between the nodal values and the interpolant for *all* globally linear functions.) A further example will provide more insight. Consider now an edge in the form of a "vee" shape, with two nodes, one at each endpoint. Suppose that the edge lies in the x, y plane, and further that the two nodes lie on the x-axis. The two nodal values establish the coefficients b_0 and b_1, but not b_2 and b_3. But, an interpolant on the edge that takes its values from the linear function $b_0 + b_1x + b_2y + b_3z$ would be sensitive to b_0, b_1, *and* b_2. Therefore, it is not possible for linear completeness to hold in this situation, because the edge interpolant cannot "know about" b_2 if it must depend only on the edge's nodal values.

The obvious solution here would be to place a third node at the apex of the vee. By doing so, we would bring into alignment the coefficients of the global linear function that are determinable by the nodal values – which are now b_0, b_1, and b_2 (assuming the edge lies in the x, y plane) – and the coefficients that influence the variation on the edge. (In fact, this alignment is necessary in order to achieve linear completeness, and constitutes the main factor in decisions about the placement of nodes at interior vertices of both edges and faces.) However, this "polynomial-coefficient alignment" does not, by itself, deliver linear completeness, for the simple reason that it is not clear how to devise an objective function to replace (4) so that the correct linear interpolant on each segment is produced by minimizing the functional, with the nodal values set consistent with a globally linear function. Certainly the simple functional (4) does not achieve this end, even though it produces pleasingly smooth interpolants on edges.

This is where the least-squares best-fit matrix $\bar{\mathbf{P}}$ comes to the rescue (here the subscript $(\ )_a$ is omitted, it being understood that we are dealing with a single edge at the moment). Consider "residual" nodal values defined by

$$\hat{\varphi}_a = \varphi_a - \sum_{i=1}^{4} \sum_{b} \bar{P}_{ib} m_i(p_a) \varphi_b, \tag{6}$$

where $m_i(p_a)$ is monomial number i evaluated at node p_a of the edge. The nodal values $\hat{\varphi}_a$ are what remain after subtracting from the original nodal values φ_a the best-fit linear function evaluated at each node. (If the φ_a are set consistent with a globally linear function, obviously $\hat{\varphi}_a = 0$.) Eqn. (6) can be used to define a matrix $\hat{\mathbf{P}}$ such that

$$\hat{\varphi}_a = \sum_{b} \hat{P}_{ab}\, \varphi_b. \tag{7}$$

The basic idea is to use the matrix $\mathbf{P}_a$, computed as described above, to map the *residual* nodal values $\hat{\varphi}_a$ to polynomial coefficients for segment γ_a, and then add back in the linear-polynomial coefficients produced by the edge-wide best-fit mapping $\bar{\mathbf{P}}$. This amounts to using the minimization of (4) to determine only that part of the edge interpolant that departs from its linear part, where the linear part is determined by a linear, least-squares best fit of the nodal values. Linear consistency on the edge is thereby guaranteed. Proceeding in this manner, the final expression for the nodal-value-to-polynomial-coefficient map for a segment is

$$\bar{P}_{ia} + \sum_{b} P_{ib} \hat{P}_{ba}. \tag{8}$$

In this expression, P_{ib} is specific to each segment of the subject edge, whereas $\bar{P}_{ia}$ and $\hat{P}_{ba}$ are both associated with the edge itself.

3.2 faces and their facets

Next, we turn to the faces and their constituent facets. The calculation follows the same general pattern as did the edge/segment procedure. Specifically, we take each face in turn and solve a minimization problem on the face, the solution to which is a set of mappings $\mathbf{Q}_a$ that take the face's nodal values into polynomial coefficients of the local interpolant on facet σ_a. Exactly the same linear-consistency issue comes up in the case of non-planar faces as did with non-straight edges, with exactly the same solution. Accordingly, the minimization problem is applied to the residual nodal values that remain after subtracting best-fit nodal values on the face, thus ensuring linear consistency, even when the face is non-planar.

The main issue to be addressed is the formulation of the minimization problem on a face. The objective function will look slightly different than it did for edges, because with faces, there is no prospect for achieving a continuous interpolant over the face. Accordingly, the objective function cannot be framed in terms of vertex values; instead, it will be cast as a function of the facet-interpolant polynomial coefficients directly. The proposed objective function takes the following form:

$$(1-\beta_2) \sum_{a \in \mathcal{L}} |\gamma_a| \int_{\gamma_a} \Delta^2 \, d\xi \; + \; \beta_2 \sum_{a \in \mathcal{L} \cup \mathcal{B}} \frac{1}{|\gamma_a|} \int_{\gamma_a} \varepsilon^2 \, d\xi \; + \; \sum_{a \in \mathcal{I}} \sum_{b \in \mathcal{J}_a} \varepsilon_{ab}^2. \tag{9}$$

Here, $\mathcal{L}$ is the set of segments on the interior of the face, and $\mathcal{B}$ is the set of segments on the boundary of the face (these segments necessarily belong to an edge). Δ is the jump in in-plane normal derivative across a facet/facet interface segment, taken normal to the segment; and ε is the jump in interpolant value across a segment. $\mathcal{I}$ is the set of nodes in the interior of the face (if any), and $\mathcal{J}_a$ is the set of facets that share interior-node a. Finally, ε_{ab} is the difference between nodal value φ_a and the facet-local interpolant on facet σ_b, evaluated at node a.

The objective function (9) is designed to measure both continuity and smoothness of the interpolant on the face. The first term of (9) measures the jumps in in-plane normal derivative as we cross from one facet to the next. The face-interior ($\mathcal{L}$) contributions to the second term quantifies the discontinuity at the interfaces between facets, while the face-boundary ($\mathcal{B}$) contributions capture the compatibility of the face interpolant with the adjacent edge interpolant on the face's boundary. The third term measures consistency of the face interpolant with the nodal values at any nodes that lie on the interior of the face. The numerical parameter $\beta_2 \in (0,1)$ adjusts the relative influence of the normal-derivative jumps vs. the interpolant discontinuities.

The objective function (9) is a quadratic function of the facet-local polynomial coefficients. For purposes of minimizing the objective function, the facet-local interpolants are expressed as linear or quadratic polynomials in each facet's in-plane coordinates ξ, η (see eqns. 1 and 2). Minimizing the objective function with respect to the facet-local polynomial coefficients produces matrices $\mathbf{E}_a$, such that $\mathbf{E}_a\mathbf{p}$ gives a column vector of facet-local polynomial coefficients for facet a of the face. Pre-multiplying by facet a's polynomial-coefficient map $\mathbf{M}_a$ then produces the "final" result $\mathbf{Q}_a = \mathbf{M}_a\mathbf{E}_a$, which constitutes the transfer matrix from nodal values to element-global polynomial coefficients of the interpolant on facet a of the face.

"Final" is in quotes here because we still need to subtract (linear or quadratic) best-fit values from the face's nodal values, and then apply the minimization-problem-derived transfer matrices $\mathbf{Q}_a$ only to the residual nodal values, just as we did in the edge case. With reference to eqns. (6, 7), we define a "residualizer" matrix $\hat{\mathbf{Q}}$ associated with the face as

$$\hat{Q}_{ab} = \delta_{ab} - \sum_{i=1}^{4} m_i(p_a)\bar{Q}_{ib}, \tag{10}$$

where again $m_i(p_a)$ is element-global monomial number i evaluated at face-node p_a, and $\bar{Q}_{ib}$ is the best-fit matrix for the face. The sum $i = 1,\ 4$ contemplates a linear best-fit; the quadratic case corresponds to $i = 1,\ 10$. The truly final nodal-value-to-polynomial-coefficient map for a particular facet of the face is now given by

$$\bar{Q}_{ia} + \sum_b Q_{ib}\hat{Q}_{ba}. \tag{11}$$

This matrix maps a column vector of nodal values associated with a facet's face, to a column vector of element-global polynomial coefficients for the local interpolant on a facet. There is one such matrix for each facet of the face, although the facet index has been dropped in (11), it being understood that we are taking about a single facet in this expression. In (11), Q_{ib} is specific to each facet of the subject face, whereas $\bar{Q}_{ia}$ and $\hat{Q}_{ba}$ are both associated with the face itself. Eqn. (11) represents the final result for the facets, which is to be used in place of **Q**.

3.3 the element and its cells

Finally we come to the calculation of the cell-local interpolants for the element. The main ideas are very similar to those of the face and edge calculations: a minimization problem is solved on the whole element, the solution to which is cell-local interpolants on each of the element's constituent cells. There are some differences in the details, however. Specifically, at the element/cell level, no least-squares best-fit projection of the nodal values is performed as a means of guaranteeing linear completeness. The reason for this is that minimization of the relevant objective function (introduced below) using the nodal values directly, instead of their best-fit residuals, achieves linear consistency. This is in contrast to the situation with (non-straight) edges and (non-planar) faces. Ultimately, the explanation for this dichotomy lies in the non-Euclidean character of edges and faces that are not straight or flat, whereas the 3D element is always Euclidean. A second difference is that the cell-local interpolants must exhibit an attribute that will be called "quadrature consistency," which must be explicitly enforced via a constraint on the minimization problem.

First we put forward the objective function for the element problem. It takes the form

$$(1-\beta_3)\sum_{a\in\mathcal{L}}|\sigma_a|\int_{\sigma_a}\Delta^2\,da \;+\; \beta_3\sum_{a\in\mathcal{L}\cup\mathcal{B}}\frac{1}{|\sigma_a|}\int_{\sigma_a}\varepsilon^2\,da, \tag{12}$$

where $\mathcal{L}$ is the set of facets that define inter-cell interfaces, $\mathcal{B}$ is the set of facets on the boundary of the element (these necessarily belong to faces), $\beta_3 \in (0,1)$, Δ is the jump in the directional derivative of the interpolant in the direction normal to an inter-cell facet, and ε is the jump in interpolant value on a facet. This jump is taken from the cell-local interpolants on the two adjacent cells in the case of inter-cell facets (set $\mathcal{L}$), and from a cell-local interpolant and a facet-local interpolant in the case of boundary facets (set $\mathcal{B}$). This latter contribution to the objective function brings in the requisite dependence on the element's nodal values.

The "quadrature consistency" requirement holds, essentially, that a constant stress field should exactly satisfy the weak form of equilibrium on a single element. This will occur if

$$\int_\Omega \varphi_{,i}\,dv = \int_{\partial\Omega} n_i\varphi da, \tag{13}$$

where φ is any interpolant on the element (e.g. a shape function), Ω is the element and n_i is the outward unit normal on the element's boundary $\partial\Omega$. This equation must hold when the integrals on both the left- and right-hand sides are evaluated using the element's quadrature rules. It amounts to a requirement that element interpolants satisfy the divergence theorem exactly when the element's quadrature rules are used. Satisfaction of this requirement, together with linear consistency, lead directly to passage of patch tests on arbitrary assemblages of elements.

To proceed further, quadrature rules on the element interior and on element faces must be made explicit. The weak-form integrals generally involve products of shape functions or shape-function derivatives, and some other function – e.g. a stress tensor or a given traction distribution. In the PEM, such integrals are evaluated by first forming a value of the other function for each cell; the function is assumed to be constant at this value over the cell. The integral over the complex of cells is then evaluated analytically, using the shape-function interpolant and the piecewise-constant representation for the other function. In the case of element-domain integrals, the "cells" in this description are the polyhedral cells into which the 3D element was subdivided. For boundary integrals, the cells derive from the subdivision of the domain-boundary face(s) that occurs as a by-product of subdivision of the element domain.

With the quadrature rules in hand, the quadrature-consistency constraint (13) can be cast explicitly as a constraint on the set $\{\mathbf{R}_a\}$ of interpolant maps for the element's cells. The right-hand side of (13) brings in the interpolant maps $\{\mathbf{Q}_a\}$ for the element's boundary facets, but these are considered to be fixed at this point. The $\mathbf{R}_a$'s are yet to be computed, based on minimization of (12), subject to the quadrature-consistency constraints (13).

To formulate this constrained minimization problem, let us consider a single vector $\mathbf{r}$ of polynomial coefficients for the collection of cell-local interpolants. Essentially, $\mathbf{r}$ consists of the vectors $\mathbf{R}_a\mathbf{p}$ stacked one on top of the other, where $\mathbf{p}$ is some fixed, but arbitrary, vector of nodal values for the element. Also, let $\mathbf{q}$ be a similar "stack" of polynomial coefficients $\mathbf{Q}_a\mathbf{p}$ for the facet-local interpolants on the element's boundary. With this notation, the objective function (12) can be cast in the form

$$\frac{1}{2}\,\mathbf{r}^T\mathbf{A}\mathbf{r} + \mathbf{r}^T\mathbf{G}\mathbf{q}, \tag{14}$$

whereas the constraints (13) take the form

$$\mathbf{B}\mathbf{r} = \mathbf{H}\mathbf{q}. \tag{15}$$

Our problem is to minimize (14) with respect to $\mathbf{r}$, subject to the linear constraints (15). Here, $\mathbf{B}$, $\mathbf{H}$ are the constraint matrices that follow from enforcement of (13). (Essentially, multiplication of the left- and right-hand sides of (15) by the column vector $\mathbf{p}$ of nodal values produces the left- and right-hand sides of (13), respectively.) Matrices $\mathbf{B}$ and $\mathbf{H}$ each have 3 rows, corresponding to the $i = 1, 2, 3$ components of (13).

To solve this constrained-minimization problem, it is convenient to first express all possible solutions $\mathbf{r}$ of (15) as

$$\mathbf{r} = \mathbf{B}^+\mathbf{H}\mathbf{q} + \mathbf{N}\mathbf{t}, \tag{16}$$

where $\mathbf{B}^+$ is the Moore-Penrose pseudo-inverse of $\mathbf{B}$, and $\mathbf{N}$ is a matrix whose columns consist of a basis for the nullspace of $\mathbf{B}$.[1] Substituting (16) into (14) and minimizing with

[1]Any matrix $\mathbf{A}$ (invertible or not, square or not) can be thought of as a bijective mapping from the row-space of $\mathbf{A}$ to its column-space. Being a bijection, this mapping has an inverse, which obviously maps the column-space of $\mathbf{A}$ to its row-space. This is the Moore-Penrose psuedo-inverse $\mathbf{A}^+$. It is equal to the regular inverse $\mathbf{A}^{-1}$ when $\mathbf{A}$ is invertible. If $\mathbf{A}$ is not invertible, then $\mathbf{A}\mathbf{x} = \mathbf{b}$ still has a solution, provided that $\mathbf{b}$ lies in the column space of $\mathbf{A}$. If it does, then in fact there is an infinity of solutions, all of which have the form $\mathbf{A}^+\mathbf{b}$ + (an arbitrary vector in the null-space of $\mathbf{A}$). High-quality routines have already been written to compute the pseudo-inverse of an arbitrary input matrix, as well as a basis for the matrix's null-space. These can be used in the present work.

respect to the parameters $\mathbf{t}$ results in a solution for $\mathbf{t}$, which can then be used in (16). The final result is

$$\mathbf{r} = \left[\mathbf{B}^{+}\mathbf{H} - (\mathbf{N}^{T}\mathbf{A}\mathbf{N})^{-1}\mathbf{N}^{T}(\mathbf{A}\mathbf{B}^{+}\mathbf{H} + \mathbf{N}^{T}\mathbf{G})\right] \mathbf{q}. \tag{17}$$

Remembering the meaning of $\mathbf{r}$ and $\mathbf{q}$, it is clear that (17) provides the desired relation between the $\mathbf{R}_a$ and $\mathbf{Q}_a$ matrices.