ICES REPORT 09-14

June 2009

Construction of H1-Conforming Hierarchical Shape Functions for Elements of All Shapes and Transfinite Interpolation

by

P. Gatto and L. Demkowicz

The Institute for Computational Engineering and Sciences
The University of Texas at Austin
Austin, Texas 78712

ICES REPORT 09-15

June 2009

Stochastic Finite Element Methods for Transport Equations

by

Regina C. Almeida

The Institute for Computational Engineering and Sciences
The University of Texas at Austin
Austin, Texas 78712

ICES REPORT 09-16

June 2009

A Class of Discontinuous Petrov-Galerkin Methods. Part II: Optimal Test Functions

by

L. Demkowicz and J. Gopalakrishnan

The Institute for Computational Engineering and Sciences
The University of Texas at Austin
Austin, Texas 78712

CONSTRUCTION OF H^1-CONFORMING HIERARCHICAL SHAPE FUNCTIONS FOR ELEMENTS OF ALL SHAPES AND TRANSFINITE INTERPOLATION

P. Gatto and L. Demkowicz

Institute for Computational Engineering and Sciences
The University of Texas at Austin, Austin, TX 78712, USA

Abstract

The paper reviews the construction of hierarchical H^1-conforming shape functions for 1D (edge), 2D (triangle, quad) and 3D (hexahedron, tetrahedron, prism, pyramid) elements. The logic of implementation is based on identifying a set of core (kernel, bubble) 1D and 2D functions, and the use of specific edge-to-element and face-to-element extensions. The resulting implementation allows for an instanteneous modification (or redefinition) of core functions without any changes in the rest of the FE code. The 3D shape functions conform automatically to 1D edge and 2D face shape functions defined in local edge and face coordinates. In other words, the dependence of 3D shape functions on different edge and face orientations is taken into account at the level of element shape functions routine. This eliminates the use of sign factors during the assembly and simplifies dramatically the implementation of constrained approximation allowing for the presence of hanging nodes. The construction of shape functions is directly related to transfinite interpolation techniques used in *Mesh Based Geometry* (MBG) descriptions.

Key words: geometry representation, higher order finite elements, transfinite interpolation

AMS subject classification: 65N30, 35L15

Acknowledgment

The work has been partially supported by grant # FA9550-06-0034 from the Air Force Office for Scientific Research.

1 Introduction

Mesh Based Geometry (MBG) description. The definition of a domain of interest is the starting point for the formulation of any boundary value problem. While this can be a trivial task for most academic problems, it turns out to be a rather challenging issue when dealing with engineering and biomedical applications.

Moreover, the standard definition of a 3D manifold that is known from differential geometry is not suitable for finite element methods. A 3D manifold is an object such that, for any of its point, we can find an open neighborhood (in the topology that the manifold naturally inherits from the Euclidean topology on $\mathbb{R}^3$) surrounding the point and a diffeomorphism between such neighborhood and the unit ball of $\mathbb{R}^3$; in rough terms, the smoothness of the diffeomorphisms gives the global smoothness of the manifold. The crucial issue is that we can view the manifold as the union of overlapping images of the unit ball under regular maps and this allows us to talk about global properties of the manifold, *i.e.* smoothness. When dealing with finite elements the previous strategy is reversed: we define the master blocks (tetrahedron, hexahedron, prism and pyramid) and describe the domain as the union of the images of those elements under some suitable mappings. In other words, we are creating a mesh-like description of our 3D object; at this stage no hanging nodes are allowed. This concept is known as *mesh based geometry* (MBG), see [1] and [2]. The concept of MBG is a substantial link between the FE community and the mesh generators community such as the development team of Sandia's CUBIT let by S.J.Owen; additional references about MBG and mesh generators are found in [10], [11], [12] and [13]. The price we have to pay when using a MBG description is the loss of control over smoothness at the interfaces between the different physical blocks; more specifically, since the maps overlap only at the interfaces, whose $\mathcal{H}^3$-measure is zero, we are not allowed to use the usual concept of $\mathcal{C}^k$ smoothness. Yet, for surfaces, we can still talk about G^1 smoothness (continuity of the normal) and G^2 smoothness (continuity of the curvature), as we discussed in [16].

In order to obtain a MBG description of the domain we need to use a mesh generator, that indeed produces linear blocks, and successively we must upgrade such blocks to curvilinear ones. Our upgrading procedure is based on the exact geometry of the domain, usually described in terms of bounding surfaces, that is input for the mesh generator. Notice that, when two surfaces meet, a sharp edge, *i.e.* a curve, is obtained. Construct of block parameterizations that match on the common faces and edges is a main issue; such compatibility condition is achieved through a bottom-up strategy that we briefly outline. The starting point is to construct parameterizations of all the curves, not only the sharp edges, that constitute our MBG model: curves that do not lay on any surface can certainly be assumed to be straight lines, while curves that lay on a surface but are not sharp edges can be reconstructed, *e.g.* as geodesics. The following step is to produce figure parameterizations by extending curve parameterizations; this task can be achieved through transfinite interpolation, a fairly well established technique that was first introduced by Gordon and Hall, see [4]. If the figure also needs to conform to a surface we must resort to a more complicated construction the employes transfinite interpolation in the parametric space; this construction, called parametric transfinite interpolation, was first proposed by Düester, see [3]. Finally, we need to extend the figure parameterizations to the blocks bounded by such figures; roughly speaking, this is accomplished by summing edge contributions and, possibly, face contributions, to a linear interpolation of the physical block. The main point of this construction is that it relies on extension of parameterizations, hence the desired compatibility condition is automatically met.

Hierarchical shape functions of arbitrary order. When constructing hierarchical spaces of shape functions it is convenient to group them into vertex modes, edge modes, face modes and interior modes. The vertex modes are functions that are linear along the element edges, while the remaining modes are bubbles of higher degree; for example, an edge bubble is supported on the pertaining edge, while it vanishes on the remaining edges. The construction of shape functions is a fairly trivial task for elements with a tensor product structure, such as the quad and the hexahedron: given a space of 1D shape functions – *e.g* Legendre polynomial or integrated Legendre polynomials – the shape functions are simply tensor products of 1D shape functions. This construction has been described in many works, among which we recall the famous book "Finite Element Analysis" by Szabó and Babuška, see [5], and a paper by Szabó, Duster and Rank, [6]. A fairly well established way of constructing triangle shape functions is by means of the Duffy's transformation that maps the master square onto the master triangle by collapsing one edge; this idea was originally proposed by Dubiner in [8]. Successively, Karniadakis and Sherwin in [7] have employed the same idea to extended the construction to tetrahedra; more recently, in [15], Zaglmair used the same approach to construct shape functions for the prism. Shape functions that are constructed via the Duffy's transformation are automatically compatible between elements of different shapes and enjoy fast integration properties; more precisely, for elements with a tensor product structure, an $\mathcal{O}(p^7)$ fast integration algorithm can be implemented, as discussed in [2]. In [3], Szabó and Babuška describe a construction of shape functions on the master triangle that does not rest on the Duffy's transformation; Devloo, in [9], taking inspiration from the construction on master triangle, extends the construction to elements of all types, including the pyramid. A similar construction of shape functions has been derived by Solin in [14], however he does not discuss the pyramidal element. A parallel between the construction of shape functions and transfinite interpolation is found in [5] for the triangular and the quadrilateral element, and in [6] for the quadrilateral element only. Comparison between the construction of shape functions and transfinite interpolation for the 3D elements in not discussed in any of the previously cited works. Finally, the pyramid is a much neglected element; construction of shape functions were proposed by Zaglmayr in [18], and Nigam and Phillips in [17].

Orientations. Orientations of geometrical entities play a fundamental role in the design of any finite element code. Many FE codes interface with a mesh generator, such as CUBIT or NETGEN, to produce a partition of the geometrical object; roughly speaking, the mesh generator outputs a list of points in the physical space and curves, figures and blocks connectivities by listing the pertaining vertices. The order in which vertices are listed dictates a global orientation for the geometrical entities. On the other hand, as soon as each figure is understood as the image of the master triangle or the master quad through some suitable mapping, local orientations for its edges are established; similarly, each block will induce local orientations on its faces and edges. In general local and global orientations do not agree; this fact requires the development of an assembly procedure of shape functions in order to construct globally continuous basis functions.

The development of an assembly procedure for 2D elements is fairly simple since orientations need to be accounted for only in the assembly of edge modes. For the ease of discussion, let's consider two adjacent quadrilateral elements; if the local orientations of the common edge are opposite, the corresponding edge

modes of the two elements will not agree along the shared edge. Notice, however, that because of the tensor product structure of the square, the edge modes reduce to the 1D shape functions along the pertaining edge. Since the 1D bubbles are in general either even or odd functions, the solution is fairly simple: when the local orientations do not agree, the odd modes of one element need to be premultiplied by -1; nothing needs to be done for the even modes. This technique also applies to triangles as soon as the restriction of the edge bubbles to the pertaining edge are the 1D shape functions; this is indeed the case for our construction.

Things become significantly more complicated for 3D elements; let's again focus on the hexahedral element. While the assembly procedure for the edge modes can be carried out with the same strategy described for 2D elements, the assembly procedure of the face modes needs to take into account the eight possible orientations of a quadrilateral face.

Scope of this presentation. In this paper we describe an implementation of H^1-conforming shape functions on the hexahedron, tetrahedron, prism and pyramid. We start by identifying sets of 1D, 2D, and 3D shape functions that span polynomials of type $\mathcal{P}$; since the 2D and 3D shape functions are only used to generate the interior modes of the master triangle and tetrahedron, they need not to be related to each other, nor to the set of 1D shape functions. This fact allows us to redefine the core functions at any time, greatly enhancing portability of the FE code. Our construction is also completely hierarchical: any shape function, when restricted to a face, coincides with the appropriate – recall that the face can be either a triangle or a quad – 2D shape function expressed in the local system of coordinates, and, when restricted to an edge, it coincides with a 1D shape function with respect to the local edge coordinate. Moreover, we realized that, if orientations are taken into account at the level of the element shape functions routines, the assembly procedure to generate globally continuous basis functions becomes trivial; this also simplifies dramatically the implementation of constrained approximation allowing for the presence of hanging nodes. The element orientation is given by a vector of edge orientations and a vector of face orientations; in general, the local orientations of the element edges and faces do not agree with the global orientations they inherit from the MBG model. A simple way to construct globally continuous basis functions is to generate shape functions that conform to the global, instead of local, orientation of edges and faces; since the global orientation of a particular geometrical entity (curve or figure) is known to all elements adjacent to such entity, then the shape functions of adjacent elements will agree along the shared geometrical entity. Finally, we emphasize how transfinite interpolation and our construction of shape functions closely resemble each other; this is a major advantage towards a successful development of a logically consistent geometry modeling package for a FE code.

In Section 2 we discuss the construction of shape functions on the unit interval and on the master triangle and square, along with transfinite interpolation on such elements; moreover, we compare our construction of edge modes for the master triangle with an alternative construction, that relies on the Duffy's transformation, proposed by Dubiner. Finally, we give a brief description of transfinite interpolation in parametric space and how it is employed to construct the maps that are needed by a MBG description. The most relevant section of the paper is Section 3; we discuss in full details the construction of shape functions for the most commonly

used three dimensional elements, and we extend it to the highly degenerate pyramidal element. Notice that, if we wish to create a mesh containing both tetrahedra and prisms, we need to have pyramids as connecting elements; this was one of the reasons that convinced us to extend our construction to the pyramid. Finally, for each element, we draw a parallel between the construction of shape functions and transfinite interpolation. In section 4 we discuss the main issues that we faced during implementation of our construction of shape functions and how we verified our code. In the last section of the paper we review its content and comment on possible extensions to $H(\mathrm{div})$ and $H(\mathrm{curl})$ spaces.

2 One- and Two-Dimensional Elements

One dimensional hierarchical shape functions. Let's present a quick overview of the most common spaces of one dimensional hierarchical shape functions, since they are indeed quite well known. The simplest choice are the *Peano* shape functions:

$$P_1 = 1 - \xi \quad ; \quad P_2 = \xi \quad ; \quad P_i = P_1(\xi)\, P_2(\xi)\, (2\xi - 1)^{i-3} \quad \text{for } i = 3, \ldots, p+1$$

which span the space $\mathcal{P}^p[0,1]$; moreover, P_1 and P_2 are vertex shape functions, while the remaining modes are either odd or even bubbles. Other common choices are the Legendre polynomials $\{L_i\}_i$ and the integrated Legendre polynomials $\{\ell_i\}_i$. One possible definition of the Legendre polynomials is through the recursive formula

$$L_0 = 1 \quad ; \quad L_1 = \xi \quad ; \quad L_i = \frac{2n-1}{n}\, \xi\, L_{i-1}(\xi) - \frac{i-1}{i} L_{i-2}\,(\xi) \quad \text{for } i = 2, \ldots, p$$

Legendre polynomials span $\mathcal{P}^p[-1,1]$ and enjoy the remarkable property of being a complete orthogonal set for both $L^2(-1,1)$ and $H^1(-1,1)$. Finally, if L_i's now denote the Legendre polynomials rescaled to the interval $[0,1]$, the integrated Legendre polynomials are defined as:

$$\ell_1 = 1 - \xi \quad ; \quad \ell_2 = \xi \quad ; \quad \ell_i = \int_0^\xi L_{i-2}(s)\, ds \quad \text{for } i = 3, \ldots, p+1$$

they span $\mathcal{P}^p[0,1]$ and it is easy to show that $\ell_3, \ell_4, \ldots$ are indeed bubbles. In the remaining of the paper we will use the notation $\hat{\chi}_i$ for the shape functions and order them in a way such that $\hat{\chi}_1, \hat{\chi}_2$ are the vertex shape functions, while the remaining $(p-1)$ ones are the bubbles. In order to keep the notation light, $\hat{\chi}$'s are also the generic shape functions we construct in the following sections; since the 1D shape functions depend on one variable only, no confusion arises.

Master square element. The master square element $[0,1]^2$, that is fully described in Fig. 1, is an anisotropic element that supports the space $\mathcal{Q}^{(p,q)}(\xi_1, \xi_2) = \mathcal{P}^p(\xi_1) \otimes \mathcal{P}^q(\xi_2)$. Since the 1D shape functions $\{\hat{\chi}_i\}_i$ form a base for $\mathcal{P}^p$, then a base for the tensor product space $\mathcal{Q}^{(p,q)}$ is canonically constructed as

$$\hat{\chi}_{i,j} = \hat{\chi}_i(\xi_1) \otimes \hat{\chi}_j(\xi_2) \quad \text{for } i = 1, \ldots, p+1 \ ; \ j = 1, \ldots, q+1$$

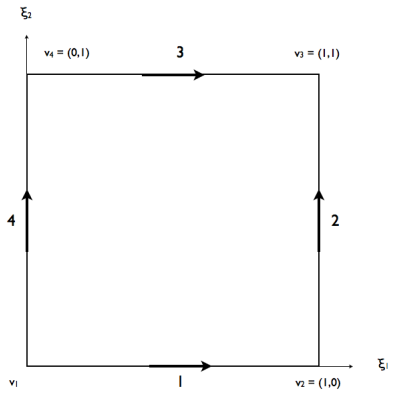

Figure 1: Master square.

where the tensor product indeed reduces to the standard algebraic product. Using a different numbering convention that more closely resembles the 1D case, the vertex shape functions are

$$\hat{\chi}_1 = \hat{\chi}_1(\xi_1)\,\hat{\chi}_1(\xi_2) \quad ; \quad \hat{\chi}_2 = \hat{\chi}_2(\xi_1)\,\hat{\chi}_1(\xi_2) \quad ; \quad \hat{\chi}_3 = \hat{\chi}_2(\xi_1)\,\hat{\chi}_2(\xi_2) \quad ; \quad \hat{\chi}_4 = \hat{\chi}_1(\xi_1)\,\hat{\chi}_2(\xi_2)$$

The edge shape functions associated to the first edge are

$$\hat{\chi}_{1,i} = \hat{\chi}_{i+2}(\xi_1)\,\hat{\chi}_1(\xi_2) \quad \text{for } i = 1, \ldots, p-1 \tag{2.1}$$

with similar definitions for the remaining edge; we remark that, in the context of equation (2.1), $\hat{\chi}_1(\xi_2)$ is sometimes referred to as the edge blending function. Finally, we use the following notations for the interior modes, *i.e* bubbles:

$$\hat{\chi}^{\text{bub}}_{i,j} = \hat{\chi}_{i+2}(\xi_1)\,\hat{\chi}_{j+2}(\xi_2) \quad \text{for } i = 1, \ldots, p-1 \;;\; \text{for } j = 1, \ldots, q-1$$

Transfinite Interpolation on a rectangle. As anticipated, transfinite interpolation strictly resembles the construction of shape functions. Let $\boldsymbol{\alpha}_i$'s be the parameterizations of the four edges of a rectangle K – possibly exhibiting curved edges – lying in a two dimensional physical space and having vertices $\mathbf{v}_i$'s. A surjective map $[0,1]^2 \mapsto K$ is obtained adding together the bilinear interpolation between the vertices, Φ^{lin}, and four vector-valued edge bubbles Φ^{edge}_i. The bilinear interpolation is easily expressed in terms of vertex shape functions as $\Phi^{\text{lin}} = \sum_{i=1}^4 \hat{\chi}_i\,\mathbf{v}_i$; mimicking the construction of edge shape functions, the first edge bubble is obtained as

$$\Phi^{\text{edge}}_1 = \big[\boldsymbol{\alpha}_1(\xi_1) - \hat{\chi}_1(\xi_1)\,\mathbf{v}_1 - \hat{\chi}_2(\xi_1)\,\mathbf{v}_2\big]\hat{\chi}_1(\xi_2)$$

where the term in brackets is a bubble in the ξ_1 direction and $\hat{\chi}_1(\xi_2)$ is the edge blending function; notice the similarity between the last formula and equation (2.1). The previous construction extends to the remaining edges with obvious modifications; the transfinite interpolation of K is given by $\Phi^{\text{lin}} + \sum_{i=1}^4 \Phi^{\text{edge}}_i$.

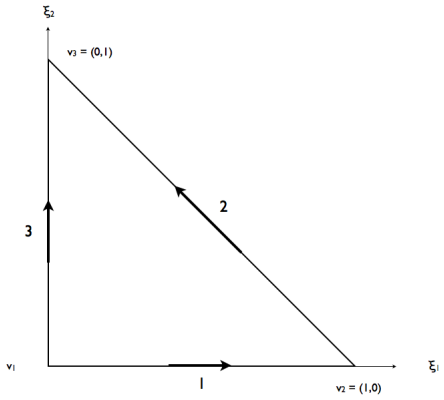

Figure 2: Master triangle.

Master triangle. The master triangle $\{(\xi_1, \xi_2) \text{ s.t. } \xi_1, \xi_2 \geq 0 \; ; \; \xi_1 + \xi_2 \leq 1\}$, described in Fig. 2, is an isotropic element that supports the space $\mathcal{P}^p(\xi_1, \xi_2)$. While the construction of the edge bubbles still relies on the set $\{\hat{\chi}_i\}_i$ of 1D shape functions, construction of the interior modes will require a set $\{\hat{\psi}_j\}_j$ of functions that span $\mathcal{P}^p$. Vertex shape functions coincide with the affine coordinates of the master triangle, *i.e.*

$$\hat{\chi}_1 = \lambda_1 = 1 - \xi_1 - \xi_2 \quad ; \quad \hat{\chi}_2 = \lambda_1 = \xi_1 \quad ; \quad \hat{\chi}_3 = \lambda_1 = \xi_2$$

while construction of edge shape functions is more complicated because of the lack of a tensor product structure on the element. Let's focus our attention on the first edge, $[\mathbf{v}_1, \mathbf{v}_2]$; a projection from the master triangle onto the interval $[0, 1]$ is defined as

$$(\xi_1, \xi_2) \mapsto \zeta \quad ; \quad \zeta = \frac{\lambda_2 - \lambda_1 + 1}{2} \tag{2.2}$$

Indeed this is equivalent to setting up a local coordinate on the first edge and projecting onto it in the direction of the median. A first attempt to define edge shape functions is the following:

$$\hat{\chi}_{1,i} = \hat{\chi}_{i+2}(\zeta)\, \lambda_1(\xi_1, \xi_2)\, \lambda_2(\xi_1, \xi_2)$$

where $\lambda_1 \lambda_2$ is the edge blending function; notice, however, that the restrictions of such functions to the edge does not recover the set of 1D shape functions. While this is not a problem if we aim to develop a code that supports only triangles, it becomes a major issue if we want to support both triangles and quads: in fact we cannot generate continuos basis functions for the edges that are shared between a quad and a triangle by simply gluing shape functions together. We overcome this problem by adopting the construction:

$$\hat{\chi}_{1,i} = \frac{\hat{\chi}_{i+2}(\zeta)}{(1-\zeta)\zeta}\, \lambda_1(\xi_1, \xi_2)\, \lambda_2(\xi_1, \xi_2)$$

We will refer to $\hat{\chi}_i^{\text{ker}} = \hat{\chi}_{i+2}(\zeta)/(1-\zeta)\zeta$ as the i-th edge kernel function; notice that since $\hat{\chi}_{i+2}$ is a polynomial bubble, then kernel functions are not singular at the endpoints as it might seem at first glance. Finally, the interior modes are products of the $\hat{\psi}_i$'s with the blending function $\lambda_1\lambda_2\lambda_3$:

$$\hat{\chi}_i^{\text{bub}} = \hat{\psi}_i(\xi_1,\xi_2)\,\lambda_1(\xi_1,\xi_2)\,\lambda_2(\xi_1,\xi_2)\,\lambda_3(\xi_1,\xi_2)$$

Transfinite Interpolation on a triangle. Transfinite interpolation on a triangle resembles the construction described on a rectangle, *i.e.* the sum of a linear interpolation and edge contributions, with the understanding that kernel functions are used in place of the bubble functions in the construction of the vector-valued edge bubbles. Recalling the notations we introduced for the rectangle, the linear interpolation is $\Phi^{\text{lin}} = \sum_{i=1}^{3} \hat{\chi}_i\,\mathbf{v}_i$, while the first edge bubble is

$$\Phi_1^{\text{edge}} = \left[\frac{\boldsymbol{\alpha}_1(\zeta) - \hat{\chi}_1(\zeta)\,\mathbf{v}_1 - \hat{\chi}_2(\zeta)\,\mathbf{v}_2}{(1-\zeta)\zeta}\right]\lambda_1(\xi_1,\xi_2)\lambda_2(\xi_1,\xi_2)$$

where the map $(\xi_1,\xi_2) \mapsto \zeta$ was defined in (2.2). The term in brackets is a vector-valued kernel function; notice that while the kernel function defined in the context of shape functions is not singular at the endpoints of the edge it pertains to, this is not the case for transfinite interpolation. The numerator indeed vanishes at the endpoints, but since the edge parameterization is not *a priori* a polynomial function, in general we cannot factor out $(1-\zeta)\zeta$. The previous construction extends to the remaining edges with obvious modifications and the the transfinite interpolation is given by $\Phi^{\text{lin}} + \sum_{i=1}^{3} \Phi_i^{\text{edge}}$.

Dubiner's construction for a triangle. An alternative construction of edge bubbles for the triangle, that relies on the Duffy's transformation, was proposed by Dubiner. Let (x_1,x_2) and (ξ_1,ξ_2) be the coordinates of the master square and the master triangle respectively, then the map

$$\xi_1 = x_1(1-x_2) \qquad ; \qquad \xi_2 = x_2$$

is the Duffy's transformation relative to the bottom edge of the square. It can be shown that the Duffy transformation maps the space $\mathcal{Q}^{(p,p)}(x_1,x_2)$ onto a space that strictly contains $\mathcal{P}^p(\xi_1,\xi_2)$. Let $u \in \mathcal{P}_0^p[0,1]$, *i.e.* the space of polynomials of degree at most p on $[0,1]$ vanishing at the endpoints, then, using the Duffy's transformation, we have that:

$$u(x_1)\,(1-x_2)^p \mapsto u\left(\frac{\xi_1}{1-\xi_2}\right)(1-\xi_2)^p$$

where the right-hand side is not a singular function on the line $[\xi_2 \equiv 1]$, since u is a polynomial of degree at most p. Notice that we need to take the edge bubble $u(x_1)\,(1-x_2)^p$, that indeed belongs to the space $\mathcal{Q}^{(p,p)}$, over the master square in order to cancel the singularity of the Duffy's transformation. This points to the fact that we need to start from an isotropic master square element in order to use Dubiner's construction; this fact poses a conflict with our overall logic.

Finally, we remark that every edge will require a different Duffy's transformation; also, such construction does not fit into our general framework. Suppose that u is an arbitrary bubble function (not necessarily a polynomial), then the function $u\big(\xi_1/(1-\xi_2)\big)\,(1-\xi_2)^p$ is in general singular on the line $[\xi_2 \equiv 1]$ since in this case the choice of the blending function is not suitable; on the other hand, there is no appropriate choice of a blending function to accommodate the singularity of the Duffy's transformation for an arbitrary bubble function. Therefore, this construction is not suitable for our transfinite interpolation.

Transfinite Interpolation in parametric space. The transfinite interpolations for the rectangle and the triangle are easily generalized to the hexahedron and the tetrahedron; they indeed produce physical elements whose edges conform to the images of the parameterizations $\boldsymbol{\alpha}_i$'s. Yet, if a physical element has a face adjacent to a bounding surface, its transfinite interpolation, in general, does not conform to the bounding surface; in order to address this issue we need to include a face contribution, *i.e.* a face bubble, in the interpolation procedure. The first step to construct a face bubble is to find a map from the appropriate 2D master element – triangle or quad – onto the face in physical space; this task is performed through parametric transfinite interpolation. Suppose that, beside the edge parameterizations $(x_1, x_2, x_3) = \boldsymbol{\alpha}_i(s)$, an explicit parameterization $(x_1, x_2, x_3) = F(u, v)$ of the bounding surface is known; then, at least in principle, maps $(u, v) = \boldsymbol{\gamma}_i(s)$ where s is the curvilinear abscissa, can be obtained. The maps $\boldsymbol{\gamma}_i$'s describe the edges of the face in the parametric space (u, v); hence the problem has been reduced to a 2D transfinite interpolation between the reference space (ξ_1, ξ_2) and the parametric space (u, v). Finally, the parametric transfinite interpolation $\boldsymbol{\beta}_j$ of the face is the composition of the 2D transfinite interpolation and map F.

Example: parametric transfinite interpolation on a spherical surface. Let's analyze the case of a spherical surface of unit radius; while the implicit parameterization of this surface is $x_1^2 + x_2^2 + x_3^2 = 1$, the explicit parameterization F is given by

$$x_1 = \sin\varphi\,\cos\theta \quad , \quad x_2 = \sin\varphi\,\sin\theta \quad , \quad x_3 = \cos\varphi$$

where $\theta \in [-\pi, \pi)$ and $\varphi \in [0, \pi)$ are suitable choices for the surface parameters. Let's consider a triangular figure laying on the spherical surface; the three edge parameterizations $(x_1, x_2, x_3) = \boldsymbol{\alpha}_i(s)$ are known, since the bounding curves are either sharp edges, which are determined by the intersection of two or more surfaces, or have been reconstructed employing the fact that they lay on the surface (in this case geodesics are a natural choice), see Fig. 3. Focusing again on the case of the sphere, the first step towards building the desired interpolation, is determining parameterizations $(\theta, \varphi) = \boldsymbol{\gamma}_i(s)$ for the edges in the parametric space; in theory, this can achieved through the composition $\boldsymbol{\gamma}_i = F^{-1} \circ \boldsymbol{\alpha}_i$, which requires inverting map F. While in this case map F can inverted analytically for every point beside the south and north pole, in the general framework it can be inverted numerically, *e.g* through Newton-Rapson iterations, and the implicit function theorem needs to be used to determine derivatives. The fact that map F^1 is singular at the south and north pole does not constitute a big issue, since by means of a rigid body motion we can always reduce to a triangular figure such that its first vertex belongs to the x_1 axis, and its second vertex lays on the

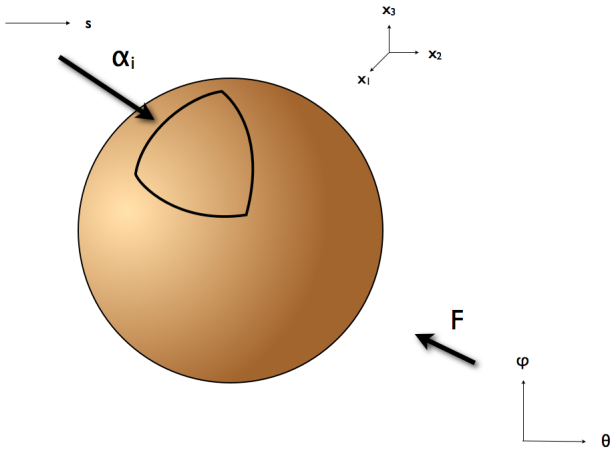

Figure 3: Edge and surface parameterizations in physical space

(x_1, x_2) plane; if the figure has reasonable shape, this motion rules out the fact that the third vertex is near to either the south or the north pole. Therefore we have reduced the problem to determining a 2D transfinite interpolation between the master triangle and a triangular figure in (θ, φ) whose three edge parameterization $\boldsymbol{\gamma}_i$ are known; we employ the technique of transfinite interpolation on a triangle. We start by building the linear interpolation and then we add the edge contributions, *i.e* the edge bubbles. Finally, if R is the 2D interpolation thus obtained, the parametric transfinite interpolation $\boldsymbol{\beta}$ is obtained through the composition $\boldsymbol{\beta} = F \circ R$; this whole construction is outlined in Fig. 4.

Elements of variable order. In the discussion carried out so far, we characterized the order of approximation of quadrilateral elements by (p, q), and the order of approximation of triangular elements by p. Those are indeed the orders of approximation associated to the interior nodes of the elements; when elements of different order are placed next to each other, an appropriate order of approximation needs to be determined for the common edge. Let's focus on the quadrilateral element, since discussion of the triangle is analogous: clearly the order of approximation of the horizontal edges cannot exceed p, while the order of approximation of the vertical edges cannot exceed q. We choose to pick the minimum between the orders of approximation the edge inherits from the adjacent elements. Although we feel like this is the most natural choice, other ones are possible. For example, Szabó assigns to the shared edge the maximum order of approximation; this requires the non trivial task of extending a high order polynomial defined on the edge to a lower order one defined on the entire element.

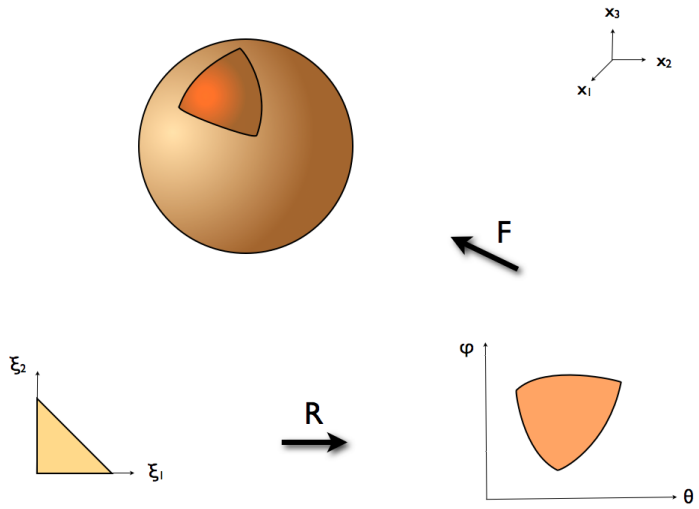

Figure 4: Transfinite interpolation in parametric space.

3 Three Dimensional Elements

The construction of shape functions for three dimensional elements rests upon the space $\{\hat{\chi}_i\}_i$ of 1D shape functions previously defined, along with two additional spaces: a space $\{\hat{\psi}_j\}_j$ of bubble functions on the master triangle, and a space $\{\hat{\varphi}_k\}_k$ that spans $\mathcal{P}^p(\xi_1, \xi_2, \xi_3)$. Notice that we are committing a slight abuse of notation since in the subsection about the master triangle $\hat{\psi}_j$'s were not bubbles; in the new contest we can certainly choose $\hat{\psi}_j = \hat{\chi}_j^{\text{bub}}$ where the $\hat{\chi}_j^{\text{bub}}$'s are the interior modes for the master triangle previously defined.

Master hexahedron. The master hexahedron $[0,1]^3$, which is described in Fig. 5, is an immediate extension of the master square; it is an anisotropic element that supports the space $\mathcal{Q}^{(p,q,r)} = \mathcal{P}^p \otimes \mathcal{P}^q \otimes \mathcal{P}^r$. Because of its tensor product structure, the shape functions are products of the 1D shape functions $\{\hat{\chi}_i\}_i$; in particular the vertex shape functions are

$$\hat{\chi}_1 = \hat{\chi}_1(\xi_1)\,\hat{\chi}_1(\xi_2)\,\hat{\chi}_1(\xi_3) \qquad \hat{\chi}_2 = \hat{\chi}_2(\xi_1)\,\hat{\chi}_1(\xi_2)\,\hat{\chi}_1(\xi_3)$$
$$\hat{\chi}_3 = \hat{\chi}_2(\xi_1)\,\hat{\chi}_2(\xi_2)\,\hat{\chi}_1(\xi_3) \qquad \hat{\chi}_4 = \hat{\chi}_1(\xi_1)\,\hat{\chi}_2(\xi_2)\,\hat{\chi}_1(\xi_3)$$
$$\hat{\chi}_5 = \hat{\chi}_1(\xi_1)\,\hat{\chi}_1(\xi_2)\,\hat{\chi}_2(\xi_3) \qquad \hat{\chi}_6 = \hat{\chi}_2(\xi_1)\,\hat{\chi}_1(\xi_2)\,\hat{\chi}_2(\xi_3)$$
$$\hat{\chi}_7 = \hat{\chi}_2(\xi_1)\,\hat{\chi}_2(\xi_2)\,\hat{\chi}_2(\xi_3) \qquad \hat{\chi}_8 = \hat{\chi}_1(\xi_1)\,\hat{\chi}_2(\xi_2)\,\hat{\chi}_2(\xi_3)$$

The first edge bubbles are

$$\hat{\chi}^{\text{edge}}_{1,i} = \hat{\chi}_{i+2}(\xi_1)\,\hat{\chi}_1(\xi_2)\,\hat{\chi}_1(\xi_3) \quad \text{for } i = 1,\dots,p-1$$

and the first face bubbles are

$$\hat{\chi}^{\text{face}}_{1,i,j} = \hat{\chi}_{i+2}(\xi_1)\,\hat{\chi}_{j+2}(\xi_2)\,\hat{\chi}_1(\xi_3) \quad \text{for } i = 1,\dots,p-1 \;;\; j = 1,\dots,q-1$$

Generalizations to the remaining edges and faces is obvious. Finally, the interior modes are

$$\hat{\chi}^{\text{bub}}_{i,j,k} = \hat{\chi}_{i+2}(\xi_1)\,\hat{\chi}_{j+2}(\xi_2)\,\hat{\chi}_{k+2}(\xi_3) \quad \text{for } i = 1,\dots,p-1 \;;\; j = 1,\dots,q-1 \;;\; k = 1,\dots,r-1$$

Transfinite interpolation on a hexahedron. As anticipated, when dealing with three dimensional elements, transfinite interpolation needs to take also into account face bubbles. Edge parameterizations $\boldsymbol{\alpha}_i$'s are supposed to be known and face parameterizations $\boldsymbol{\beta}_j$'s can be obtained through parametric transfinite interpolation as previously discussed. The linear interpolation is $\Phi^{\text{lin}} = \sum_{i=1}^{8} \hat{\chi}_i\,\mathbf{v}_i$; for the sake of simplicity let's focus on the first edge, $[\mathbf{v}_1, \mathbf{v}_2]$, and the first face, $[\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4]$. The edge bubble is obtained as

$$\Phi^{\text{edge}}_1 = \Big\{\boldsymbol{\alpha}_1(\xi_1) - \big[\hat{\chi}_1(\xi_1)\mathbf{v}_1 + \hat{\chi}_2(\xi_1)\mathbf{v}_2\big]\Big\}\hat{\chi}_1(\xi_2)\,\hat{\chi}_1(\xi_3)$$

For ease of notation let's first define the local face bubble $\Phi^{\text{face bub}}_1$; it is obtained subtracting the bilinear interpolation of the face, which is the first bracketed term in the following expression, and the local edge bubbles, which are enclosed between curly braces, to the face parameterization $\boldsymbol{\beta}_1$:

$$\begin{aligned}\Phi^{\text{face bub}}_1 = \boldsymbol{\beta}_1(\xi_1,\xi_2) &- \big[\hat{\chi}_1(\xi_1)\hat{\chi}_1(\xi_2)\mathbf{v}_1 + \hat{\chi}_2(\xi_1)\hat{\chi}_1(\xi_2)\mathbf{v}_2 + \hat{\chi}_2(\xi_1)\hat{\chi}_2(\xi_2)\mathbf{v}_3 + \hat{\chi}_1(\xi_1)\hat{\chi}_2(\xi_2)\mathbf{v}_4\big] \\ &- \Big\{\big[\boldsymbol{\alpha}_1(\xi_1) - \hat{\chi}_1(\xi_1)\mathbf{v}_1 - \hat{\chi}_2(\xi_1)\mathbf{v}_2\big]\hat{\chi}_1(\xi_2)\Big\} - \Big\{\big[\boldsymbol{\alpha}_2(\xi_2) - \hat{\chi}_1(\xi_2)\mathbf{v}_2 - \hat{\chi}_2(\xi_2)\mathbf{v}_3\big]\hat{\chi}_2(\xi_1)\Big\} \\ &- \Big\{\big[\boldsymbol{\alpha}_3(\xi_1) - \hat{\chi}_1(\xi_1)\mathbf{v}_4 - \hat{\chi}_2(\xi_1)\mathbf{v}_3\big]\hat{\chi}_2(\xi_2)\Big\} - \Big\{\big[\boldsymbol{\alpha}_4(\xi_2) - \hat{\chi}_1(\xi_2)\mathbf{v}_1 - \hat{\chi}_2(\xi_2)\mathbf{v}_4\big]\hat{\chi}_1(\xi_1)\Big\}\end{aligned}$$

where $\mathrm{e}_1, \mathrm{e}_2, \mathrm{e}_3, \mathrm{e}_4$ are indeed the edges of the first face, as can be deduced from Fig. 5. Because of the tensor product structure of the hexahedron, the previous expression simplifies to:

$$\begin{aligned}\Phi^{\text{face bub}}_1 = \boldsymbol{\beta}_1(\xi_1,\xi_2) &- \Big\{\big[\boldsymbol{\alpha}_1(\xi_1) - \hat{\chi}_2(\xi_1)\mathbf{v}_2\big]\hat{\chi}_1(\xi_2)\Big\} - \Big\{\big[\boldsymbol{\alpha}_2(\xi_2) - \hat{\chi}_2(\xi_2)\mathbf{v}_3\big]\hat{\chi}_2(\xi_1)\Big\} \\ &- \Big\{\big[\boldsymbol{\alpha}_3(\xi_1) - \hat{\chi}_1(\xi_1)\mathbf{v}_4\big]\hat{\chi}_2(\xi_2)\Big\} - \Big\{\big[\boldsymbol{\alpha}_4(\xi_2) - \hat{\chi}_1(\xi_2)\mathbf{v}_1\big]\hat{\chi}_1(\xi_1)\Big\}\end{aligned}$$

this makes the implementation of the transfinite interpolation computationally cheaper than it looks at first glance. The face bubble is obtained multiplying the local face bubble by the appropriate face blending function, $\Phi^{\text{face}}_1 = \Phi^{\text{face bub}}_1\,\hat{\chi}_1(\xi_3)$. Finally, the transfinite interpolation we seek is

$$\Phi^{\text{lin}} + \sum_{i=1}^{12} \Phi^{\text{edge}}_i + \sum_{j=1}^{6} \Phi^{\text{face}}_j$$

Master tetrahedron. The construction of shape functions on the master tetrahedron, which is fully described in Fig. 6 and whose domain are the points $(\xi_1, \xi_2, \xi_3) \in \mathbb{R}^3$ such that $\xi_1,\ \xi_2,\ \xi_3 \geq 0$ and $\xi_1 + \xi_2 + \xi_3 \leq 1$, extends the philosophy previously developed for the master triangle. Let's first define a projection onto edge $[\mathbf{v}_i, \mathbf{v}_j]$:

$$(\xi_1, \xi_2, \xi_3) \mapsto \zeta \quad ; \quad \zeta = \frac{\lambda_j - \lambda_i + 1}{2} \tag{3.3}$$

where λ_i's are the affine coordinates with respect to the master tetrahedron. This is indeed the projection parallel to the plane passing through the mid-point of edge $[\mathbf{v}_i, \mathbf{v}_j]$ and the endpoints of the opposite edge; such a projection sets up a local coordinate ζ in the direction $[\mathbf{v}_i \rightarrow \mathbf{v}_j]$, as illustrated in Fig. 6. Projection onto face $[\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k]$ is defined as

$$(\xi_1, \xi_2, \xi_3) \mapsto (\zeta_1, \zeta_2) \quad ; \quad \zeta_1 = \lambda_j + \frac{\lambda_l}{3}\ ,\ \zeta_2 = \lambda_k + \frac{\lambda_l}{3} \tag{3.4}$$

where l refers to the vertex opposite to the face. This is the projection in the direction of the line connecting the center of mass of the face to the opposite vertex; it sets up local coordinates ζ_1, in the direction $[\mathbf{v}_i \rightarrow \mathbf{v}_j]$, and ζ_2 in the direction $[\mathbf{v}_i \rightarrow \mathbf{v}_k]$. While the vertex shape functions are the affine coordinates

$$\hat{\chi}_1 = \lambda_1 = 1 - \xi_1 - \xi_2 - \xi_3 \quad ; \quad \hat{\chi}_2 = \lambda_2 = \xi_1 \quad ; \quad \hat{\chi}_3 = \lambda_3 = \xi_2 \quad ; \quad \hat{\chi}_4 = \lambda_4 = \xi_3$$

construction of the remaining modes relies on 1D bubbles $\{\hat{\chi}_i\}_{i\geq 3}$, 2D bubbles $\{\hat{\psi}_j\}_j$, and the space $\{\hat{\varphi}_k\}_k$. Let's focus on the first edge and the first face; construction of the first edge modes relies on the edge kernel functions previously defined:

$$\hat{\chi}^{\text{edge}}_{1,i} = \hat{\chi}^{\text{ker}}_i(\zeta)\ \lambda_1(\xi_1, \xi_2, \xi_3)\lambda_2(\xi_1, \xi_2, \xi_3)$$

where ζ is determined by the projection on the first edge and $\lambda_1\lambda_2$ is the edge blending function. Construction of face bubbles is done in a similar fashion:

$$\hat{\chi}^{\text{face}}_{1,i} = \frac{\hat{\psi}_i(\zeta_1, \zeta_2)}{(1 - \zeta_1 - \zeta_2)\zeta_1\zeta_2}\lambda_1(\xi_1, \xi_2, \xi_3)\lambda_2(\xi_1, \xi_2, \xi_3)\lambda_3(\xi_1, \xi_2, \xi_3)$$

where ζ_1, ζ_2 are determined by the projection on the first face and $\lambda_1\lambda_2\lambda_3$ is the face blending function. Similarly to the 1D case, we refer to function $\hat{\chi}^{\text{face ker}}_i = \hat{\psi}_i(\zeta_1, \zeta_2)/(1 - \zeta_1 - \zeta_2)\zeta_1\zeta_2$ as the face kernel function.

Transfinite interpolation on a tetrahedron. Recalling the notation introduced for the master hexahedron, the transfinite interpolation for the master tetrahedron is given by:

$$\Phi^{\text{lin}} + \sum_{i=1}^{6} \Phi^{\text{edge}}_i + \sum_{j=1}^{4} \Phi^{\text{face}}_j$$

where the linear interpolation is of course $\Phi^{\text{lin}} = \sum_{i=1}^{4} \hat{\chi}_i\, \mathbf{v}_i$. As usual let's focus on the first edge $[\mathbf{v}_1, \mathbf{v}_2]$ and the first face $[\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]$; recall that the first face is bounded by edges e_1, e_2, e_3. The edge bubble is

given by

$$\Phi_1^{\text{edge}} = \left[\frac{\boldsymbol{\alpha}_1(\zeta) - \hat{\chi}_1(\zeta)\,\mathbf{v}_1 - \hat{\chi}_2(\zeta)\,\mathbf{v}_2}{(1-\zeta)\zeta}\right]\lambda_1(\xi_1,\xi_2,\xi_3)\lambda_2(\xi_1,\xi_2,\xi_3)$$

where ζ is given by the appropriate choice of equation (3.3); the term in brackets is the edge kernel and $\lambda_1\lambda_2$ is the edge blending function. Construction of the face bubble follows the same philosophy employed for the hexahedron, however the local face bubble needs to be replaced by the local face kernel because of the lack of tensor product structure. The face kernel function is defined as:

$$\begin{aligned}\Phi_1^{\text{face ker}} = \Big(&\boldsymbol{\beta}_1(\zeta_1,\zeta_2) - \big[(1-\zeta_1-\zeta_2)\mathbf{v}_1 + \zeta_1\mathbf{v}_2 + \zeta_2\mathbf{v}_3\big] \\ &- \Big\{\big[\boldsymbol{\alpha}_1(\zeta) - \hat{\chi}_1(\zeta)\mathbf{v}_1 - \hat{\chi}_2(\zeta)\mathbf{v}_2\big]\lambda_1(\zeta_1,\zeta_2)\lambda_2(\zeta_1,\zeta_2)\Big\} \\ &- \Big\{\big[\boldsymbol{\alpha}_2(\zeta) - \hat{\chi}_1(\zeta)\mathbf{v}_2 - \hat{\chi}_2(\zeta)\mathbf{v}_3\big]\lambda_2(\zeta_1,\zeta_2)\lambda_3(\zeta_1,\zeta_2)\Big\} \\ &- \Big\{\big[\boldsymbol{\alpha}_3(\zeta) - \hat{\chi}_1(\zeta)\mathbf{v}_1 - \hat{\chi}_2(\zeta)\mathbf{v}_3\big]\lambda_1(\zeta_1,\zeta_2)\lambda_3(\zeta_1,\zeta_2)\Big\}\Big)/(1-\zeta_1-\zeta_2)\zeta_1\zeta_2\end{aligned}$$

where the first bracketed term is the linear interpolation and the terms in curly braces are the local edge bubbles; we remark that the λ_i's are the affine coordinates with respect to the master triangle. Coordinates ζ_1, ζ_2 are determined using (3.4); moreover the different instances of ζ need to be interpreted as the local coordinate on the first, second and third edge respectively, which are determined by (3.3). Finally, the face bubble is obtained multiplying the face kernel by the face blending function $\lambda_1\lambda_2\lambda_3$, where λ_i's are the affine coordinates with respect to the master tetrahedron.

Master prism. The master prism, described in Fig. 7, is an element with a partial tensor product structure; it is isotropic in the (ξ_1,ξ_2) plane and supports the space $\mathcal{P}^p(\xi_1,\xi_2)\otimes\mathcal{P}^q(\xi_3)$. Shape functions on the prism are immediately obtained by considering tensor products of the form $\hat{\chi}_i(\xi_1,\xi_2)\,\hat{\chi}_j(\xi_3)$ where $\hat{\chi}_i$'s are the master triangle shape functions and $\hat{\chi}_j$'s are the 1D shape functions previously constructed; the shape functions thus obtain are indeed vertex, edge, face, and interior modes. For ease of implementation, as we will explain in the next section, it is convenient to express the modes associated to horizontal edges as products of a 1D kernel function and a blending function, and the modes associated to vertical edges as products of a 1D bubble and a blending function; similarly, since the vertical faces are rectangles, the vertical face bubbles turn out to be products of a 1D bubble, a 1D kernel, and a blending function.

The vertex shape functions are products of the 1D vertex shape functions $\hat{\chi}_1, \hat{\chi}_2$ and the master triangle affine coordinates $\lambda_1, \lambda_2, \lambda_3$:

$$\begin{aligned}\hat{\chi}_1 &= \lambda_1(\xi_1,\xi_2)\lambda_2(\xi_1,\xi_2)\hat{\chi}_1(\xi_3) & \hat{\chi}_2 &= \lambda_2(\xi_1,\xi_2)\lambda_3(\xi_1,\xi_2)\hat{\chi}_1(\xi_3) \\ \hat{\chi}_3 &= \lambda_1(\xi_1,\xi_2)\lambda_3(\xi_1,\xi_2)\hat{\chi}_1(\xi_3) & \hat{\chi}_4 &= \lambda_1(\xi_1,\xi_2)\lambda_2(\xi_1,\xi_2)\hat{\chi}_2(\xi_3) \\ \hat{\chi}_5 &= \lambda_2(\xi_1,\xi_2)\lambda_3(\xi_1,\xi_2)\hat{\chi}_2(\xi_3) & \hat{\chi}_6 &= \lambda_1(\xi_1,\xi_2)\lambda_3(\xi_1,\xi_2)\hat{\chi}_2(\xi_3)\end{aligned}$$

In order to discuss edge bubbles let's focus on a horizontal edge, $\mathrm{e}_1 = [\mathbf{v}_1, \mathbf{v}_2]$, and a vertical edge, $\mathrm{e}_7 = [\mathbf{v}_1, \mathbf{v}_4]$; the horizontal edge modes are

$$\hat{\chi}^{\text{edge}}_{1,i} = \hat{\chi}_{1,i}(\xi_1, \xi_2)\hat{\chi}_1(\xi_3) = \hat{\chi}^{\text{ker}}_i(\zeta)\lambda_1(\xi_1, \xi_2)\lambda_2(\xi_1, \xi_2)\hat{\chi}_1(\xi_3)$$

where $\hat{\chi}_{1,i}$'s are the bubbles associated to the first edge of the master triangle, $\hat{\chi}^{\text{ker}}_i$ are the 1D kernel function and the coordinate ζ is determine by the projection on the first edge of the master triangle; we can refer to function $\lambda_1\lambda_2\hat{\chi}_1$ as the blending function relative to edge e_1. The construction of the vertical edge modes is simpler:

$$\hat{\chi}^{\text{edge}}_{7,i} = \hat{\chi}_{i+2}(\xi_3)\lambda_1(\xi_1, \xi_2)$$

where $\hat{\chi}_{i+2}$'s are the 1D bubbles and λ_1 is the edge blending function. Let's now consider the horizontal face $\mathrm{f}_1 = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]$ and the vertical face $\mathrm{f}_3 = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_5, \mathbf{v}_4]$; we have, respectively, that

$$\hat{\chi}^{\text{face}}_{1,i} = \hat{\chi}^{\text{bub}}_i(\xi_1, \xi_2)\hat{\chi}_1(\xi_3)$$

where $\hat{\chi}^{\text{bub}}_i$'s are the master triangle bubbles, and

$$\hat{\chi}^{\text{face}}_{3,i,j} = \hat{\chi}_{1,i}(\xi_1, \xi_2)\,\hat{\chi}_{j+2}(\xi_3) = \hat{\chi}^{\text{ker}}_i(\zeta)\lambda_1(\xi_1, \xi_2)\lambda_2(\xi_1, \xi_2)\,\hat{\chi}_{j+2}(\xi_3)$$

where $\hat{\chi}_{1,i}$'s are the bubbles associated to the first fist edge of the master triangle and ζ is the coordinate on such edge. Finally, the interior modes are simply products of bubble functions

$$\hat{\chi}^{\text{bub}}_{i,j} = \hat{\chi}^{\text{bub}}_i(\xi_1, \xi_2)\,\hat{\chi}_{j+2}(\xi_3)$$

where $\hat{\chi}^{\text{bub}}_i$ are the master triangle interior modes.

Transfinite interpolation on a prism. Transfinite interpolation on a prism follows the usual construction

$$\Phi^{\text{lin}} + \sum_{i=1}^{9} \Phi^{\text{edge}}_i + \sum_{j=1}^{5} \Phi^{\text{face}}_j$$

where the linear contribution is given by $\Phi^{\text{lin}} = \sum_{i=6} \hat{\chi}_i \mathbf{v}_i$. Mimicking the construction of shape functions, contributions due to horizontal edges involve kernel functions, while contributions due to vertical edges involve bubble functions; as usual, let's focus on a specific horizontal edge, $\mathrm{e}_1 = [\mathbf{v}_1, \mathbf{v}_2]$, and vertical edge, $\mathrm{e}_7 = [\mathbf{v}_1, \mathbf{v}_4]$. We have that

$$\Phi^{\text{edge}}_1 = \left[\frac{\boldsymbol{\alpha}_1(\zeta) - \hat{\chi}_1(\zeta)\mathbf{v}_1 - \hat{\chi}_2(\zeta)\mathbf{v}_2}{(1-\zeta)\zeta}\right]\lambda_1(\xi_1, \xi_2)\lambda_2(\xi_1, \xi_2)\hat{\chi}_1(\xi_3)$$

where the bracketed term is indeed the edge kernel and ζ is the coordinate along the first edge of the master triangle; construction for the vertical edge is as follows:

$$\Phi^{\text{edge}}_7 = \big[\boldsymbol{\alpha}_7(\xi_3) - \hat{\chi}_1(\xi_3)\mathbf{v}_1 - \hat{\chi}_2(\xi_3)\mathbf{v}_4\big]\lambda_1(\xi_1, \xi_2)$$

where the local edge bubble, the term in brackets, has replaced the edge kernel function. Let's now turn our attention to the face contributions; specifically, let's consider a horizontal face, $\mathrm{f}_4 = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]$, and the vertical face, $\mathrm{f}_1 = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_5, \mathbf{v}_4]$. The contribution from the horizontal face involves use of the local face bubble, *i.e.* the term between parenthesis in the following expression:

$$\begin{aligned}\Phi_1^{\text{face}} = \Big(\boldsymbol{\beta}_1(\xi_1,\xi_2) - \big[\lambda_1(\xi_1,\xi_2)\mathbf{v}_1 + \lambda_2(\xi_1,\xi_2)\mathbf{v}_2 + \lambda_3(\xi_1,\xi_2)\mathbf{v}_3\big] \\ - \Big\{\big[\boldsymbol{\alpha}_1(\zeta) - \hat{\chi}_1(\zeta)\mathbf{v}_1 - \hat{\chi}_2(\zeta)\mathbf{v}_2\big]\lambda_1(\xi_1,\xi_2)\lambda_2(\xi_1,\xi_2)\Big\} \\ - \Big\{\big[\boldsymbol{\alpha}_2(\zeta) - \hat{\chi}_1(\zeta)\mathbf{v}_2 - \hat{\chi}_2(\zeta)\mathbf{v}_3\big]\lambda_2(\xi_1,\xi_2)\lambda_3(\xi_1,\xi_2)\Big\} \\ - \Big\{\big[\boldsymbol{\alpha}_3(\zeta) - \hat{\chi}_1(\zeta)\mathbf{v}_1 - \hat{\chi}_2(\zeta)\mathbf{v}_3\big]\lambda_1(\xi_1,\xi_2)\lambda_3(\xi_1,\xi_2)\Big\}\Big)\hat{\chi}_1(\xi_3)\end{aligned}$$

Notice that the first bracketed term is the linear interpolation of the face and the terms in curly braces are the local edge bubbles; moreover, the three instances of ζ need to be interpreted as the local coordinate on the first, second and third edge of the master triangle respectively. Construction of the vertical face bubble requires the face kernel which is defined as:

$$\begin{aligned}\Phi_3^{\text{face ker}} = \Big(\boldsymbol{\beta}_3(\zeta,\xi_3) - \big[\hat{\chi}_1(\zeta)\hat{\chi}_1(\xi_3)\mathbf{v}_1 + \hat{\chi}_2(\zeta)\hat{\chi}_1(\xi_3)\mathbf{v}_2 + \hat{\chi}_2(\zeta)\hat{\chi}_2(\xi_3)\mathbf{v}_5 + \hat{\chi}_1(\zeta)\hat{\chi}_2(\xi_3)\mathbf{v}_4\big] \\ - \Big\{\big[\boldsymbol{\alpha}_1(\zeta) - \hat{\chi}_1(\zeta)\mathbf{v}_1 - \hat{\chi}_2(\zeta)\mathbf{v}_2\big]\hat{\chi}_1(\xi_3)\Big\} - \Big\{\big[\boldsymbol{\alpha}_8(\xi_3) - \hat{\chi}_1(\xi_3)\mathbf{v}_2 - \hat{\chi}_2(\xi_3)\mathbf{v}_5\big]\hat{\chi}_2(\zeta)\Big\} \\ - \Big\{\big[\boldsymbol{\alpha}_4(\zeta) - \hat{\chi}_1(\zeta)\mathbf{v}_4 - \hat{\chi}_2(\zeta)\mathbf{v}_5\big]\hat{\chi}_2(\xi_3)\Big\} - \Big\{\big[\boldsymbol{\alpha}_7(\xi_3) - \hat{\chi}_1(\xi_3)\mathbf{v}_1 - \hat{\chi}_2(\xi_3)\mathbf{v}_4\big]\hat{\chi}_1(\zeta)\Big\}\Big)/(1-\zeta)\zeta\end{aligned}$$

The first term in brackets is the bilinear interpolation of the face and the following four terms in curly braces are the linear interpolation of the edges; ζ is the coordinate along the first edge of the master triangle. The face bubble is obtained multiplying by the appropriate face blending function: $\Phi_3^{\text{face}} = \Phi_3^{\text{face ker}}\,\lambda_1(\xi_1,\xi_2)\lambda_2(\xi_1,\xi_2)$.

Master pyramid. The master pyramid is described by Fig. 8; as anticipated, the pyramid is needed as a connecting element in any mesh that includes tetrahedra and prisms. Construction of an appropriate functional space on the pyramid is a non trivial task; we follow the construction proposed by Zaglmayr, see [18], that relies on the Duffy's transformation. Such a transformation, defined as

$$\xi_1 = x_1(1-x_3) \quad ; \quad \xi_2 = x_2(1-x_3) \quad ; \quad \xi_3 = x_3$$

maps the master hexahedron to the master pyramid by collapsing the top face onto vertex $\mathbf{v}_5$. With this tool in place, an appropriate functional space for the pyramid is determined by applying the Duffy's transformation to the space $\mathcal{Q}^{p,q,r}$, supported by the hexahedron, and imposing appropriate constraints. It is convenient to apply the Duffy's transformation to monomials of the form $x_1^i x_2^j (1-x_3)^k$, where $0 \le i \le p$, $0 \le j \le q$, and $0 \le k \le r$, since:

$$x_1^i x_2^j (1-x_3)^k \mapsto \xi_1^i \xi_2^j (1-\xi_3)^{k-i-j}$$

Appropriate constraints are determined by requiring that the previous functions reduce to polynomials of type $\mathcal{P}$ on the lateral faces. In particular, on face $\xi_1 + \xi_3 = 1$ we obtain $\xi_2^j (1-\xi_3)^{k-j}$, hence $0 \leq j \leq k$, and on face $\xi_2 + \xi_3 = 1$ we have that $\xi_1^i (1-\xi_3)^{k-i}$, which implies $0 \leq i \leq k$. This shows that the pyramid is indeed an isotropic element, $p = q = r$, and leads and it supports the space spanned by the monomials $\xi_1^i \xi_2^j (1-\xi_3)^{k-i-j}$ under the condition $0 \leq i, j \leq k \leq p$. Notice that the restrictions to face $\xi_1 = 0$ and to face $\xi_2 = 0$ are indeed the desired ones, provided we set $i = 0$ and $j = 0$ respectively, and define $0^0 = 1$; moreover, the space supported by the pyramid contains the space of polynomials $\mathcal{P}^p$ and some rational functions. Vertex shape functions are defined as

$$\hat{\chi}_1 = \frac{(1-\xi_1-\xi_3)(1-\xi_2-\xi_3)}{1-\xi_3} \qquad \hat{\chi}_2 = \frac{\xi_1(1-\xi_2-\xi_3)}{1-\xi_3}$$
$$\hat{\chi}_3 = \frac{\xi_1\xi_2}{1-\xi_3} \qquad \hat{\chi}_4 = \frac{\xi_2(1-\xi_1-\xi_3)}{1-\xi_3}$$
$$\hat{\chi}_5 = \xi_3$$

Notice that first four vertex shape functions are indeed singular at the pyramid vertex, although in the limit we indeed recover $\hat{\chi}_i(\mathbf{v}_j) = \delta_{ij}$; moreover, the vertex shape functions are linear along the pyramid edges. In order to construct edge and face modes we must define some suitable projection operators. Projections on vertical edges $[\mathbf{v}_i, \mathbf{v}_j]$ are defined in terms of vertex shape functions as:

$$(\xi_1, \xi_2, \xi_3) \mapsto \zeta \qquad ; \qquad \zeta = \frac{\hat{\chi}_j - \hat{\chi}_i + 1}{2}$$

where the local coordinate ζ is set in the direction $[\mathbf{v}_i \rightarrow \mathbf{v}_j]$. Such a construction is indeed the projection parallel to the surface $[\hat{\chi}_j - \hat{\chi}_i + 1 - 2\zeta \equiv 0]$ where ζ can be interpreted as a parameter. A graph of such a surface for edge $[\mathbf{v}_1, \mathbf{v}_5]$ and $\zeta = 1/2$ is shown in Fig. 9; notice that the surface is made up of two bilinear sheets, however only the lower one intersects the edge of interest. Projections on the horizontal edges parallel to the ξ_1 axis and on the edges parallel to the ξ_2 axis are defined, respectively, as

$$(\xi_1, \xi_2, \xi_3) \mapsto \zeta \quad , \quad \zeta = \xi_1 + \frac{\xi_3}{2} \qquad ; \qquad (\xi_1, \xi_2, \xi_3) \mapsto \zeta \quad , \quad \zeta = \xi_2 + \frac{\xi_3}{2}$$

while projections on faces $[\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_5]$ and $[\mathbf{v}_4, \mathbf{v}_3, \mathbf{v}_5]$, and on faces $[\mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_5]$ and $[\mathbf{v}_1, \mathbf{v}_4, \mathbf{v}_5]$ are defined, respectively, as

$$(\xi_1, \xi_2, \xi_3) \mapsto (\zeta_1, \zeta_2) \quad ; \quad \zeta_1 = \xi_1 \ , \ \zeta_2 = \xi_3 \qquad ; \qquad (\xi_1, \xi_2, \xi_3) \mapsto (\zeta_1, \zeta_2) \quad ; \quad \zeta_1 = \xi_2 \ , \ \zeta_2 = \xi_3$$

Finally, the projections on the bottom face is defined as:

$$(\xi_1, \xi_2, \xi_3) \mapsto (\zeta_1, \zeta_2) \qquad ; \qquad \zeta_1 = \frac{\xi_1}{1-\xi_3} \quad , \quad \zeta_2 = \frac{\xi_2}{1-\xi_3}$$

For ease of exposition it is convenient to first define all the blending functions; the edge blending functions are defined as:

$$\Lambda_1 = \frac{\xi_1(1-\xi_1-\xi_3)(1-\xi_2-\xi_3)}{1-\xi_3} \qquad \Lambda_2 = \frac{\xi_1\xi_2(1-\xi_2-\xi_3)}{1-\xi_3}$$
$$\Lambda_3 = \frac{\xi_1\xi_2(1-\xi_1-\xi_3)}{1-\xi_3} \qquad \Lambda_4 = \frac{\xi_2(1-\xi_1-\xi_3)(1-\xi_2-\xi_3)}{1-\xi_3}$$
$$\Lambda_5 = \frac{\xi_1\xi_2(1-\xi_2-\xi_3)}{1-\xi_3} \qquad \Lambda_6 = \frac{\xi_1\xi_2\xi_3}{1-\xi_3}$$
$$\Lambda_7 = \frac{\xi_2\xi_3(1-\xi_1-\xi_3)}{1-\xi_3} \qquad \Lambda_8 = \frac{\xi_1(1-\xi_1-\xi_3)(1-\xi_2-\xi_3)}{1-\xi_3}$$

while the face blending functions are defined as

$$\Xi_1 = \xi_3\Lambda_1 = \frac{\xi_1\xi_3(1-\xi_1-\xi_3)(1-\xi_2-\xi_3)}{1-\xi_3} \qquad \Xi_2 = \xi_3\Lambda_2 = \frac{\xi_1\xi_2\xi_3(1-\xi_2-\xi_3)}{1-\xi_3}$$
$$\Xi_3 = \xi_3\Lambda_3 = \frac{\xi_1\xi_2\xi_3(1-\xi_1-\xi_3)}{1-\xi_3} \qquad \Xi_4 = \xi_3\Lambda_4 = \frac{\xi_2\xi_3(1-\xi_1-\xi_3)(1-\xi_2-\xi_3)}{1-\xi_3}$$
$$\Xi_5 = \hat{\chi}_1\hat{\chi}_3 = \frac{\xi_1\xi_2(1-\xi_1-\xi_3)(1-\xi_2-\xi_3)}{(1-\xi_3)^2}$$

Finally, the interior mode blending function is

$$\Psi = \xi_3\hat{\chi}_1\hat{\chi}_3 = \frac{\xi_1\xi_2\xi_3(1-\xi_1-\xi_3)(1-\xi_2-\xi_3)}{(1-\xi_3)^2}$$

Edge shape functions are computed employing the kernel functions; the edge modes associated to the i-th edge are

$$\hat{\chi}^{\text{edge}}_{i,j} = \hat{\chi}^{\text{ker}}_j(\zeta)\,\Lambda_i(\xi_1,\xi_2,\xi_3) \qquad \text{for } j = 1,\ldots,p-1$$

where ζ is the local coordinate along the first edge. Similarly, vertical face modes are computed by means of the triangle kernel functions; hence, for $i = 2,\ldots,5$, we have that

$$\hat{\chi}^{\text{face}}_{i,j} = \hat{\chi}^{\text{ker}}_j(\zeta_1,\zeta_2)\,\Xi_i(\xi_1,\xi_2,\xi_3) \qquad \text{for } j = 1,\ldots,\frac{(p+1)(p+2)}{2}-3$$

Construction of the bottom face modes and the interior modes is more complicated because we need to take into account the singularities of the blending functions. Let's start by considering the bottom face modes. Although the pyramid is an isotropic element, its bottom face is anisotropic of order (p,q) since it is a quad; the face modes are constructed in the following way:

$$\hat{\chi}^{\text{face}}_{1,i,j} = \hat{\chi}^{\text{ker}}_i(\zeta_1)\hat{\chi}^{\text{ker}}_j(\zeta_2)(1-\xi_3)^{\max\{i-1,j-1\}}\,\Xi_1(\xi_1,\xi_2,\xi_3) \qquad \text{for } i = 1,\ldots,p \quad ; \quad j = 1,\ldots,q$$

Let now p be the order of approximation associated to the pyramid; then the interior modes are constructed in a similar fashion as:

$$\hat{\chi}^{\text{bub}}_i = \hat{\chi}_i(\zeta_1)\hat{\chi}_j(\zeta_2)(1-\xi_3)^{k-1}\,\Psi(\xi_1,\xi_2,\xi_3) \qquad \text{for } i,j = 1,\ldots,k \quad ; \quad k = 1,\ldots,p-1$$

Transfinite interpolation on a pyramid. Using the pyramid blending functions we just constructed, transfinite interpolation on the pyramid turns out to be a rather easy task; as usual, the transfinite interpolation is expressed as

$$\Phi^{\text{lin}} + \sum_{i=1}^{8} \Phi_i^{\text{edge}} + \sum_{j=1}^{5} \Phi_j^{\text{face}}$$

where $\Phi^{\text{lin}} = \sum_{i=1}^{5} \hat{\chi}_1 \mathbf{v}_i$ is the linear interpolation. Construction of the edge bubbles relies on the kernel functions. For ease of discussion let's consider the first edge; we define

$$\Phi_1^{\text{edge}} = \left[\frac{\boldsymbol{\alpha}_1(\zeta) - \hat{\chi}_1(\zeta)\mathbf{v}_1 - \hat{\chi}_2(\zeta)\mathbf{v}_2}{(1-\zeta)\zeta} \right] \Lambda_1(\xi_1, \xi_2, \xi_3)$$

where ζ is the local coordinate along the first edge and the bracketed expression is indeed the edge kernel function. In order to construct lateral face bubbles we need to use the face kernel functions; if we focus our attention on the first lateral face, $\mathrm{f}_1 = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_5]$, then we have that

$$\begin{aligned}
\Phi_1^{\text{face ker}} = \Big(& \boldsymbol{\beta}_1(\zeta_1, \zeta_2) - \big[(1-\zeta_1-\zeta_2)\mathbf{v}_1 + \zeta_1\mathbf{v}_2 + \zeta_2\mathbf{v}_5\big] \\
& - \Big\{ \big[\boldsymbol{\alpha}_1(\zeta) - \hat{\chi}_1(\zeta)\mathbf{v}_1 - \hat{\chi}_2(\zeta)\mathbf{v}_2\big] \lambda_1(\zeta_1, \zeta_2)\lambda_2(\zeta_1, \zeta_2) \Big\} \\
& - \Big\{ \big[\boldsymbol{\alpha}_2(\zeta_2) - \hat{\chi}_1(\zeta)\mathbf{v}_2 - \hat{\chi}_2(\zeta)\mathbf{v}_5\big] \lambda_2(\zeta_1, \zeta_2)\lambda_3(\zeta_1, \zeta_2) \Big\} \\
& - \Big\{ \big[\boldsymbol{\alpha}_6(\zeta) - \hat{\chi}_1(\zeta)\mathbf{v}_1 - \hat{\chi}_2(\zeta)\mathbf{v}_5\big] \lambda_1(\zeta_1, \zeta_2)\lambda_3(\zeta_1, \zeta_2) \Big\} \Big) / (1-\zeta_1-\zeta_2)\zeta_1\zeta_2
\end{aligned}$$

where ζ_1, ζ_2 are the local coordinates on the face and $\lambda_1, \lambda_2, \lambda_3$ are the affine coordinates with respect to the master triangle. Notice that the first bracketed expression is the linear interpolation of the face, while the three terms in curly braces are local edge bubbles; moreover, the three instances of ζ need to be interpreted, respectively, as the local coordinate along the first, second, and third edge of the triangular face. The face bubble is obtained multiplying by the appropriate blending function: $\Phi_1^{\text{face}} = \Phi_1^{\text{face ker}}\, \Xi_1(\xi_1, \xi_2, \xi_3)$. Finally, in order to construct the bottom face bubble, we first define the face kernel function:

$$\begin{aligned}
\Phi_5^{\text{face ker}} = \Big(& \boldsymbol{\beta}_5(\zeta_1, \zeta_2) - \big[\hat{\chi}_1(\zeta_1)\hat{\chi}_1(\zeta_2)\mathbf{v}_1 + \hat{\chi}_2(\zeta_1)\hat{\chi}_1(\zeta_2)\mathbf{v}_2 + \hat{\chi}_2(\zeta_1)\hat{\chi}_2(\zeta_2)\mathbf{v}_3 + \hat{\chi}_1(\zeta_1)\hat{\chi}_2(\zeta_2)\mathbf{v}_4\big] \\
& - \Big\{ \big[\boldsymbol{\alpha}_1(\zeta_1) - \hat{\chi}_1(\zeta_1)\mathbf{v}_1 - \hat{\chi}_2(\zeta_1)\mathbf{v}_2\big] \hat{\chi}_1(\zeta_2) \Big\} \\
& - \Big\{ \big[\boldsymbol{\alpha}_2(\zeta_2) - \hat{\chi}_1(\zeta_2)\mathbf{v}_2 - \hat{\chi}_2(\zeta_2)\mathbf{v}_3\big] \hat{\chi}_2(\zeta_1) \Big\} \\
& \Big\{ \big[\boldsymbol{\alpha}_3(\zeta_1) - \hat{\chi}_1(\zeta_1)\mathbf{v}_3 - \hat{\chi}_2(\zeta_1)\mathbf{v}_4\big] \hat{\chi}_2(\zeta_2) \Big\} \\
& - \Big\{ \big[\boldsymbol{\alpha}_4(\zeta_2) - \hat{\chi}_1(\zeta_2)\mathbf{v}_4 - \hat{\chi}_2(\zeta_2)\mathbf{v}_1\big] \hat{\chi}_1(\zeta_1) \Big\} \Big) / \zeta_1\zeta_2(1-\zeta_1)(1-\zeta_2)
\end{aligned}$$

As in the previous expression, ζ_1, ζ_2 are the local coordinates on the face, the first bracketed expression is the bilinear interpolation of the face, and the four terms in curly braces are local edge bubbles; we finally have that $\Phi_5^{\text{face}} = \Phi_5^{\text{face ker}}\, \Xi_5(\xi_1, \xi_2, \xi_3)$.

Generalization to elements of variable order. In the previous discussion, the hexahedron has order of approximation (p, q, r), the tetrahedron and the pyramid have order of approximation p, since they are indeed isotropic elements, and the prism has order of approximation (p, q), where p is the order in the (ξ_1, ξ_2) plane and q in the order in the ξ_3 direction. As already pointed out in the discussion for the 2D elements, those are indeed the orders of approximation associated to the interior nodes; when elements are placed next to each other, an appropriate order of approximation needs to be determined for the common edges and faces. Things are rather trivial for the hexahedron: edges that are oriented in the ξ_1, ξ_2 and ξ_3 direction inherit order of approximation, respectively, equal to p, q, and r. Similarly, faces parallel to the (ξ_1, ξ_2) plane have order (p, q), faces parallel to the (ξ_1, ξ_3) plane have order (p, r), and faces parallel to the (ξ_2, ξ_3) plane have order (q, r). Since the tetrahedron is an isotropic element, all faces and edges inherit order of approximation p; similarly, all triangular faces of the pyramid and edges have order of approximation p, while the bottom face has order (p, p). Finally, the vertical faces of the prism have order of approximation (p, q), while the horizontal ones have order p; the vertical edges have order q, and the horizontal ones have order p. When two quadrilateral faces having order (p_i, q_i) are placed next to each other, the order of approximation associated to the middle node of the figure is again determined by the minimum rule $p = \min\{p_1, p_2\}$, $q = \min\{q_1, q_2\}$; similarly $p = \min p_1, p_2$ when two triangular faces of order p_1 and p_2 are glued together. Things are slightly more complicated for edges, since we need to take into account all figures that are attached to given edge. The edge will inherit from the attached faces orders of approximation p_i according to the minimum rule described for 2D element; clearly we choose $p = \min\{p_i\}$.

4 Implementation

Data structures. In our code elements are described by the following user defined prototype:

```
type element
  character(len=5)                          :: type
  integer, dimension(:), pointer            :: bcond
  integer, dimension(:), pointer            :: nodes
  integer                                   :: edge_orient
  integer                                   :: face_orient
  integer, dimension(:), pointer            :: neig
end type element
```

Let us briefly describe the geometrical variables we are interested in: `edge_orient` and `face_orient` are, respectively, the edge and face orientations and `neig` is a pointer to the neighboring elements. We keep track of the nodes pertaining to the element by means of the pointer `nodes`; vertex nodes are listed first, followed by mid-edge and mid-face nodes and, finally, by the interior node. The face and edge orientations of the elements in the initial mesh are directly inherited from the corresponding GMP blocks. Globally,

the nodes are ordered as interior (prisms, hexas, tets and pyramids), vertex (points), mid-edge (curves), and mid-face (rectangles, triangles). Finally, the nodes are described by the following prototype:

```
type node
  character(len=5)                                :: type
  integer                                         :: order
  integer                                         :: bcond
  double precision, dimension(:,:), pointer :: coord
  double precision, dimension(:,:), pointer :: zdofH
end type node
```

We describe only a few variables: `case` store the kind of physical attributes associated to the node, `order` is the order of approximation and, finally, `zdofH` is a pointer to the degrees of freedom associated to the node.

Verification of shape functions. Numerical verification of shape functions is essential for building a reliable FE code; more specifically, we need to check whether our routines indeed produce consistent orientations and whether we can reproduce polynomial solutions up to machine precision. In order to address the latter issue, we employ an L^2 and an H^1 projection onto the space of polynomials $\xi_1^i \xi_2^j \xi_3^k$ for $0 \leq i+j+k \leq p$:

$$L^2 \text{ projection:} \qquad \int_\Omega u_h \, v_h = \int_\Omega u \, v_h \qquad \forall v_h$$

$$H^2 \text{ projection:} \qquad \int_\Omega u_h \, v_h + \int_\Omega \nabla u_h \cdot \nabla v_h = \int_\Omega u \, v_h + \int_\Omega \nabla u \cdot \nabla v_h \qquad \forall v_h$$

We are indeed interested in the H^1 projection only; however, since the H^1 projection is made up of two contributions, it is convenient to first check the contribution due to the L^2 projection of the function, and then, if the test were successful, add the contribution due to the L^2 projection of the gradient. We anticipate that our shape functions have been verified up to order $p = 6$, since for higher orders our routines fail on the pyramid due to bad conditioning of our core sets of shape functions. In order to account for elements of all types and all possible adjacencies we constructed three different toys meshes of affine elements, see Figures 10, 11, 12: a single tetrahedron surrounded by four tetrahedra; a pyramid surrounded by four tetrahedra on the lateral faces, and a pyramid on the bottom face; and a prism surrounded by three prisms across the lateral faces and two tetrahedra across the horizontal faces. We used affine elements so that the functional spaces supported on the reference elements and the corresponding physical ones are indeed the same. Finally, the issue of consistency of orientations was addressed by rotating the surrounding elements in each mesh and comparing all solutions to the problem. More specifically, for every mesh, we identified the shared edges and faces; then we reversed the orientation of on edge at a time and compared the result to the previous solution. The same procedure was finally repeated for triangular and rectangular faces.

Mesh generation. Our code works with isoparametric elements, *i.e.* shape functions of the same order of the element are used to generate approximate geometry maps; we remark that if the exact geometry maps were used, then the bulk of the computations of the FE code would take place inside the geometry modeling package. In order to determine such maps, we need to generate geometry degrees of freedom; we call this procedure *mesh generation*. We determine geometry d.o.f.'s through Projection-Based Interpolation, see [1] and [2], whose guiding principles are locality, optimality and global continuity; by locality we imply that we want to project over one element only at a time, and by optimality we mean that we want to minimize the difference between the exact geometry map and the interpolation, according to some suitable norm. Without loss of generality we restrict our attention to only one component, call it u^{FE}, of the geometry map; let u be the corresponding component of the exact geometry map. The Projection-Based interpolation is determined by solving the following variational problem:

$$
\begin{aligned}
& u(\mathbf{v}_i) = u^{\mathrm{FE}}(\mathbf{v}_i) && \text{for each vertex } \mathbf{v}_i \in K \\
& \int_{\mathrm{e}} \left(\frac{du}{ds} - \frac{du^{\mathrm{FE}}}{ds} \right) \frac{d\chi_{\mathrm{e}}}{ds} \frac{ds}{d\zeta} = 0 && \text{for each edge bubble } \chi_{\mathrm{e}} \quad , \quad \text{for each edge } \mathrm{e} \in K \\
& \int_{\mathrm{f}} (\nabla u - \nabla u^{\mathrm{FE}}) \cdot \nabla \chi_{\mathrm{f}} = 0 && \text{for each face bubble } \chi_{\mathrm{f}} \quad , \quad \text{for each face } \mathrm{f} \in K \\
& \int_{K} (\nabla u - \nabla u^{\mathrm{FE}}) \cdot \nabla \chi^{\mathrm{bub}} = 0 && \text{for each bubble } \chi^{\mathrm{bub}}
\end{aligned}
$$

where χ's are the edge and face bubble functions and $ds/d\zeta$ can be regarded as a weight function. Verification of this interpolation procedure is done by checking p-convergence rates; in other words, we should observe that

$$\|\boldsymbol{u} - \boldsymbol{u}^{\mathrm{FE}}\|_{H^1} \to 0$$

as $p \downarrow 0$.

A numerical example. The described shape functions have been employed in solving a simplified model for the study of propagation of high intensity acoustic waves inside the human head; this problem falls in the class of acoustics coupled with elasticity. More specifically, we are interested in the study of bone conduction of sounds and its effects on the middle ear, with emphasis on the damages it might cause to an organ called cochlea. Figures 13 and 15 describes the pressure distribution on the simplified model of the middle ear, while Fig. 14 describes the real part of the displacement; we remark that this model is built with tetrahedra, prisms, and pyramids. The cylinder representing the ossicles is made of tetrahedra; the elements inside the three membranes are prisms while the ones on the brim are either tetrahedra or pyramids.

5 Conclusions

The starting point for the application of the FE method to a problem is the definition of a domain of interest; a convenient way to achieve this objective is through a MBG description of the domain which relies on maps

of the master elements onto the physical elements. We have discussed how those maps can be constructed thorough a technique called transfinite interpolation; moreover, we have considered the full family of 3D elements: hexahedron, tetrahedron, prism and pyramid. We have reviewed the well known construction of shape functions on the quad and the hexahedron and have given a literature overview on the subject, including some possible extensions to 3D elements. In the main section of the paper we have proposed a bottom up approach for the construction of shape functions on the master tetrahedron, prism and pyramid. A set of 1D shape functions was extended inside the element in order to obtain edge modes and, on triangular faces, a set of 2D shape functions was extended inside the element to create face modes; on rectangular faces we extended tensor products of 1D shape functions. We were brought to this approach because we think it is the most logically coherent: this construction parallels transfinite interpolation and, because of its bottom up nature, the sets of 1D and 2D shape functions can be replaced at any moment to better accommodate for the application we are dealing with. Finally, we have quickly described the main challenges that we faced during implementation of the shape functions in our hp FE code and we discuss how we have dealt with orientations and how we have verified our new routines; we have included some picture and data from an ongoing acoustics project we are currently working on.

It is necessary to point out that our construction conflicts with fast integration, while this is not the case for constructions that rely on the Duffy's transformation, such as those proposed by Zaglmayr, Karniadakis and Sherwin. Yet, we feel like the logical cleanness of our construction and the parallel with transfinite interpolation is a remarkable achievement; it is also worth recalling that the Duffy's transformation is edge and face depending, hence constructions that rely on it need to take into account different transformations on different geometrical entities. When facing the challenge of implementation, this fact leads to more complicated routines. Also, we are very satisfied with our new approach of taking edge and face orientations into account at the level of the element shape functions routines, since it dramatically reduces the complexity of the assembly procedure; this is indeed a major improvement over our previous logic of implementation. We have not addressed yet possible extensions of our construction to the other spaces of the exact sequence, *i.e.* $H(\mathrm{div})$ and $H(\mathrm{curl})$; we recall that the construction of Zaglmayr indeed extends to those spaces. This will the be the purpose of future work.

References

[1] L. Demkowicz, *Computing with hp-adaptive Finite Elements*, vol. 1, Chapman and Hall 2007

[2] L. Demkowicz *et. al.*, *Computing with hp-adaptive Finite Elements*, vol. 2, Chapman and Hall 2008

[3] A. Düster, H Bröker, E. Rank, *The p-version of the finite element method for three-dimensional curved thin walled structures*, Int. J. Numer. Eng. 52:673-703, 2001

[4] W.J. Gordon, C.A. Hall, *Transfinite element methods: Blending function interpolation over arbitrary curved element domain*, Numer. Math. 21:109-129, 1973

[5] B. Szabó, I. Babuška, *Finite Element Analysis*, Wiley 1991

[6] B. Szabó, A. Düster, E. Rank, *The p-version of the finite element method*, Encyclopedia of Computational Mechanics, 1:119-139, Wiley 2004

[7] G.E. Karniadakis, S.J. Sherwin. *Spectral/hp Element Methods for CFD*, Numerical Mathematics and Scientic Computation. Oxford University Press, New York, Oxford, 1999.

[8] M. Dubiner, *Spectral methods on triangles and other domains*, Journal of Scientic Computing, 6(4):345390, 1991.

[9] P. Devloo et al. *Systematic and generic construction of shape functions for p-adaptive meshes of multidimensional finite elements*, Computer Methods in Applied Mechanics and Engineering, Volume 198, Issues 21-26, Advances in Simulation-Based Engineering Sciences - Honoring J. Tinsley Oden, 1 May 2009, Pages 1716-1725

[10] S. Owen, Meshing research corner, `http://www.andrew.cmu.edu/user/sowen/mesh.html`

[11] S. Owen, *A survey of unstructured mesh generation technology*, `http://www.andrew.cmu.edu/user/sowen/survey`, 2007.

[12] S. J. Owen and M. S. Shepherd, editors. *Special Issue on Trends in Unstructured Mesh Generation*, volume 20. Engineering With Computers, September 2004.

[13] S. J. Owen and D. R. White. *Mesh-based geometry*, Int. J. Num. Meth. Eng., (2):375395, July 2003.

[14] P. Solin et al. *Higher-Order Finite Element Methods*, Chapman and Hall, July 2003

[15] S. Zaglmayr, *High Order Finite Element Methods for Electromagnetic Field Computation*, Ph.D Thesis, Johannes Kepler University Linz, 2006

[16] L. Demkowicz, P. Gatto, W. Qiu, A. Joplin G^1*-Interpolation and geometry reconstruction for higher order finite elements*, Computer Methods in Applied Mechanics and Engineering, Vol. 198, Issue 13-14, pp. 1198 - 1212, 2009

[17] N. Nigam, J. Phillips, *Higher-order finite elements on pyramids*, 2007

[18] S. Zaglmayr, *Higher-order exact sequence for pyramid*, in preparation

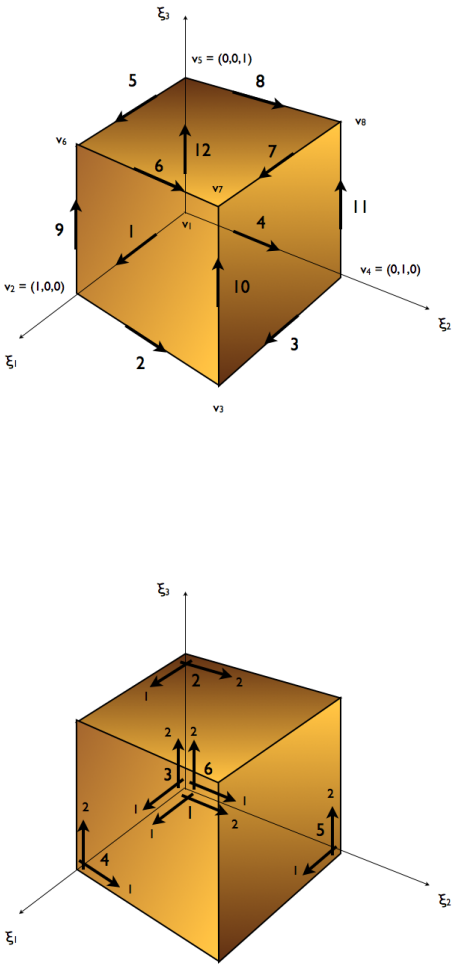

Figure 5: Master hexahedron.

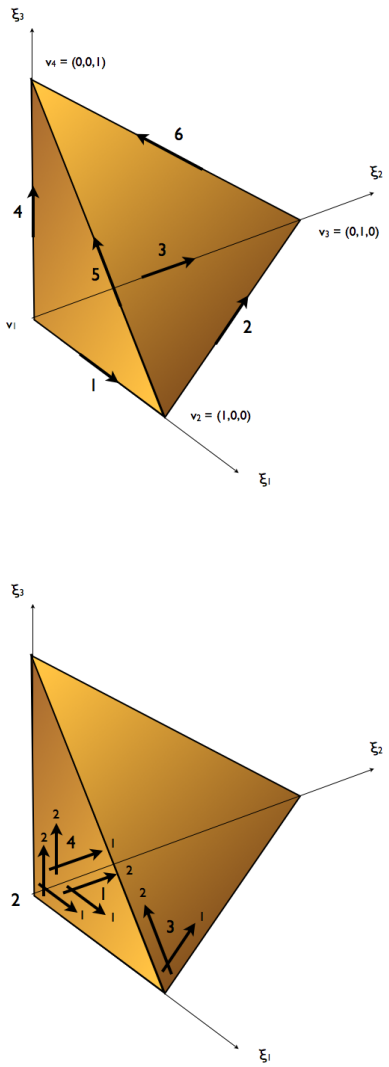

Figure 6: Master tetrahedron.

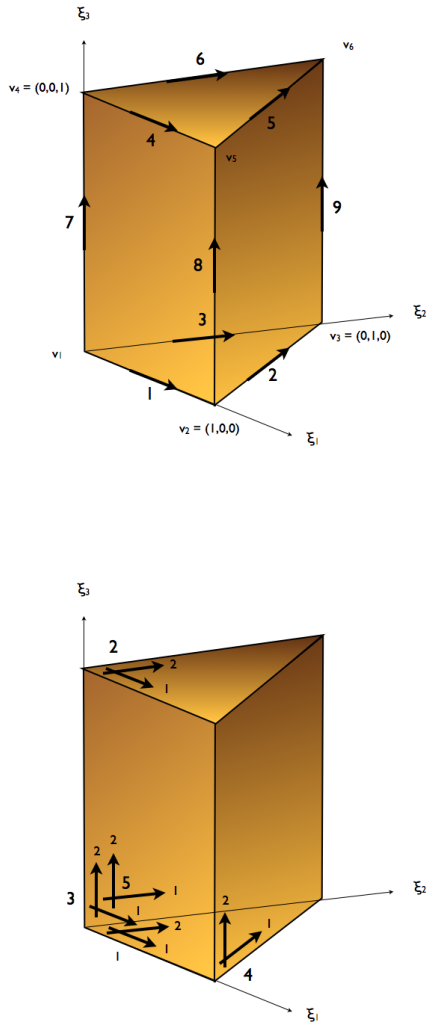

Figure 7: Master prism.

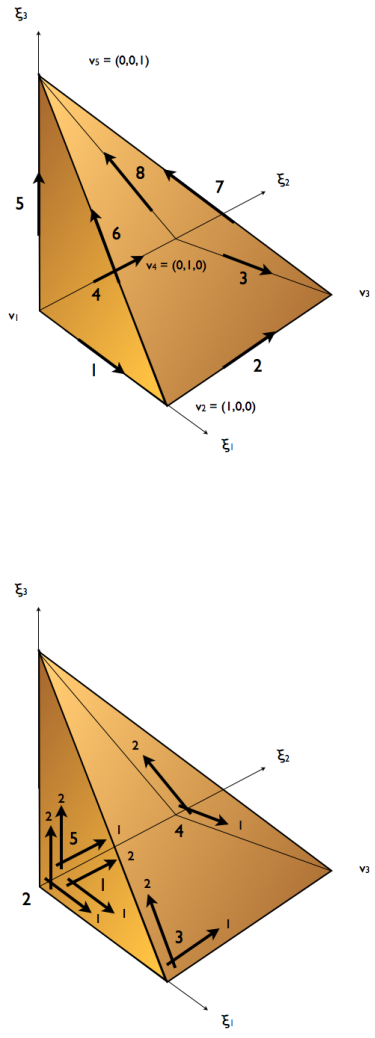

Figure 8: Master pyramid.

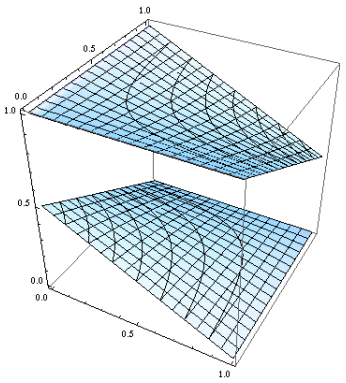

Figure 9: Graph of surface $[\hat{\chi}_5 - \hat{\chi}_1 \equiv 0]$.

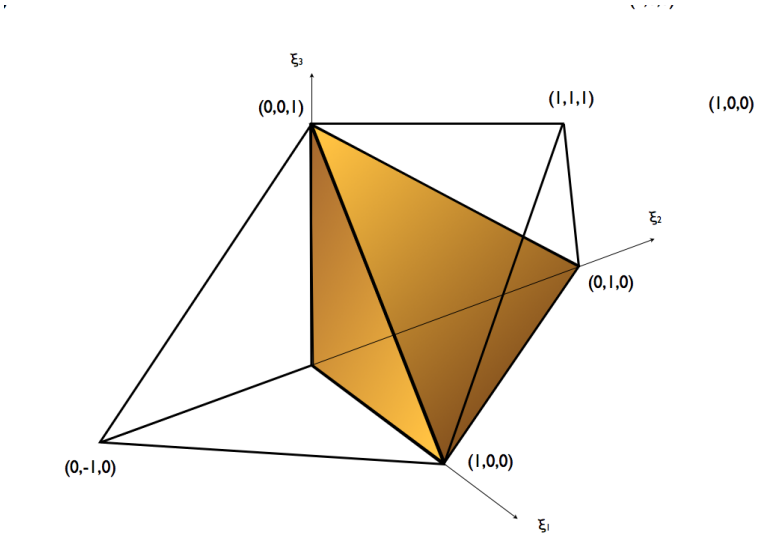

Figure 10: First toy mesh: a central tetrahedron is surrounded by four tetrahedra; for clarity only two surrounding tetrahedra are shown.

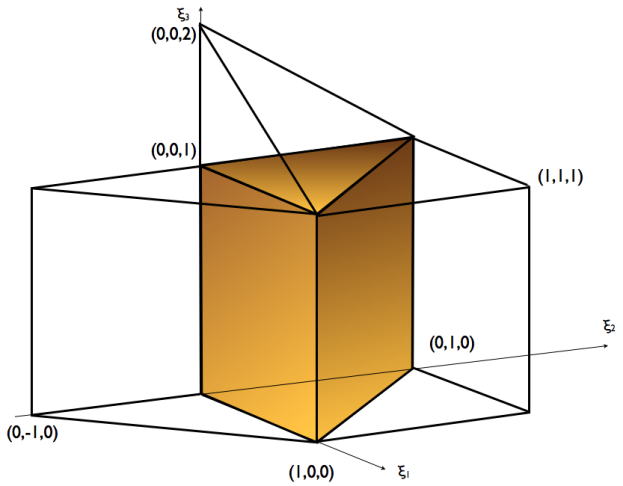

Figure 11: Second toy mesh: a central prism is surrounded by three prisms and two tetrahedra; for clarity only two surrounding prism and one tetrahedra are shown.

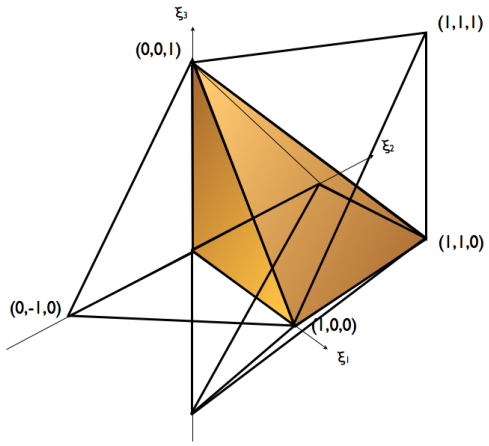

Figure 12: Third toy mesh: a central pyramid is surrounded by four tetrahedra and a pyramid; for clarity only two surrounding tetrahedra and the pyramid are shown.

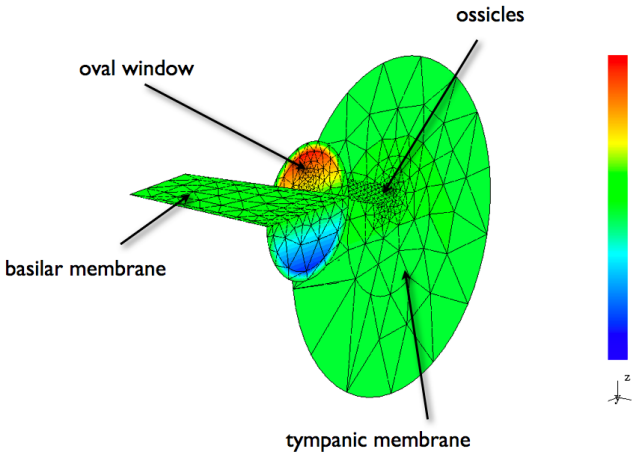

Figure 13: Distribution of real part of pressure on the middle ear.

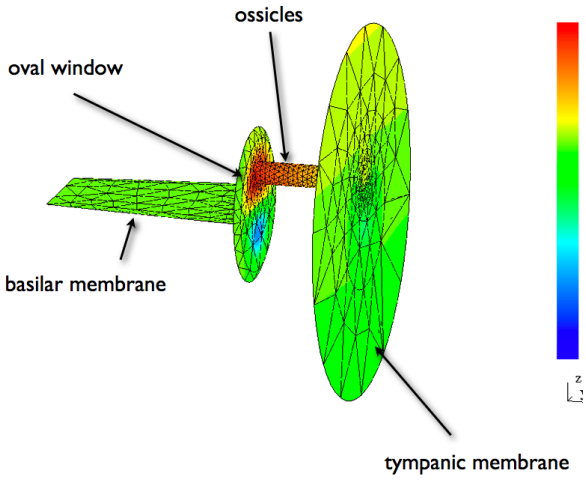

Figure 14: Distribution of real part of displacement on the middle ear.

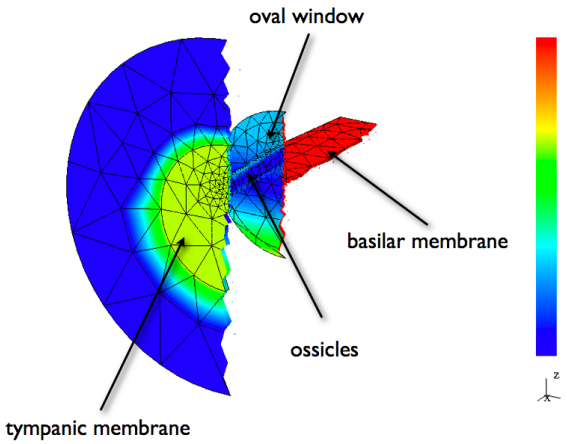

Figure 15: Distribution of absolute value of pressure on the middle ear.