Dolomites Research Notes on Approximation

Special Issue SA2022 – *Software for Approximation 2022*, Volume 15 · 2022 · Pages 1–12

Computing weights for high order Whitney edge elements*

Ludovico Bruni Bruno[a] · Ana Alonso Rodríguez[a] · Francesca Rapetti[b]

Abstract

The interpolation of differential forms is a challenging problem that is getting increasing attention. The issue of finding unisolvent degrees of freedom to describe a differential form in terms of high-order Whitney forms is an active area of research nowadays. In this paper we deal with a family of such degrees of freedom, called weights, that fits with the physical and geometrical nature of the field to interpolate. These weights play the role of interpolation coefficients when reconstructing scalar/vector fields in terms of a set of selected multivariate polynomial forms. Weights are a generalization of the evaluations of a scalar function at a set of nodes in view of its reconstruction on multivariate polynomial bases. As in the nodal case, different sets of such weights are compared in terms of a Lebesgue constant. In this contribution, we briefly recall their definition and provide examples of algorithms in low dimension to compute their associated Lebesgue constant value. Insights to greater dimensions are offered as well.

1 Introduction

High order finite elements (FEs) methods combine the geometrical flexibility, namely the possibility of using a mesh to describe the domain, with some basic facts from polynomial approximation theory which allow to construct a good interpolant. These facts concern the polynomial bases which are used to express the interpolant, the sets of points which are chosen to compute the coefficients appearing in the expression of the interpolant and high order quadrature formula which are necessary to preserve accuracy when computing integrals. To get an idea on how these different numerical aspects play a crucial role in multivariate polynomial approximations, we refer to, *e.g.*, [19], [15], [9], [11], [6], [13], [26] and the references therein. In this work, we wish to look at the cited basic facts when the fields to be interpolated are intended as differential k-forms [18, 5] (with $0 \leq k \leq n$ and n the space dimension). Polynomial bases become high order Whitney forms, namely appropriate subspaces of trimmed high order polynomial differential forms. The coefficients of the interpolant are suitable degrees of freedom (moments [21] or weights [24]) of the field on sets of geometric objects belonging to the mesh which represents the physical domain. To compute moments or weights, we may involve high order quadrature formula and we rely on the known literature, see *e.g.* [13].

High order weights are a physically motivated alternative [7] when working in electromagnetism or CFD. A weight is the integral of a k-form on a k-simplex. This allows to keep the intuitive meaning of these degrees of freedom as evaluations at points ($k = 0$), circulations along edges ($k = 1$), fluxes across faces ($k = 2$) and, generally, densities in volumes ($k = n$), regardless of the polynomial degree r. Weights thus reflect the nature of the fields they are associated with and preserve for any approximation degree $r \geq 0$ the meaning of degrees of freedom as cochains [12], in the spirit of Whitney original work [28] dating back to 1957.

As nodal evaluations of a scalar field are done at the nodes of suitably selected sets in the mesh elements, similarly, weights of a field, intended as a k-form, are computed on suitably selected sets of small simplices of dimension k (nodes for $k = 0$, edges if $k = 1$, faces for $k = 2$, volumes for $k = n$) [24]. We thus introduce for $k > 0$ notions such as the Vandermonde matrix, the Lebesgue constant, which are used in classical polynomial interpolation for $k = 0$ to evaluate the quality of the approximation.

We recall that, for scalar fields w, weights coincides with classical nodal degrees of freedom, namely the values of w at a set of points in the computational domain. A lot of work in the literature has been devoted to study how these points have to be chosen (see for example [22, 17, 27]). A guiding aspect consists in reducing the Lebesgue constant in order to expect for a more reliable interpolated. This concept has recently been exteded to small simplices [3]. It is thus possible to imagine to consider different sets of small k-simplices whose vertices are associated with different sets of interpolations points as explained in [1] and thus to compare different families of weights by such a *generalized Lebesgue constant*. For weights associated with small simplices a direct proof of unisolvence has been given [12] in the case of uniform distributions. For the generation of non uniform distributions of small simplices, we refer to [2].

Polynomial differential forms and operators acting on them are straightforward implementable, we refer to [14] for the interested reader. In this work we present algorithms that allow to compute the generalized Lebesgue constant associated with 1-simplices and thus compare different choices of weights for Whitney 1-forms. In particular, in Section 2 we recall definitions and develop the machinery needed to perform such computation. Section 3 is devoted to a formal construction of small simplices and a technique for moving towards different families is offered. We present an explicit example of this construction considering *warp and blend nodes* for the availability of explicit routines. Weights and associated quantities are introduced. A generic pseudo-algorithm, which can be applied for any degree r and k-forms in $\mathbb{R}^n$, is provided. Few Matlab programs in the cases $k = 0$

*The preface of this special issue to which the article belongs is given in [10].

[a]Università degli Studi di Trento, Loc. Povo, Trento (IT)
[b]Université Côte d'Azur, Parc Valrose, Nice (FR)

and $k=1$ for $n=1$ are presented in Section 4. The software provided for $n=1$ and $k=1$ is a model to write the algorithm for the other cases, as shown in Section 5. Finally, some numerical results are offered in Section 6.

2 Polynomial differential forms on simplices

The role of the spaces of polynomial differential forms in numerical analysis has been improving its relevance in the last fifteen years [5]. We recall basic concepts about the spaces $\mathcal{P}_r^-\Lambda^k(T)$ and address the reader to [4] for more details.

Let $\mathrm{Alt}^k(\mathbb{R}^n)$ be the vector space of multi-linear alternating maps $(\mathbb{R}^n)^k \to \mathbb{R}$. We denote with $\mathcal{P}_r(\mathbb{R}^n)$ and $\mathcal{H}_r(\mathbb{R}^n)$ the spaces of polynomials and homogeneous polynomials in n variables respectively. Spaces of differential k-forms on $\mathbb{R}^n$ are denoted by $\Lambda^k(\mathbb{R}^n)$. We put the spaces $\mathcal{P}_r\Lambda^k(\mathbb{R}^n)$ and $\mathcal{H}_r\Lambda^k(\mathbb{R}^n)$ of, respectively, complete and homogeneous polynomial differential k-forms of degree r, as

$$\mathcal{P}_r\Lambda^k(\mathbb{R}^n) \doteq \mathcal{P}_r(\mathbb{R}^n)\otimes \mathrm{Alt}^k(\mathbb{R}^n), \qquad \mathcal{H}_r\Lambda^k(\mathbb{R}^n) \doteq \mathcal{H}_r(\mathbb{R}^n)\otimes \mathrm{Alt}^k(\mathbb{R}^n). \tag{1}$$

They are both finite dimensional subspaces of $\Lambda^k(\mathbb{R}^n)$. Letting d denote the exterior derivative, the space of *trimmed polynomial differential forms* is the unique subspace $\mathcal{P}_r^-\Lambda^k(\mathbb{R}^n)$ of $\mathcal{P}_r\Lambda^k(\mathbb{R}^n)$ satisfying (see [5])

$$\mathcal{P}_r\Lambda^k(\mathbb{R}^n) = \mathcal{P}_r^-\Lambda^k(\mathbb{R}^n) \oplus \mathrm{d}\mathcal{H}_{r+1}\Lambda^{k-1}(\mathbb{R}^n). \tag{2}$$

Note that $\mathcal{P}_r^-\Lambda^0(\mathbb{R}^n) = \mathcal{P}_r\Lambda^0(\mathbb{R}^n) = \mathcal{P}_r(\mathbb{R}^n)$. Let $T\subset\mathbb{R}^n$ be an n-simplex and denote its vertices by $[\boldsymbol{x}_0,\dots,\boldsymbol{x}_n]$. We will extensively use the following known facts about simplices:

- A k-subsimplex $F_\sigma \doteq [\boldsymbol{x}_{\sigma(0)},\dots,\boldsymbol{x}_{\sigma(k)}]$ of T is a simplex generated by only $k+1$ vertices of T. We call *edge* a simplex generated by two vertices and face a simplex generated by three vertices. We denote the collection of k-subsimplices of T by $\Delta_k(T)$.
- We denote by $\mathfrak{S}_n^k$ the set of increasing permutations $\sigma : \{0,\dots,k\} \to \{0,\dots,n\}$. We may associate a k-subsimplex $F_\sigma \in \Delta_k(T)$ with the permutation $\sigma\in\mathfrak{S}_n^k$ that indexes its vertices. We consequently denote by $F_\sigma \setminus \sigma(i)$ the $(k-1)$-simplex generated by all the vertices of F_σ with the exception of $\sigma(i)$.
- Simplices may be formally combined to obtain *chains* [20], i.e. finite formal linear combinations of simplices with real coefficients. The most natural example of chain is the *boundary* ∂T of T, which is not a simplex itself but the collection of $(k-1)$-simplices $T\setminus\sigma(i)$, $i=0,\dots,n$, with a $\pm$ sign induced by the orientation. For example, if $T=[\boldsymbol{x}_0,\boldsymbol{x}_1,\boldsymbol{x}_2]$, then $\partial T = [\boldsymbol{x}_1,\boldsymbol{x}_2]-[\boldsymbol{x}_0,\boldsymbol{x}_2]+[\boldsymbol{x}_0,\boldsymbol{x}_1]$, which is a 1-chain on the edges $\Delta_1(T)$ of T.
- A simplicial complex $\mathbb{X}$ is a collection of simplices with the property that any two simplices of $\mathbb{X}$ are either disjoint or they intersect in a simplex of $\mathbb{X}$. The collection of all subsimplices of T has thus a structure of simplicial complex.

Any point of T can be defined in terms of its *barycentric coordinates* $\{\lambda_0,\dots,\lambda_n\}$ with respect to the vertices of T, namely

$$T = \left\{ \boldsymbol{x}\in\mathbb{R}^n,\ \ \boldsymbol{x} = \sum_{i=0}^n \lambda_i \boldsymbol{x}_i, \quad \text{with} \quad 0\le\lambda_i\le 1 \quad \text{and} \quad \sum_{i=0}^n \lambda_i = 1 \right\}.$$

Polynomial differential forms $\omega\in\mathcal{P}_r\Lambda^k(\mathbb{R}^n)$ may be written as $\omega = \sum_{\sigma\in\mathfrak{S}_n^k} p(x_{\sigma(0)},\dots,x_{\sigma(k)})\mathrm{d}x_\sigma$ with $p\in\mathcal{P}_r(\mathbb{R}^n)$, being $\mathrm{d}x_\sigma$ the volume form, namely $\mathrm{d}x_{\sigma(1)}\wedge\ldots\wedge\mathrm{d}x_{\sigma(k)}$ such that $\int_{[\boldsymbol{x}_{\sigma(0)},\dots,\boldsymbol{x}_{\sigma(k)}]}\mathrm{d}x_\sigma = vol([\boldsymbol{x}_{\sigma(0)},\dots,\boldsymbol{x}_{\sigma(k)}]) = 1/k!$. Spaces $\mathcal{P}_r\Lambda^k(T)$ and $\mathcal{P}_r^-\Lambda^k(T)$ are obtained by restricting from $\mathbb{R}^n$ to T. One has (see [12])

$$\dim\mathcal{P}_r^-\Lambda^k(T) = \binom{r+k-1}{k}\binom{n+r}{n-k}.$$

A basis for $\mathcal{P}_r^-\Lambda^k(T)$ is given by the following rule. Recall that any $F_\sigma\in\Delta_k(T)$ is associated with an increasing permutation $\sigma:\{0,\dots,k\}\to\{0,\dots,n\}$. For each $F_\sigma\in\Delta_k(T)$ define the associated *Whitney* k-form $\omega^{F_\sigma}\in\Lambda^k(T)$ as

$$\omega^{F_\sigma} \doteq k!\sum_{i=0}^k (-1)^i \lambda_{\sigma(i)}\mathrm{d}\lambda_{\sigma(0)}\wedge\ldots\wedge\widehat{\mathrm{d}\lambda_{\sigma(i)}}\wedge\ldots\wedge\mathrm{d}\lambda_{\sigma(k)},$$

where the hat means that the underlying term is omitted from the expression. A basis for $\mathcal{P}_1^-\Lambda^k(T)$ is $\left\{\omega^{F_\sigma},\ F_\sigma\in\Delta_k(T)\right\}$, as shown in [28]. Note that for $k=0$, if F_σ is the vertex $\boldsymbol{x}_i$, then ω^{F_σ} is just the barycentric coordinate λ_i. The case $0<k<n$ is less evident (see [21], [5]). Denoting by ω^{F_σ} the Whitney k-form associated with the k-simplex F_σ and by $\mathbf{w}_{F_\sigma}$ the corresponding vector (referred to as proxy) field, we have for example:

- If $k=1$ and $F_\sigma=[\mathbf{x}_i,\mathbf{x}_j]$, then
$$\omega^{F_\sigma} = \lambda_i\mathrm{d}\lambda_j - \lambda_j\mathrm{d}\lambda_i, \quad \mathbf{w}_{F_\sigma} = \lambda_i\nabla\lambda_j - \lambda_j\nabla\lambda_i.$$
- If $k=2$ and $F_\sigma=[\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k]$, then
$$\begin{aligned}\omega^{F_\sigma} &= 2(\lambda_i\mathrm{d}\lambda_j\wedge\lambda_k - \lambda_j\mathrm{d}\lambda_i\wedge\lambda_k + \lambda_k\mathrm{d}\lambda_i\wedge\lambda_j),\\ \mathbf{w}_{F_\sigma} &= 2(\lambda_i\nabla\lambda_j\times\nabla\lambda_k - \lambda_j\nabla\lambda_i\times\nabla\lambda_k + \lambda_k\nabla\lambda_i\times\nabla\lambda_j).\end{aligned}$$

Whitney forms of lower polynomial degree $r=1$ can be defined by means of the following recursive formula.

Lemma 2.1. *If F_σ is a k-simplex of T with $k \geq 1$, then*

$$\omega^{F_\sigma} = \sum_{i=0}^{k} (-1)^i \lambda_{\sigma(i)} \mathrm{d}\omega^{F_\sigma \setminus \sigma(i)}.$$

Proof. A direct computation (see [12]) shows that $\mathrm{d}\omega^{F_\sigma} = (k+1)!\, \mathrm{d}\lambda_{\sigma(0)} \wedge \ldots \wedge \mathrm{d}\lambda_{\sigma(k)}$, and this yields the equality $\sum_{i=0}^{k} (-1)^i \lambda_{\sigma(i)} \mathrm{d}\omega^{F_\sigma \setminus \sigma(i)} = \sum_{i=0}^{k} (-1)^i \lambda_{\sigma(i)} k! \mathrm{d}\lambda_\sigma(0) \wedge \ldots \wedge \widehat{\mathrm{d}\lambda_{\sigma(i)}} \wedge \ldots \wedge \mathrm{d}\lambda_{\sigma(k)} = \omega^{F_\sigma}$. □

Now, let $\boldsymbol{\lambda}^{\boldsymbol{\alpha}}$ denote the product $\lambda_0^{\alpha_0} \cdot \ldots \cdot \lambda_n^{\alpha_n}$ with $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_n) \in \mathcal{I}(n+1, r-1)$, the set of multi-indices $(\alpha_0, \ldots, \alpha_n)$ of $n+1$ integers $\alpha_i \geq 0$ such that $|\boldsymbol{\alpha}| \doteq \sum_{i=0}^{n} \alpha_i = r-1$. Whitney k-forms of higher polynomial degree $r > 1$ are simply obtained by considering products of $\boldsymbol{\lambda}^{\boldsymbol{\alpha}}$, homogeneous polynomials of degree $(r-1)$ in the barycentric coordinates λ_i, with Whitney k-forms of polynomial degree 1. Note that redundancies may occur, as proved in [24], [12]. Indeed, we have the following result.

Lemma 2.2. *Let F_σ be a $(k+1)$-simplex. Then we have the following relation among Whitney k-forms:*

$$\sum_{i=0}^{k+1} (-1)^i \lambda_i \omega^{F_\sigma \setminus \sigma(i)} = 0.$$

As an illustration of Lemma 2.2, if F_σ is a 2-simplex, it can be verified by a direct computation that $\lambda_0 \omega^{[\mathbf{x}_1, \mathbf{x}_2]} - \lambda_1 \omega^{[\mathbf{x}_0, \mathbf{x}_2]} + \lambda_2 \omega^{[\mathbf{x}_0, \mathbf{x}_1]} = 0$. Let us consider, for each increasing permutation $\sigma : \{0, \ldots, k\} \to \{0, \ldots, n\}$, the subset

$$\mathcal{I}_\sigma(n+1, r-1) \doteq \{\boldsymbol{\alpha} \in \mathcal{I}(n+1, r-1) \mid \alpha_i = 0 \ \forall i < \sigma(0)\}.$$

One then has

$$\mathcal{P}_r^- \Lambda^k(T) = \mathrm{span}\ \left\{\boldsymbol{\lambda}^{\boldsymbol{\alpha}} \omega^{F_\sigma} \mid F_\sigma \in \Delta_k(T),\ \boldsymbol{\alpha} \in \mathcal{I}_\sigma(n+1, r-1)\right\}. \tag{3}$$

The restriction $i < \sigma(0)$ in the definition of $\mathcal{I}_\sigma(n+1, r-1)$ is imposed to avoid, in the set $\boldsymbol{\lambda}^{\boldsymbol{\alpha}} \omega^S$ generating $\mathcal{P}_r^- \Lambda^k(T)$, the redundancies predicted by Lemma 2.2. For a detailed construction we address the reader to [8]. Before going to the next section, we recall that spaces of polynomial differential forms are endowed with a norm, called *0-norm* [16], which reads the features of the support they are defined on. Such a norm reads

$$\|\omega\|_0 \doteq \sup_{c \in \mathcal{C}_k(T)} \frac{1}{|c|_0} \left| \int_c \omega \right|, \tag{4}$$

where $\mathcal{C}_k(T)$ denotes the space of all k-chains supported in T (i.e. the formal linear combinations of simplices supported in T and not only of the k-subsimplices of T) and $|c|_0$ the k-th Hausdorff measure of c. For $k = 0$, we have $\|.\|_0 = \|.\|_\infty$.

3 Small simplices and generalized Lebesgue constant

In the following, we are interested in the space $\mathcal{P}_r^- \Lambda^k(\mathbb{R}^n)$ that allows for reconstructions of polynomial degree r of regular fields with minimal information. In this section we present the so-called *weights*, namely degrees of freedom introduced in [24] for $\omega \in \mathcal{P}_r^- \Lambda^k(T)$, in an n-simplex T, which will be involved in such as reconstruction.

One possible choice of degrees of freedom for fields in $\mathcal{P}_r \Lambda^0(T)$ are evaluations on the points of the principal lattice of top-dimesional simplices T. More precisely, for a k-simplex T let $L_r(T)$

$$L_r(T) \doteq \left\{ \boldsymbol{x} \in T \mid \lambda_i(\boldsymbol{x}) \in \left\{0, \frac{1}{r}, \ldots, \frac{r-1}{r}, 1\right\} \text{ for } i = 0, \ldots, k \right\}, \tag{5}$$

be the set of points of the principal lattice of order r of T (see Figure 1, left). Then the spaces $\{\omega \mapsto \omega(s) \mid s \in L_r(T)\}$ are a unisolvent set of degreed of freedom for $\mathcal{P}_r \Lambda^0(T)$. The natural way of extending these degrees of freedom to the cases $k = 1, \ldots, n$ is to consider integrals on k-cells topologically contained in T. These degrees of freedom are called *physical* since they have a natural physical interpretation: e.g. for $k = 1$ they are the work done by a force (a 1-form) along a line (a 1-cell).

Definition 3.1. Let $\omega \in \mathcal{P}_r^- \Lambda^k(T)$ and let $s \subset T$ be a k-simplex. The *weight* of ω on s is the scalar quantity $\int_s \omega$.

The term "weight" is chosen in relation with the double role of Whitney forms. These k-forms are shape functions that reconstruct fields in a simplex T from their degrees of freedom associated with objects in $\Delta_k(T)$. Furthermore, the same k-forms allow to express any manifold of dimension k contained in T as weighted sum of objects in $\Delta_k(T)$, see [23]. This generalizes what is known for $k = 0$. Whitney 0-forms on a simplex T are the barycentric functions $\lambda_i : T \to [0, 1]$ associated with the nodes in $\Delta_0(T)$. Barycentric functions are the shape functions that allow to linearly reconstruct scalar fields from their nodal values at the vertices of T. Furthermore, any point $x \in T$ is the weighted sum of the vertices of T with coefficients given by $\lambda_i(x)$, $i = 0, \ldots, \dim(T)$.

Assume now that, for each simplex T, for each k, we have a *finite* set of k-cells $\{s_1, s_2, \ldots\}$ contained in T and such that distinct cells have disjoint interiors. To use these k-cells to supports weights, we need them to be unisolvent, in the sense of the following definition.

Definition 3.2. A family of small k-cells $\{s_1, \ldots, s_N\}$ is said to be *unisolvent* for $\mathcal{P}_r^- \Lambda^k(T)$ if $N = \dim \mathcal{P}_r^- \Lambda^k(T)$ and for all $\omega \in \mathcal{P}_r^- \Lambda^k(T)$ we have

$$\int_{s_i} \omega = 0 \quad \forall i = 1, \ldots, N \qquad \Longrightarrow \qquad \omega = 0.$$

An example of such k-cells are the small k-simplices introduced in [24]. For any $\xi \in \mathbb{R}^n$, consider then the map

$$\beta : \quad x \mapsto \beta_\xi(x) = \frac{1}{r}(x - x_0) + \xi,$$

which allows to define the *small simplices* as follows.

Definition 3.3. Let T be an n-simplex and $F \in \Delta_k(T)$. The set of small k-simplices of order r of T is

$$X_r^k(T) \doteq \left\{\beta_\xi(F) \mid F \in \Delta_k(T),\ \xi \in Z_r(T)\right\}. \tag{6}$$

An easy dimension count shows that

$$\#X_r^k(T) \geq \dim \mathcal{P}_r^-\Lambda^k(T),$$

where equality holds only for $k = 0$, $k = n$ or $r = 1$. Spaces $X_r^k(T)$ bear the following feature, as proved in [12].

Proposition 3.1. *Let $\omega \in \mathcal{P}_r^-\Lambda^k(T)$. If*

$$\int_s \omega = 0 \quad \forall s \in X_r^k(T),$$

then $\omega = 0$.

An algorithm for extracting N unisolvent simplices from $X_r^k(T)$ was proposed in [1] following the enumeration given for extracting a basis of $\mathcal{P}_r^-\Lambda^k(T)$ from its system of generators. Observe that we may associate any $s \in X_r^k(T)$ with a pair $(F_\sigma, \boldsymbol{\alpha})$ through the mapping

$$(F_\sigma, \boldsymbol{\alpha}) \mapsto s = \frac{1}{r}\left(F_\sigma + [x_0|\dots|x_n]\boldsymbol{\alpha}^T\right).$$

By direct computation one sees that any $s \in X_r^k(T)$ can be written in such a way, constructing then a bijection between $X_r^k(T)$ and $\Delta_k(T) \times \mathcal{I}(n+1, r-1)$. However, if one considers the subset associated with $\Delta_k(T) \times \mathcal{I}_\sigma(n+1, r-1)$, which is

$$X_{r,min}^k(T) \doteq \left\{\beta_\xi(F) \mid F \in \Delta_k(T),\ \xi = [x_0|\dots|x_n]\boldsymbol{\alpha},\ \boldsymbol{\alpha} \in \mathcal{I}(n+1, r-1)\right\}. \tag{7}$$

In Figure 1 an example of the set $X_{r,min}^k(T)$ is represented for $r = 3$ and $T \subset \mathbb{R}^n$. It is evident that $X_{r,min}^k(T) = X_r^k(T)$ when $k = 0$ and $k = n$, whereas unisolvent 1-simplices are obtained by discarding horizontal inner edges.

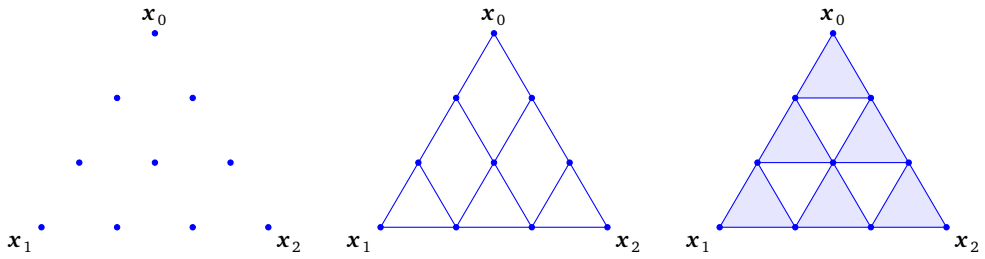

Figure 1: Left to right: a depiction of the sets $X_{3,min}^0(T)$, $X_{3,min}^1(T)$ and $X_{3,min}^2(T)$, being T a 2-simplex.

One has the following result, which was claimed in [1] and proved in [8].

Theorem 3.2. *Let $\omega \in \mathcal{P}_r^-\Lambda^k(T)$. If*

$$\int_s \omega = 0 \quad \forall s \in X_{r,min}^k(T),$$

then $\omega = 0$. Moreover,

$$\#X_{r,min}^k(T) = \dim \mathcal{P}_r^-\Lambda^k(T).$$

Definition 3.4. Fixed any basis $\mathcal{B} \doteq \{\omega_1, \dots, \omega_N\}$ for $\mathcal{P}_r^-\Lambda^k(T)$ and any set of k-simplices $S_r^k(T) = \{s_1, \dots, s_N\}$, we define the associated *generalized Vandermonde matrix* V as the matrix whose (i, j)-th entry is

$$V_{i,j} \doteq \int_{s_i} \omega_j. \tag{8}$$

Constructing the matrix V as in (8), one sees that a collection of k-simplices $S_r^k(T)$ is unisolvent for $\mathcal{P}_r^-\Lambda^k(T)$ if and only if the associated generalised Vandermonde matrix is not singular. This latter property does not depend on the choice of the basis for $\mathcal{P}_r^-\Lambda^k(T)$ but the condition number of V. In the following we rely on the Bernstein polynomial basis (see, e.g., [19]) to construct high order Whitney forms. This Bernstein basis is classically used in discrete exterior calculus because of its properties with respect to derivation (after derivation we obtain again a Bernstein polynomial) and it is expressed in terms of barycentric coordinates.

We in fact follow [8] and construct different families of simplices as follows. The space $X^k_{r,min}(T)$ has a structure of simplicial complex [20]. We thus deform it by means of a *simplicial isomorphism* φ, which is, roughly speaking, a mapping that pushes vertices of $X^k_{r,min}(T)$ to other points preserving adjacencies. To be more definite, we ask the image of $X^k_{r,min}(T)$ under φ to be a simplicial complex such that $\{\varphi(v_{\sigma(0)}),\dots,\varphi(v_{\sigma(k)})\}$ form a small k-simplex of the new family if and only if $\{v_{\sigma(0)},\dots,v_{\sigma(k)}\}$ form a small k-simplex in $X^k_{r,min}(T)$. As an example of this construction, we considered the *warp and blend* deformation proposed by Warburton in [17]. We thus considered simplices whose vertices do not belong anymore to $L_r(T)$ but to the set of warp and blend nodes to construct new sets of small k-simplices starting from $X^k_r(T)$. A three-dimensional depiction is offered in Figure 2: on the left we show the simplicial complex $X^k_{r,min}(T)$ and on the right its image under the simplicial isomorphism induced by the vertex maps that sends points of the principal lattice $L_r(T)$ to warp and blend nodes. We stress that this choice is driven by the computability of the set of warp and blend nodes for an arbitrary degree r, see [17]. The associated Vandermonde matrix resulted invertible for any checked $r \geq 0$.

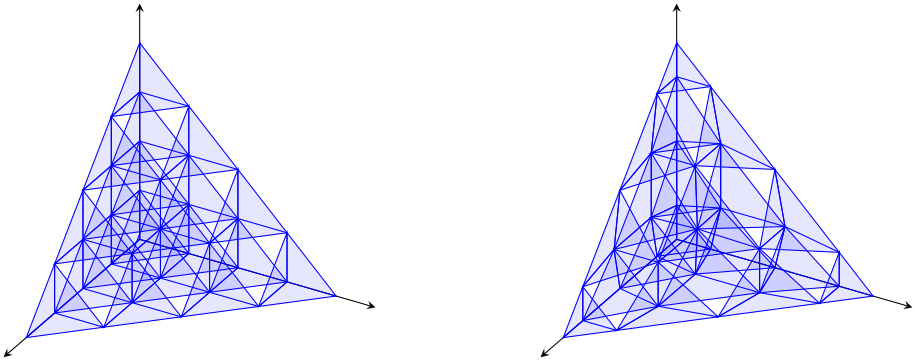

Figure 2: The action of the simplicial isomorphism induced by the warping and blending deformation.

In the following, we sketch the set of functions which yield an estimation of the Lebesgue constant. The description is detailed for $n = 1$ but needs to be completed by the user for $n > 1$.

The general procedure we adopt for constructing a set of edges from the uniform one is the following. We enumerate vertices of $X^0_r(T)$ by a lexicographical order, which means that $(x_1,\dots,x_n) > (y_1,\dots,y_n)$ if $x_1 > y_1$ or $x_i = y_i$ for $i = 1,\dots,\ell$ and $x_{\ell+1} > y_{\ell+1}$. We construct the connectivity matrix $A \in \mathbb{Z}^{N\times(k+1)}$ associated with $X^k_r(T)$, which will be the same used for constructing the set $S^k_r(T)$. In details, the i-th row of the matrix A contains the global indices of the $(k+1)$ vertices of the simplex s_i following the orientation induced by T.

Step 1: Construction of the simplicial complex

---

compute in T the set of uniformly distributed nodes $X^0_r(T)$
save the connectivity matrix A of $X^k_{r,min}(T)$
apply the simplicial isomorphism φ to $X^0_r(T)$
construct $S^k_r(T) \doteq \varphi\left(X^k_{r,min}(T)\right)$ using A and $\varphi(X^0_r(T))$
return A and $S^k_r(T)$

The following pseudo-algorithm shows how to compute the generalized Vandermonde matrix, with no restriction on the degree r, of the order of the form k and in any dimension n.

Step 2: Computation of the generalized Vandermonde matrix

---

fix a basis $\mathcal{B} \doteq \{\omega_1,\dots,\omega_N\}$ for $\mathcal{P}^-_r\Lambda^k(T)$
choose a set of k-simplices $S^k_r(T) = \{s_1,\dots,s_N\}$
compute $V_{\ell,j} = \int_{s_\ell} \omega_j$ by a simplicial quadrature rule for $\ell, j = 1,\dots,N$

Once the generalized Vandermonde matrix is computed, if it is invertible, one may define the so called cardinal basis for $\mathcal{P}^-_r\Lambda^k(T)$, which is in duality with weights in the sense that

$$\int_{s_i} \mathtt{w}_j = \delta_{i,j},$$

being the right hand side the Kronecker delta. In other words $\mathtt{w}_j = \sum_{\ell=1}^{N} (V^{-1})_{\ell j}\ \omega_\ell$, being V^{-1} the inverse of the generalised Vandermonde matrix given in Definition 3.4. In contrast with the case $k = 0$, the explicit expression of the cardinal basis functions when $k > 1$ relies on V^{-t}. For this reason, in what follows, we shall discuss a basis for $\mathcal{P}_r^-\Lambda^k(T)$ which is characterised by a limited growth of the conditioning of V. The cardinal basis extends the concept of Lagrange basis, which was defined only for polynomials, i.e. for the space $\mathcal{P}_r^-\Lambda^0(T)$. Such an object is crucial as it yields the *generalized Lebesgue constant* [3].

Definition 3.5. Let $S_r^k(T) = \{s_1, \dots, s_N\}$ be a unisolvent and minimal collection of small simplices for $\mathcal{P}_r^-\Lambda^k(T)$. The generalized Lebesgue constant associated with $S_r^k(T)$ is

$$\Lambda_r \doteq \sup_{c \in C_k(T)} \frac{1}{|c|_0} \sum_{j=1}^{N} |s_j|_0 \left| \int_c \mathtt{w}_j \right|. \tag{9}$$

For $k = 0$, the quantity Λ_r defined in (9) reduces to the classical Lebesgue constant $\Lambda_r = \sup_{\mathbf{x} \in T} \sum_{j=1}^{N} |\mathtt{w}_j(\mathbf{x})|$.

To estimate Λ_r we consider a test set $\mathcal{M}$ of M k-simplices supported in T and a quadrature rule and proceed in the following way. The larger the collection of k-simplices $\mathcal{M}$, the more reliable the computed value Λ_r. Note that, as soon as $\mathcal{M}$ is fixed and finite, it is sufficient to compute Λ_r on simplices and not on chains anymore. Let us define

$$W_{i,j} \doteq \int_{c_i} \omega_j, \quad c_i \in \mathcal{M},\ \omega_j \in \mathcal{B}. \tag{10}$$

We construct the diagonal matrices S and C

$$S_{i,j} \doteq \begin{cases} |s_i|_0 \text{ if } i = j \\ 0 \text{ otherwise} \end{cases} \qquad \text{and} \qquad C_{i,j} \doteq \begin{cases} |c_i|_0^{-1} \text{ if } i = j \\ 0 \text{ otherwise,} \end{cases}$$

where $|s|_0$ denotes the measure of the simplex s, see [16]. Hence one obtains that

$$\Lambda_r \approx \|CWV^{-1}S\|_\infty. \tag{11}$$

Note that this does not depend on the dimension n of the ambient space nor on the order of the form k.

Step 3: The numerical estimation of the Lebesgue constant

---

choose a set $\mathcal{M}$ of test k-simplices
compute the $N \times N$ diagonal matrix S such that $S_{i,i} = |s_i|_0$
compute the $M \times M$ diagonal matrix C such that $C_{i,i} = |c_i|_0^{-1}$, for $c_i \in \mathcal{M}$
compute $W_{\ell,j} = \int_{c_\ell} \omega_j$ by a simplicial quadrature rule for $\ell = 1, \dots, M$ and $j = 1, \dots, N$
compute Λ_r as $\Lambda_r = \|CWV^{-1}S\|_\infty$.

The importance of controlling such a quantity is explained by the next result. Let $S_r^k(T)$ be a unisolvent set of small k-simplices. Consider the associated cardinal basis. It is then immediate to construct the interpolator

$$\Pi : \quad \Lambda^k(T) \to \mathcal{P}_r^-\Lambda^k(T)$$
$$\eta \mapsto \Pi\eta \doteq \sum_{i=1}^{N} \left(\int_{s_i} \eta \right) \mathtt{w}_i.$$

The Lebesgue constant bounds the quality of this interpolation, in an appropriate norm. In fact, one has (see [3])

$$\|\eta - \Pi\eta\|_0 \le (1 + \Lambda_r)\|\eta - \Pi\eta^*\|_0,$$

being $\Pi\eta^*$ the best fit approximation, that is, the element of $\mathcal{P}_r^-\Lambda^k(T)$ which minimizes the quantity $\|\eta - \Pi\eta\|_0$. In the next sections, we aim at presenting a Matlab sofware implementing the proposed pseudo-codes. Numerical results obtained for $k = 0, 1$ in any dimension $n = 1, 2, 3$ can be found in [2]. They will be exemplified in Section 6.

4 Starting with $n = 1$

We show some algorithms that allow to compute the generalized Vandermonde matrix and hence the generalized Lebesgue constant for different choices of points in the unit segment $I = [0, 1]$. The reason for starting with $n = 1$ is twofold. On the one side, it is easier to underline the changes in the software for $n = 1$ when passing from $k = 0$ to $k = 1$. On the other hand, the modifications to adapt the software for $n = 1$ to higher dimension $n = 2$ or 3 are only a few, as we are going to explain in the next section. In what follows we consider some Matlab functions. Among these functions, the following are intrinsic:

- `nchoosek(n,k)` is the binomial $\binom{n}{k}$;
- `linspace(x,y,n)` provides a vector of n equispaced points between x and y, extremal points included;

- `length(v)` returns the number of elements of the vector `v`;
- `norm(A,p)` computes the p-norm of the matrix A. If $p = \text{inf}$ it computes the infinity norm;
- `vecnorm(A)` computes the 2-norm of each column of the matrix A.
- `abs(x)` computes the absolute value of the number `x`. If the argument is a vector, it works component-wise;
- `diag(v)` creates a diagonal matrix whose elements are that of the vector `v`;

and some others have been implemented:

- `gll(n)` provides a vector of $n+1$ Gauss-Lobatto-Legendre points (GLL for short) between -1 and 1, extremal points included. Recall that the i-th GLL point is the i-th zero of $P'_{r-1}(x)$, being $P_r(x)$ the Legendre polynomial of degree r. Such a routine is taken from [17], and moves Chebyshev points towards GLL ones by performing a Newton-Raphson approximation. Nodes are then sorted in an increasing order. We refer to the warp and blend procedure since we are using directly the functions in [17] that provides the simplicial isomorphism. This non uniform distribution of nodes is just an example (see [26] for an overview on existing such distributions). The procedure to estimate the Lebesgue constants is completely independent of the choice of the set of small edges;
- `evalBern(x,r)` evaluates all the Bernstein basis functions of $\mathcal{P}_r(T)$ at x (Algorithm 2);
- `integrateBern(a,b,r)` computes the integral of all the Bernstein basis functions of $\mathcal{P}_r(T)$ on $[a,b]$ (Algorithm 5);
- `QuadGaussLeg(r)` offers the Gauss-Legendre quadrature rule of degree r. It returns a triple containing the number of quadrature nodes, the coordinates of quadrature nodes in $[-1,1]$ and the corresponding weights (from [17]);
- `connect(Nint)` constructs the small edge-to-small node connectivity matrix of the $Nint$ small edges (Algorithm 6).

The first Algorithm constructs the small simplices and the test mesh. They play the role, respectively, of $X^0_{r,min}(T)$ and $\mathcal{C}_0(T)$ introduced in Section 2.

In the following algorithms the test mesh $\mathcal{M}$ is represented by the array TMesh.

Algorithm 1 Initialization

Ensure: The test mesh TMesh = (x) and the interpolation mesh $\mathcal{T}$ = (xequi) or $\mathcal{T}$ = (xgll) in the interval $[0,1]$.

```
1: M = input('number of points in the test mesh: ');          ▷ Test mesh size
2: x = linspace(0, 1, M);                                     ▷ Test mesh points
3: r = input('degree of the polynomial interpolation over the interval: ');   ▷ Polynomial interpolation degree
4: N = r+1;                                                   ▷ Number of interpolation mesh points
5: xequi = linspace(0, 1, N);                                 ▷ Uniform interpolation mesh points
6: xgll = (-gll(r)'+1)/2;                                     ▷ Gauss-Lobatto-Legendre interpolation mesh points
```

The sets of points introduced in Algorithm 1 are used also in the case $k = 1$ as extremities of the small edges to consider, as we will explain.

4.1 Nodal case $k = 0$

To begin with, we implement the case of scalar functions when interpolated by polynomials ($k = 0$). Algorithm 3 describes the construction of the (generalized) Vandermonde matrix with respect to the Bernstein basis. Such a basis is well suited as it reduces the conditioning of the Vandermonde matrix. Its elements are

$$b_{i,r}(x) = \binom{r}{i} x^i (1-x)^{r-i}, \quad \text{for } i = 0,\dots,r. \tag{12}$$

Note that, in one dimension, Bernstein basis functions are obtained by introducing a binomial coefficient in front of the monomial basis functions $\boldsymbol{\lambda}^{\boldsymbol{\alpha}}$, since $\lambda_1 = x$, $\lambda_0 = 1-x$ and $\boldsymbol{\alpha} = (\alpha_0, \alpha_1) = (r-i, i)$ with $i = 0,\dots,r$.

Algorithm 2 the $N = r+1$ Bernstein polynomial basis functions in dimension $n = 1$ evaluated at x

Ensure: The vector bernk0 containing the values of the N Bernstein polynomials (of degree r) at x.

```
1: function bernk0 = evalBern(x, r)
2:  N = r+1;                                   ▷ Number of Bernstein basis functions of P_r(R)
3:  bernk0 = zeros(N,1);
4:  for i = 1: N
5:      coef = nchoosek(r, i-1);               ▷ Bernstein coefficients
6:      bernk0(i) = coef*x^(i-1)*(1-x)^(N-i);
7:  end
8: end
```

Algorithm 4 implements the approximation of the Lebesgue constant.

In Algorithm 3, lines 7 and 8 are simplified since for a node $\boldsymbol{x}$ we have $|\boldsymbol{x}|_0 = 1$. Algorithm 5 is the analogous of Algorithm 3 and we need to take into account correctly the norm $|s|_0$ for any small or support edge s.

Algorithm 3 Construction of the Vandermonde matrix for $k = 0$ on the interpolation mesh

Require: The interpolation meshes $\mathcal{T} =$ (xequi) and $\mathcal{T} =$ (xgll) of N points.
Ensure: The Vandermonde matrix V (called VE for uniform and VG for GLL points) on the Bernstein basis in $[0, 1]$.

```
function [B, C] = Vandk0(xequi, xgll)
N = length(xequi); r = N-1;
VE = zeros(N); VG = zeros(N);                ▷ Vandermonde matrix at uniform and GLL interpolation nodes
for i = 1 : N
    bernk0E = evalBern(xequi(i), r); bernk0G = evalBern(xgll(i), r);
    VE(i,:) = bernk0E'; VG(i,:) = bernk0G';
end
condk0E = cond(VE); condk0G = cond(VG);                ▷ Conditioning of VE and VG
B = VE \ eye(N);                                       ▷ Invert and transpose VE and VG
C = VG \ eye(N);                ▷ eye is the identity matrix and \ is the direct system solution command
end
```

Algorithm 4 Evaluation of the Lebesgue constant for $k = 0$ on the test mesh

Require: The test mesh TMesh = (x) of M points and the matrix V^{-1} (called B for uniform and C for GLL interpolation).
Ensure: The Lebesgue constant (called LebConstE0 for uniform and LebConstG0 for GLL points).

```
function [LebConstE0, LebConstG0]= Lebk0(x, B, C)
M = length(x); N = size(B,1); r = N-1;
W0 = zeros(M, N);                ▷ Vandermonde matrix W0 at the test mesh points
for i = 1 : M
    bernk0 = evalBern(x(i), r);
    W0(i,:) = bernk0';
end
LebFunctE0 = W0*B;                ▷ Uniform node Lebesgue function at the M test mesh points
LebFunctG0 = W0*C;                ▷ GLL node Lebesgue function at the M test mesh points
LebConstE0 = norm(LebFunctE0, inf)                ▷ Lebesgue constant for uniform interpolation
LebConstG0 = norm(LebFunctG0, inf)                ▷ Lebesgue constant for GLL interpolation
end
```

4.2 Edge case $k = 1$

Algorithm 7 creates the generalized Vandermonde matrix associated with a collection of edges supported in $[0, 1]$. A description of the extension of this algorithm to higher dimensions is given in Section 5.

Algorithm 5 The integral of the $N = r + 1$ Bernstein polynomial basis functions in dimension $n = 1$ on $[a, b]$

Ensure: The vector bernk1 containing the integrals of the N Bernstein polynomials (of degree r) on $[a, b]$.

```
function bernk1 = integrateBern(a, b, r)
N = r+1;
[nng, sg, wg] = QuadGaussLeg(N);                ▷ GLL quadrature nodes and weights in [−1, 1]
c = (b-a)/2; d = (a+b)/2;
xqd = c*sg + d; wqd = c*wg;                ▷ Transport GLL quadrature nodes and weights on [a, b]
bernk0 = zeros(N,1); bernk1 = bernk0;
for i = 1: nng
    bernk0 = evalBern(xqd(i), r);
    bernk1(:) = bernk1(:) + wqd(i)*bernk0(:);
end
end
```

It is immediate to modify Algorithm 7 to obtain a rectangular matrix W as defined in (10). Algorithm 8 computes the Lebesgue constant for $k = 1$ provided a test mesh $\mathcal{M}$ and the generalized Vandermonde matrix associated with the Bernstein basis for $\mathcal{P}_r^-\Lambda^1([a, b])$. Since we aim to estimate such a quantity by using a finite test mesh, we may write it in terms of matrix calculus as in the nodal case $k = 0$. We assume that the test mesh $\mathcal{M}$ is passed as a matrix TMesh whose i-th column contains the endpoints of the i-th edge. For instance, when $n = 1$, TMesh(:, i) $= [p_0\ p_1]^T$ represents the segment $[p_0, p_1]$ contained in $[a, b]$.

Algorithm 6 Small edge to small node connectivity for $n = 1$

Ensure: The array Ed2Pt(2, Nint).

```
1: function Ed2Pt = connect(Nint)
2: Ed2Pt = zeros(2,Nint);
3: Ed2Pt(1, :) = [1:Nint];
4: Ed2Pt(2, :) = [2:Nint+1];
5: end
```

Algorithm 7 Construction of the Vandermonde matrix for $k = 1$ on the interpolation mesh

Require: The interpolation meshes $\mathcal{T} =$ (xequi) and $\mathcal{T} =$ (xgll) and their connectivity Ed2Pt.
Ensure: The Vandermonde matrix V (called VEe for uniform and VGe for GLL edges) on the Bernstein basis in $[0, 1]$.

```
1: function [A, D] = Vandk1(xequi, xgll, Ed2Pt)
2: N = length(xequi); r = N-1;
3: Nint = N-1;                                   ▷ Number of edges in the interpolation mesh
4: VEe = zeros(Nint); VGe = zeros(Nint);         ▷ Vandermonde matrix at uniform and GLL edges
5: for i = 1 : Nint
6:     e1 = Ed2Pt(1,i); e2 = Ed2Pt(2,i);
7:     a = xequi(e1); b = xequi(e2); bernk1E = integrateBern(a, b, r-1);
8:     a = xgll(e1); b = xgll(e2); bernk1G = integrateBern(a, b, r-1);
9:     VEe(i,:) = bernk1E'; VGe(i,:) = bernk1G';
10: end
11: condk1E = cond(VEe); condk1G = cond(VGe);    ▷ Conditioning of VEe and VGe
12: A = VEe \ eye(Nint);                         ▷ Invert and transpose VEe and VGe
13: D = VGe \ eye(Nint);                         ▷ eye is the identity matrix and \ is the direct system solution command
14: end
```

Algorithm 8 Evaluation of the Lebesgue constant for $k = 1$ on the test mesh

Require: The test mesh TMesh = (x), (inverse of the) Vandermonde matrices V^{-1} (called A for uniform and D for GLL interpolation edges), vertices of small simplices xequi and xgll and their connectivity Ed2Pt.
Ensure: The Lebesgue constant (called LebConstE1 for uniform and LebConstG1 for GLL edges).

```
1: function [LebConstE1, LebConstG1] = Lebk1(xequi, xgll, A, D, Ed2Pt, TMesh)
2: M = size(TMesh,2); N = size(A,1);
3: vSe = abs(xequi(Ed2Pt(2,:))-xequi(Ed2Pt(1,:)));   ▷ Vector containing lengths of equispaced small edges
4: vSg = abs(xgll(Ed2Pt(2,:))-xgll(Ed2Pt(1,:)));     ▷ Vector containing lengths of gll small edges
5: Se = diag(vSe);
6: Sg = diag(vSg);
7: for i=1:M
8:     bernk1 = integrateBern(TMesh(1,i), TMesh(2,i), N-1);
9:     W1(i,:) = bernk1';
10: end
11: vC = 1./abs(TMesh(2,:)-TMesh(1,:));
12: C = diag(vC) ;
13: LebConstE1 = norm(C*W1*A*Se, inf)             ▷ Lebesgue constant for uniform edge interpolation
14: LebConstG1 = norm(C*W1*D*Sg, inf)             ▷ Lebesgue constant for GLL edge interpolation
15: end
```

5 Making for $n > 1$

In higher spatial dimension, the Bernstein basis (12) is extended to Whitney forms by means of barycentric coordinates:

$$b_\alpha(\lambda) = \binom{r-1}{\alpha_0 \cdot \ldots \cdot \alpha_n} \lambda^\alpha \omega^{F_\sigma}. \tag{13}$$

Few changes have to be done in the proposed algorithms to stick with this generalisation, in particular:

- For what concerns the initialization, the test mesh of the reference element T can be computed using a mesh generator. Lines 6 and 7 in Algorithm 1 have to be replaced by two functions computing the coordinates of the points of the principal lattice of order r of T, namely the set $X_r^0(T)$, and the coordinates of the points of the set $S_r^k(T)$. For the non uniform distribution of nodes, we can adopt any among those proposed in the literature (see [17, 27]).

- The connectivity matrix `Ed2Pt` of the small edges has to be re-implemented in the case $n > 1$ in Algorithm 6. Its definition depends on the numbering of the nodes at the generation step. We keep the same name `Ed2Pt` for the connectivity matrix. We keep either *xequi* (for equidistributed) or *xgll* (for nonuniform), in accordance with the case $n = 1$, for the sets of vertices of small simplices.
- Algorithm 2 is replaced with its greater dimensional counterpart, giving Bernstein basis functions in n variables and degree $r-1$. The remaining degree to reach r comes from equation (3).
- A new version of Algorithm 5 for higher dimensions has to be written. In fact, if ω^{F_σ} is the Whitney 1-form associated with $F_\sigma \in \Delta_1(T)$ and e is an edge supported in the simplex T, we have

$$\int_e \omega^{F_\sigma} = |e|_0(\omega^{F_\sigma}(b_e)\cdot \boldsymbol{t}_e),$$

where b_e denotes the barycenter of e and $\boldsymbol{t}_e$ the unit vector tangent to e. Recalling that elements of $\mathcal{P}_r^-\Lambda^k(T)$ may be written as $\boldsymbol{\lambda}^{\boldsymbol{\alpha}}\omega^{F_\sigma}$, $|\boldsymbol{\alpha}| = r-1$, this yields (see [25])

$$\int_e \boldsymbol{\lambda}^{\boldsymbol{\alpha}}\omega^{F_\sigma} = |e|_0(\omega^{F_\sigma}(b_e)\cdot \boldsymbol{t}_e)\int_e \boldsymbol{\lambda}^{\boldsymbol{\alpha}}.$$

The latter term is a constant that depends on $\boldsymbol{\alpha}$ and the measure of e, making the computation a consequence of the old version of Algorithm 5, with the definition of the Bernstein basis functions generalized for $n > 1$.

In accordance with the case $n = 1$, we assume that the test mesh $\mathcal{M}$ is passed as a matrix whose i-th column contains the endpoints of the i-th edge. Thus the matrix TMesh representing $\mathcal{M}$ has $2n$ rows, since it contains on the same column coordinates of points $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$. For instance, when $n = 2$, TMesh$(:, i) = [x_0\ y_0\ x_1\ y_1]^T$ represents the segment that joins $\boldsymbol{p}_0 = (x_0, y_0)$ and $\boldsymbol{p}_1 = (x_1, y_1)$. We implicitly assume that this segment is contained in T. If an explicit function for computing generalised Vandermonde matrices for $k = 1$ and $n > 1$ is given and a collection of 1-simplices is provided, Algorithm 8 is easily adapted to this case as follows.

Algorithm 9 Evaluation of the Lebesgue constant for $k = 1$ and $n > 1$ on the test mesh

Require: The test mesh TMesh = (x), matrix W and (inverse of the) Vandermonde matrices V^{-1} (called A for uniform and D for GLL interpolation edges), vertices of small simplices xequi and xgll and their connectivity.
Ensure: The Lebesgue constant (called LebConstE1 for uniform and LebConstG1 for GLL edges).

```
1:  function [LebConstE1, LebConstG1] = Lebk1(xequi, xgll, A, D, Ed2Pt, TMesh)
2:  n = size(TMesh,1)/2;                          ▷ The dimension of the space
3:  vSe = vecnorm(xequi(:,Ed2Pt(2,:))-xequi(:,Ed2Pt(1,:)));   ▷ Vector containing edge measures
4:  Se = diag(vS);
5:  vSg = vecnorm(xgll(:,Ed2Pt(2,:))-xgll(:,Ed2Pt(1,:)));     ▷ Vector containing edge measures
6:  Sg = diag(vS);
7:  vC = 1./vecnorm(TMesh(n+1:2*n,:)-TMesh(1:n,:));   ▷ Vector containing (inverse of) edge lengths of TMesh
8:  C = diag(vC)
9:  LebConstE1 = norm(C*W*A*Se, inf);          ▷ Lebesgue constant for uniform edge interpolation
10: LebConstG1 = norm(C*W*D*Sg, inf);          ▷ Lebesgue constant for GLL edge interpolation
11: end
```

6 Numerical results

We present two kinds of results. The first one aims at motivating the use of Bernstein polynomials instead of the monomial ones. We compare the condition number of the Vandermonde matrix associated with uniform small edges $X^1_{r,min}(T)$, when the matrix is computed with respect to monomial and Bernstein bases. Results for $n = 2, 3$ in Table 1 show that Bernstein polynomials help in reducing the conditioning of the generalised Vandermonde matrix. Results for other choices of simplices given in [2] agree with this behaviour, in the sense that the conditioning of the Vandermonde matrix is more sensible to the selection of the polynomial basis rather than to the selection of the small simplices supporting the degrees of freedom.

The second result illustrates Lebesgue constants computed by means of the algorithms given in Section 4. In particular, Table 2 presents a comparison between these constants computed on the set $X^1_{r,min}(T)$ and on that associated with warp and blend 1-simplices for $n = 2, 3$. In accordance with what is known for the nodal case, we see that for edges as well non uniform distributions yield lower values of the Lebesgue constant.

We end this section with the script (Algorithm 10) that allows to compute the approximation of the Lebesgue constant using a uniform test mesh with M nodes mesh for $n = 1$, and the two values of k, $k = 0$ and $k = 1$. Note that lines 1 to 6 in Algorithm 10 are those of Algorithm 1. The routines `gll` and `QuadGaussLeg` have to be available.

	cond(V), $T \subset \mathbb{R}^2$		cond(V), $T \subset \mathbb{R}^3$	
r	monomial b.	Bernstein b.	monomial b.	Bernstein b.
1	1.0000×10^0	1.0000×10^0	1.0000×10^0	1.0000×10^0
2	9.6391×10^0	9.6391×10^0	1.1074×10^1	1.1074×10^1
3	3.3437×10^1	2.6374×10^1	7.1243×10^1	4.2973×10^1
4	1.6905×10^2	7.1572×10^1	4.3252×10^2	1.3742×10^2
5	9.9617×10^2	2.0038×10^2	3.4161×10^3	4.2761×10^2
6	5.3075×10^3	5.6324×10^2	2.3708×10^4	1.3035×10^3
7	3.3716×10^4	1.5876×10^3	1.9066×10^5	3.9360×10^3
8	2.1106×10^5	4.4782×10^3	1.5399×10^6	1.1751×10^4
9	1.3707×10^6	1.2635×10^4	1.3852×10^7	3.4779×10^4

Table 1: Condition number of the generalised Vandermonde matrix associated with uniform small edges $X^1_{r,min}(T)$ for the standard triangle $T \subset \mathbb{R}^2$ and the standard tetrahedron $T \subset \mathbb{R}^3$ with respect to the total polynomial degree r. We compare the monomial basis $\boldsymbol{\lambda}^{\boldsymbol{\alpha}}\omega_F$ for $\mathcal{P}_r^-\Lambda^k(T)$ with the Bernstein basis $\mathcal{B}_{\boldsymbol{\alpha}}\omega_F$.

	Estimated value Λ_r, $T \subset \mathbb{R}^2$		Estimated value Λ_r, $T \subset \mathbb{R}^3$	
r	uniform	warp and blend	uniform	warp and blend
1	1.00	1.00	1.00	1.00
2	4.94	4.94	6.41	6.41
3	7.92	6.71	11.23	10.80
4	12.17	8.16	18.04	15.25
5	18.92	9.60	29.37	20.79
6	29.95	11.62	46.76	28.32
7	48.31	14.51	74.19	36.03
8	79.45	17.65	127.53	45.82
9	133.03	20.32	218.19	57.24

Table 2: Lebesgue constants Λ_r associated with the set $X^1_{r,min}(T)$ of uniform small edges and the set of small edges with vertices in warp and blend nodes for the standard triangle $T \subset \mathbb{R}^2$ and the standard tetrahedron $T \subset \mathbb{R}^3$.

Algorithm 10 Example of main for $n = 1$ and $k = 0, 1$

```
M = input('number of points in the test mesh: ');                          ▷ Test mesh size
x = linspace(0, 1, M);                                                     ▷ Test mesh points
r = input('degree of the polynomial interpolation over the interval: ');   ▷ Polynomial interpolation degree
N = r+1;                                                                   ▷ Number of interpolation mesh points
xequi = linspace(0, 1, N);                                                 ▷ Uniform interpolation mesh points
xgll = (-gll(r)'+1)/2;                                                     ▷ Gauss-Lobatto-Legendre interpolation mesh points
[B, C] = Vandk0(xequi, xgll);                                              ▷ Vandermonde matrices for k = 0
[LebConstE0, LebConstG0] = Lebk0(x, B, C)                                  ▷ Lebesgue constants for k = 0
Ed2Pt = connect(r);                                                        ▷ Same connectivity for xequi and xgll
[A, D] = Vandk1(xequi, xgll, Ed2Pt);                                       ▷ Vandermonde matrices for k = 1
TMesh = [x(1:end-1); x(2:end)];                                            ▷ Edges of the test mesh
[LebConstE1, LebConstG1] = Lebk1(xequi, xgll, A, D, Ed2Pt, TMesh)          ▷ Lebesgue constants for k = 1
```

References

[1] A. Alonso Rodríguez, L. Bruni Bruno, F. Rapetti. Minimal sets of unisolvent weights for high order Whitney forms on simplices. *Numerical mathematics and advanced applications—Enumath 2019* in Lect. Notes Comput. Sci. Eng., 2020.

[2] A. Alonso Rodríguez, L. Bruni Bruno, F. Rapetti. Towards nonuniform distributions of unisolvent weights for Whitney finite element spaces on simplices: the edge element case, to appear in *Calcolo*, 2022, https://hal.inria.fr/hal-03114568/.

[3] A. Alonso Rodríguez, F. Rapetti. On a generalization of the Lebesgue's constant. *J. Comp. Phys*, 428: 109964, 2021.

[4] D. N. Arnold. Finite element exterior calculus. SIAM, 2018.

[5] D. N. Arnold, R. S. Falk, R. Winther. Finite element exterior calculus, homological techniques, and applications. *Acta Numer.* 15: 1–155, 2006.

[6] L. Bos, M.A. Taylor, B.A. Wingate. Tensor product Gauss-Lobatto points are Fekete points for the cube. *Math. Comp.*, 70: 1543–1547, 2001.

[7] A. Bossavit. Computational electromagnetism. *Academic Press, Inc., San Diego, CA*, 1998.

[8] L. Bruni Bruno. Weights as degrees of freedom for high order Whitney finite elements. PhD Thesis, University of Trento, 2022.

[9] C. Canuto, M.Y. Hussaini, A. Quarteroni, T.A. Zang. Spectral methods. Fundamentals in single domains. *Springer*, New York, 2007.

[10] R. Cavoretto, A. De Rossi. Software for Approximation 2022 (SA2022). *Dolomites Res. Notes Approx.*, Special Issue SA2022, 15:i–ii, 2022.

[11] Q. Chen, I. Babuska. Approximate optimal points for polynomial interpolation of real functions in an interval and in a triangle. *Comput. Methods Appl. Mech. Engrg.*, 128: 405–417, 1995.

[12] S. H. Christiansen, F. Rapetti., On high order finite element spaces of differential forms. *Math. Comp.*, 298(45): 517–548, 2016.

[13] R. Cools. Advances in multidimensional integration. *J. Comput. Appl. Math.*, 149: 1–12, 2002.

[14] M. Desbrun, E. Kanso, Y. Tong. Discrete differential forms for computational modeling. *ACM SIGGRAPH Procs.*, 2005.

[15] C. F. Dunkl, Y. Xu. Orthogonal Polynomials of Several Variables. *Cambridge University Press*, 2014.

[16] J. Harrison. Continuity of the integral as a function of the domain. *J. Geom. Anal.*, 8(5): 769–795, 1998.

[17] J. S. Hesthaven, T. Warburton. Nodal discontinuous Galerkin methods: algorithms, analysis and applications. *Texts in Applied Mathematics*, 54, Springer New York, 2008.

[18] R. Hiptmair. Canonical construction of finite elements. *Math. Comp.*, 228(68): 1325–1346, 1999.

[19] G. E. Karniadakis, S.J. Sherwin. Spectral/hp element methods for CFD. *Oxford Univ. Press*, New York 1999.

[20] J. R. Munkres. Elements of Algebraic Topology. *Springer*, 1984.

[21] J.-C. Nédélec. Mixed finite elements in $\mathbf{R}^3$. *Numer. Math.*, 35(3): 315–341, 1980.

[22] R. Pasquetti, F. Rapetti. Spectral element methods on unstructured meshes: which interpolation points? *Numer. Alg.*, 55, 349–366, 2010.

[23] F. Rapetti. Comments on a high-order Whitney complex for simplices. *IEEE J. on Multiscale and Multiphysics Comput. Technique*, 4:348–355, 2019.

[24] F. Rapetti, A. Bossavit. Whitney forms of higher degree. *SIAM J. Numer. Anal.*, 47(3): 2369–2386, 2009.

[25] F. Rapetti. High order edge elements on simplicial meshes. *Meth. Math. Anal. Numer.*, 41(6): 1001–1020, 2007.

[26] M. J. Roth, Nodal configurations and Voronoi Tessellations for triangular spectral elements. PhD thesis, University of Victoria, 2005.

[27] M. Vianello. Selected software packages. https://www.math.unipd.it/~marcov/MVsoft.html

[28] H. Whitney. Geometric Integration Theory. *Princeton University Press*, 1957.