ESTIMATING THE NUMBER OF SIMILARITY CLASSES FOR MARKED BISECTION IN GENERAL DIMENSIONS

Guillem Belda-Ferrín[1] Eloi Ruiz-Gironés[1] Xevi Roca[1,2]

[1] *Computer Applications in Science and Engineering, Barcelona Supercomputing Center - BSC, 08034 Barcelona, Spain*
[2] *Corresponding author: xevi.roca@bsc.es*

ABSTRACT

To measure the stability of a marked bisection method, we estimate in general dimensions an upper bound of the number of generated similarity classes. Moreover, to understand the cyclic similarity structure, we estimate the number of uniform refinements required to generate all the similarity classes. We first prove that switching to the newest vertex bisection after n uniform refinements is equivalent to switching after $n-2$ uniform refinements. Then, we obtain the similarity bound, a bound that we use to derive the number of uniform refinements required to generate all the similarity classes. Although the similarity bound is not tight, the results show that it estimates the magnitude of the expected number of classes. We also show the ratio of the number of similarity classes for marked bisection and newest vertex bisection. Because this ratio grows exponentially with the dimension, we conclude that marked bisection is suitable for low-dimensional applications.

Keywords: adaption, local bisection, n-dimensional bisection, similarity classes

1. INTRODUCTION

In adaptive n-dimensional refinement, conformal simplicial meshes must be locally modified. One systematic modification for arbitrary dimensions is to bisect a set of selected simplices. This operation splits each simplex by introducing a new vertex on a previously selected refinement edge. Then, this new vertex is connected to the original vertices to define two new simplices. To ensure that the mesh is still conformal, the bisection has to select additional refinement edges on a surrounding conformal closure.

In n-dimensional bisection, the edge selection is commonly based on choosing the longest edge [1–4], the newest vertex [5–10], or using marked bisection [11–13]. Although all these edge selections are well-suited for adaptation, marked n-dimensional bisection has shown to be suitable for local refinement of unstructured simplicial meshes of three or more dimensions [11–13].

Moreover, on unstructured meshes, marked bisection enforces a key advantage for n-simplicial adaption. That is, successive refinement leads to a fair number of simplex similarity classes [11–13]. Two simplices belong to the same similarity class if the first one can be transformed into the second one using uniform scalings, rotations, reflections, and translations. That is, both simplices have the same shape, but different sizes, alignments, orientations, and positions. As a consequence, both simplices have the same shape quality. Since the number of similarity classes is bounded, so is the minimum mesh quality, and therefore, the bisection method is stable.

To understand this stability advantage, we overview the structure of marked bisection. These methods feature a first stage performing a specific-purpose bisection for marked simplices. This marked bisection enforces that after a few initial steps, one can switch independently on each element to another stage featuring Maubach's newest vertex bisection [7]. The

number of initial bisection steps is comparable with the spatial dimension. Hence, these steps are responsible for a reasonable increase in the total number of similarity classes.

Estimating the number of similarity classes is key to assess the quality of a marked bisection method. This is so because this number indicates *a priori* the minimum mesh quality under successive mesh refinement. If the number of classes is smaller, the minimum quality tends to be higher. In the 3D case, there is a tight bound on the number of similarity classes [11]. However, still missing is an estimation of the number of similarity classes generated under successive refinement in arbitrary dimensions.

Accordingly, the main contribution of this work is to estimate in arbitrary dimensions an upper bound of the number of similarity classes obtained with a conformal marked bisection method. Moreover, to understand the cyclic structure of the minimum quality, we estimate the number of uniform refinements required to generate all the similarity classes. In particular, we will analyze the number of similarity classes and the number of iterations to obtain them when bisecting simplices with the algorithm presented in [13]. To estimate an upper bound of the number of similarity classes in the worst case, we consider a general simplex without any special symmetry. Note that simplices featuring specific symmetries feature a smaller number of similarity classes. Thus, the corresponding estimates are not upper bounds for the worst case.

To estimate the number of similarity classes, we use two ingredients. First, the maximum number of similarity classes for the newest vertex bisection [11]. Second, for marked bisection, we deduce that switching after n refinements to newest vertex with Maubach tag equal to n is equivalent to switching after $n-2$ refinements to tag equal to 2. Using our estimate, we derive the number of uniform refinements required to generate all the similarity classes. The results are devised to check how tight is the estimation of the number of similarity classes.

For marked bisection, the number of similarity classes has not been estimated for a general simplex in arbitrary dimensions. Alternatively, existing works estimate the number of similarity classes for longest edge bisection [14, 15]. For general shapes, the work [14] bounds the number of similarity classes for longest-edge bisection in two dimensions. For general dimensions, the work [15] numerically computes the number of similarity classes for longest-edge bisection for the equilateral simplex. In our work, we estimate the number of similarity classes for marked bisection and a general simplex in arbitrary dimensions.

The rest of the paper is structured as follows. In Section 2, we introduce the preliminary notation and concepts. In Section 3, we summarize the used marked bisection. In Section 4, we provide an upper bound to the number of similarity classes. In Section 5, we deduce a lower bound of the number of uniform refinements to obtain all the similarity classes. In Section 6, we show several examples. Finally, in Section 7, we present the concluding remarks of this work.

2. PRELIMINARIES

We proceed to introduce the necessary notation and concepts. Specifically, we introduce the preliminaries related simplicial meshes, conformity, and bisection methods. Then, to summarize the marked bisection algorithm [13], we introduce the notion of multi-id to provide a unique identifier to the mid-vertices, and the selection of the bisection edge in a consistent manner.

2.1 Simplicial meshes, conformity, and bisection

A *simplex* is the convex hull of $n+1$ points $p_0, \ldots, p_n \in \mathbb{R}^n$ that do not lie in the same hyperplane. We denote a simplex as $\sigma = conv(p_0, \ldots, p_n)$. We identify each point p_i with a unique integer identifier v_i that we refer as *vertex*. Thus, a simplex is composed of $n+1$ vertices and we denote it as $\sigma = (v_0, \ldots, v_n)$ where v_i is the identifier of point p_i. We have an application Π that maps each identifier v_i to the corresponding point p_i.

Given a simplex σ, a *k-entity* is a sub-simplex composed of $k+1$ vertices of σ, for $0 \leq k \leq n-1$. We say that a 1-entity is an *edge* and an $(n-1)$-entity is a *face*. The number of k-entities contained in a simplex σ is

$$\binom{n+1}{k+1}.$$

Particularly, the number of edges and faces of σ is

$$\binom{n+1}{2} = \frac{n(n+1)}{2}, \quad \binom{n+1}{n} = n,$$

respectively. We associate each face of a simplex σ to its opposite vertex in σ. Specifically, the opposite face to v_i is

$$\kappa_i = (v_0, v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n).$$

We say that two simplices σ_1 and σ_2 are *neighbors* if they share a face.

We define the *bisection* of a simplex as the operation that splits a simplex by introducing a new vertex on the selected refinement edge, see Figure 1. Then, the vertices not lying on this refinement edge are connected to the new vertex. These connections determine two new simplices.

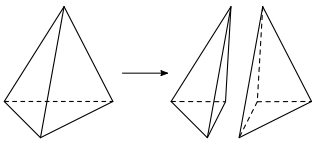


Figure 1: Bisection of a tetrahedron.

Algorithm 1 Refining a subset of a mesh.

input: `Mesh` $\mathcal{T}$, `SimplicesSet` $\mathcal{S} \subset \mathcal{T}$
output: `ConformalMarkedMesh` $\mathcal{T}_2$
1: **function** `refineMesh`($\mathcal{T}$, $\mathcal{S}$)
2: $\mathcal{T}_1 =$ `markMesh`($\mathcal{T}$)
3: $\mathcal{T}_2 =$ `localRefine`($\mathcal{T}_1, \mathcal{S}$)
4: **return** $\mathcal{T}_2$
5: **end function**

2.2 Marked bisection

To perform the bisection process, we adapt to the n-dimensional case the recursive refine-to-conformity scheme proposed in [11]. The marked bisection method, Algorithm 1, starts by marking the initial unstructured conformal mesh and then applies a local refinement procedure to a set of simplices of the marked mesh. To do it so, we need to specify a conformal marking procedure for simplices to obtain a marked mesh $\mathcal{T}_1$. Using this marked mesh, the local refinement procedure, Algorithm 9, first refines a set of simplices, then calls a recursive refine-to-conformity strategy, and finally renumbers the mesh. The refine-to-conformity strategy, Algorithm 10, terminates when successive bisection leads to a conformal mesh. Both algorithms use marked bisection to refine a set of elements, see Algorithm 11. See more details of the involved algorithms in Appendix A.

2.3 Unique mid-vertex identifiers

We use *multi-ids* to uniquely identify the new vertices that are created during the bisection process. A multi-id is a sorted list of vertices, $\mathbf{v} = [v_1, \ldots, v_k]$, where $v_1 \leq v_2 \leq \ldots \leq v_k$. A simplex that contains multi-ids is denoted as $\sigma = (\mathbf{v}_0, \ldots, \mathbf{v}_n)$.

When creating a new vertex after bisecting and edge, we generate a multi-id for the new vertex. The new multi-id is the combination of the multi-ids of the edge vertices, $\mathbf{v}_0$ and $\mathbf{v}_1$. In particular, the resulting multi-id is created by merging and sorting the the multi-ids of $\mathbf{v}_0$ and $\mathbf{v}_1$. We remark that the ids can appear more than once after generating a new multi-id.

2.4 Consistent bisection edge

For all mesh entities shared by different mesh elements, we must ensure that these entities have the same bisection edge on all those elements. To this end, we base this selection on a strict total order of the mesh edges. The main idea is to order the edges from the longest one to the shortest one, and use a tie-breaking rule for the edges with the same length. Specifically, we define the consistent bisection edge of a simplex as the longest edge with the lowest global index.

A shared edge between two simplices may have a different order of vertices, which can induce different results when computing the edge length from different elements. To avoid these discrepancies, we first order the edge vertices according to the vertex ordering. Then, we compute the length of the edge using the ordered vertices.

To define a strict total order of edges, when two edges have the same length, we need a tie-breaking rule. To this end, we use a lexicographic order for the global edges in terms of the order of the vertices.

2.5 Similarity classes

Two simplices σ_1 and σ_2 are similar if there exists an affine mapping $F(\mathbf{x}) = \lambda \mathbf{A}\mathbf{x} + \mathbf{b}$ for $\lambda \in \mathbb{R}$, an orthonormal matrix $\mathbf{A}$, and $F(\sigma_1) = \sigma_2$. Because similarity is an equivalence relation, all the simplices that are similar between them form a similarity class.

Although similar simplices may have different sizes, alignments, orientations, or positions, they have the same shape. Therefore, all the simplices in a similarity class have the same shape quality. In Figure 2, we show the obtained similarity classes by refining a triangle. The initial triangle defines the first similarity class, Figure 2(a). In the first bisection step, Figure 2(b), we obtain two additional similarity classes. The next uniform refinement obtains an additional similarity class and repeats the initial similarity class, see Figure 2(c). Finally, when bisecting the triangles of the fourth similarity class, we obtain the second and third similarity classes, see Figure 2(d).

3. MARKED BISECTION IN GENERAL DIMENSIONS

Following, we summarize an n-dimensional marked bisection algorithm [13]. First, we detail the co-dimensional marking process, which is based on the consistent bisection edge of a simplex. Then, we define the three stages of the bisection process.

3.1 Co-dimensional marking process

We detail the codimensional marking process for a simplex, in which the resulting mark is a bisection tree. The bisection tree is computed by traversing

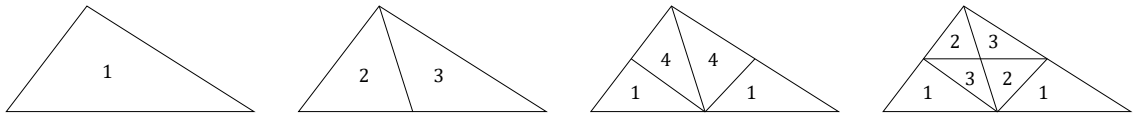


Figure 2: Similarity classes of a triangle obtained by bisection: (a) initial triangle; (b) one uniform refinement; (c) two uniform refinements; and (d) further refinements do not increase similarity classes.

Algorithm 2 Mark a k-simplex.

input: `k-Simplex` σ
output: `BisectionTree` t

```
 1: function stageOneTree(σ)
 2:     e = consistentBisectionEdge(σ)
 3:     if dim σ = 1 then
 4:         t = tree(node = e)
 5:     else
 6:         ([v1], [v2]) = e
 7:         κ1 = oppositeFace(σ, [v1])
 8:         κ2 = oppositeFace(σ, [v2])
 9:         t1 = stageOneTree(κ1)
10:         t2 = stageOneTree(κ2)
11:         t = tree(node = e, left = t1, right = t2)
12:     end if
13:     return t
14: end function
```

Algorithm 3 Bisection of a marked simplex ρ.

input: `MarkedSimplex` ρ
output: `MarkedSimplex` ρ_1, `MarkedSimplex` ρ_2

```
 1: function bisectSimplex(ρ)
 2:     l = level(ρ)
 3:     if l < n − 1 then
 4:         τ = TreeSimplex(ρ)
 5:         τ1, τ2 = bisectStageOne(τ)
 6:         ρ1, ρ2 = MarkedSimplex(τ1, τ2)
 7:     else if l = n − 1 then
 8:         τ = TreeSimplex(ρ)
 9:         μ1, μ2 = bisectCastToMaubach(τ)
10:         ρ1, ρ2 = MarkedSimplex(μ1, μ2)
11:     else
12:         μ = MaubachSimplex(ρ)
13:         μ1, μ2 = bisectMaubach(μ)
14:         ρ1, ρ2 = MarkedSimplex(μ1, μ2)
15:     end if
16:     return ρ1, ρ2
17: end function
```

Algorithm 4 Bisect a marked tree-simplex.

input: `TreeSimplex` τ
output: `TreeSimplex` τ_1, `TreeSimplex` τ_2

```
 1: function bisectStageOne(τ)
 2:     (σ, κ̄, t, l) = τ
 3:     e = root(t)                          ▷ Bisection edge
 4:     σ1, κ̄1, σ2, κ̄2 = bisectTreeSimplex(σ, κ̄, e, l)
 5:     t1 = left(t); t2 = right(t)          ▷ Bisect tree
 6:     l1 = l + 1; l2 = l + 1               ▷ Bisect level
 7:     τ1 = (σ1, κ̄1, t1, l1)
 8:     τ2 = (σ2, κ̄2, t2, l2)
 9:     return τ1, τ2
10: end function
```

the sub-entities of the simplex in a recursive manner and selecting the consistent bisection edge of each sub-simplex, see Algorithm 2. The resulting *bisection tree* has height n, and the tree nodes of level i correspond to the consistent bisection edges of sub-simplices of co-dimension i (dimension $n - i$).

Next, we detail the codimensional marking process for a single simplex, Algorithm 2. Since the codimensional marking process is the first step of the mesh refinement algorithm, the length of the multi-ids of all simplices is one. The input of the function is a simplex $\sigma = ([v_0], \ldots, [v_n])$ and the output is the corresponding bisection tree. First, we obtain the consistent bisection edge, e, of the simplex, see Line 2. If σ is an edge, this corresponds to the base case of the recursion and we return a tree with only the root node. Otherwise, we obtain the opposite faces of the vertices of the bisection edge, see Lines 7–8. Then, we recursively call the marking process algorithm for the faces κ_1 and κ_2, and we obtain the corresponding trees t_1 and t_2, see Lines 9–10. Finally, we build the bisection tree t with the bisection edge as root node and the trees t_1 and t_2 as left and right branches, see Line 11.

3.2 First bisection stage: tree simplices

In the first stage, we bisect the simplices using the bisection trees computed with the codimensional marking process. The first stage is used in the first $n - 2$ bisection steps. Moreover, during the refinement process we store the new mid-vertices into $\bar{\kappa}$. Thus, in the second stage, we are able to reorder the generated simplices and, in the third stage, use newest vertex bisection. To this end, Algorithms 4 and 5 detail the

Algorithm 5 Bisect a tree-simplex.

input: `Simplex` σ, l-`List` $\bar{\kappa}$, `Edge` e, `Level` l
output: `Simplex` σ_1, $(l+1)$-`List` $\bar{\kappa}_1$, `Simplex` σ_2, $(l+1)$-`List` $\bar{\kappa}_2$

function `bisectTreeSimplex`(σ, $\bar{\kappa}$, e, l)
  $(\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_n) = \sigma$
  $([v_{1,l-1}, v_{2,l-1}], \ldots, [v_{1,0}, v_{2,0}]) = \bar{\kappa}$
  $([v_{1,l}], [v_{2,l}]) = e$
  $[v_{1,l}, v_{2,l}] =$ `midVertex`$([v_{1,l}], [v_{2,l}])$
  $(i_1, i_2) =$ `simplexVertices`(σ, e)
  $\sigma_1 = (\mathbf{v}_0, \ldots, \mathbf{v}_{i_2-1}, [v_{1,l}, v_{2,l}], \mathbf{v}_{i_2+1}, \ldots, \mathbf{v}_n)$
  $\sigma_2 = (\mathbf{v}_0, \ldots, \mathbf{v}_{i_1-1}, [v_{1,l}, v_{2,l}], \mathbf{v}_{i_1+1}, \ldots, \mathbf{v}_n)$
  $\bar{\kappa}_1 = ([v_{1,l}, v_{2,l}], [v_{1,l-1}, v_{2,l-1}] \ldots, [v_{1,0}, v_{2,0}])$
  $\bar{\kappa}_2 = ([v_{1,l}, v_{2,l}], [v_{1,l-1}, v_{2,l-1}] \ldots, [v_{1,0}, v_{2,0}])$
  return σ_1, $\bar{\kappa}_1$, σ_2, $\bar{\kappa}_2$
end function

Algorithm 6 Bisect to Maubach

input: `TreeSimplex` τ
output: `MaubachSimplex` μ_1, `MaubachSimplex` μ_2

function `bisectToMaubach`(τ)
  $(\sigma, \bar{\sigma}, t, l) = \tau$
  $e =$ `root`(t)
  $\sigma_1, \bar{\kappa}_1, \sigma_2, \bar{\kappa}_2 =$ `bisectTreeSimplex`$(\sigma, \bar{\kappa}, e, l)$
  $\bar{\sigma}_1, \bar{\sigma}_2 =$ `castToMaubach`$(e, \bar{\kappa}_1, \bar{\kappa}_2)$
  $d_1 = n$; $d_2 = n$
  $l_1 = l+1$; $l_2 = l+1$
  $\mu_1 = (\bar{\sigma}_1, d_1, l_1)$
  $\mu_2 = (\bar{\sigma}_2, d_2, l_2)$
  return μ_1, μ_2
end function

bisection process of a tree simplex in the first stage.

3.3 Second bisection stage: casting to Maubach

We next detail the second stage of the bisection method for simplices, a stage that is used when the descendant level of a tree-simplex is $l = n-1$. In this stage, after bisecting a tree simplex, we reorder the vertices of the bisected simplices in order to apply newest vertex bisection in the third stage. This process is explained in Algorithms 6 and 7, in which a tree simplex, τ, is bisected into two Maubach simplices, μ_1 and μ_2.

3.4 Third stage: Maubach's bisection

Finally, we detail the third stage of the bisection method for simplices, which used when the descendant level of a Maubach simplex, μ, is $l \geq n$. In this stage, we use Maubach's algorithm to favor the conformity, finiteness, stability, and locality properties. We reinterpret Maubach's algorithm using tagged simplices

Algorithm 7 Cast to Maubach.

input: `Edge` e, n-`List` $\bar{\kappa}_1$, n-`List` $\bar{\kappa}_2$
output: n-`Simplex` $\bar{\sigma}_1$, n-`Simplex` $\bar{\sigma}_2$

function `castToMaubach`(e, $\bar{\kappa}_1$, $\bar{\kappa}_2$)
  $([v_{1,n-1}], [v_{2,n-1}]) = e$
  $([v_{1,n-1}, v_{2,n-1}], \ldots, [v_{1,0}, v_{2,0}]) = \bar{\kappa}_1$
  $([v_{1,n-1}, v_{2,n-1}], \ldots, [v_{1,0}, v_{2,0}]) = \bar{\kappa}_2$
  $\bar{\sigma}_1 = ([v_{1,n-1}], [v_{1,n-1}, v_{2,n-1}], \ldots, [v_{1,0}, v_{2,0}])$
  $\bar{\sigma}_2 = ([v_{2,n-1}], [v_{1,n-1}, v_{2,n-1}], \ldots, [v_{1,0}, v_{2,0}])$
  return $\bar{\sigma}_1$, $\bar{\sigma}_2$
end function

Algorithm 8 Adapted Maubach's algorithm.

input: `MaubachSimplex` μ
output: `MaubachSimplex` μ_1, `MaubachSimplex` μ_2

function `bisectMaubach`(μ)
  $((\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_n), d, l) = \mu$
  $\mathbf{w} =$ `midVertex`$(\mathbf{v}_0, \mathbf{v}_d)$
  $\bar{\sigma}_1 = (\mathbf{v}_0, \ldots, \mathbf{v}_{d-1}, \mathbf{w}, \mathbf{v}_{d+1}, \ldots, \mathbf{v}_n)$
  $\bar{\sigma}_2 = (\mathbf{v}_1, \ldots, \mathbf{v}_d, \mathbf{w}, \mathbf{v}_{d+1}, \ldots, \mathbf{v}_n)$
  Set $d' = \begin{cases} d-1, & d > 1 \\ n, & d = 1 \end{cases}$
  $d_1 = d'$; $d_2 = d'$
  $l_1 = l+1$; $l_2 = l+1$
  $\mu_1 = (\bar{\sigma}_1, d_1, l_1)$
  $\mu_2 = (\bar{\sigma}_2, d_2, l_2)$
  return μ_1, μ_2
end function

and multi-ids in Algorithm 8.

4. ESTIMATION OF THE NUMBER OF SIMILARITY CLASSES

We estimate an upper bound of the number of similarity classes obtained with marked bisection, and we show that this number is sub-optimal. That is, it is greater than the number of similarity classes obtained using newest vertex bisection.

Lemma 4.1 (Newest vertex bisection for triangular meshes)**.** *The marked bisection is equivalent to Maubach's bisection for 2-simplices.*

Proof. The co-dimensional marking process generates three possible bisection trees for 2-simplices. Those

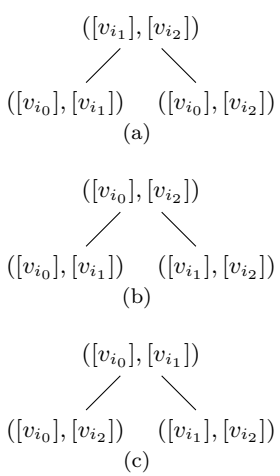
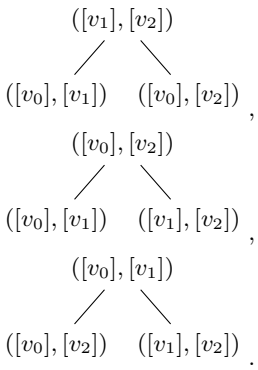


Figure 3: The three possible bisection trees t_i corresponding to the consistent bisection edges (a) $([v_{i_1}],[v_{i_2}])$, (b) $([v_{i_0}],[v_{i_2}])$, and (c) $([v_{i_0}],[v_{i_1}])$.

bisection trees are



These trees are equivalent to the obtained ones when applying Maubach's method to the triangles $([v_1],[v_0],[v_2])$, $([v_0],[v_1],[v_2])$, and $([v_0],[v_2],[v_1])$, with Maubach tag $d = 2$ and bisection level $l = 0$. Therefore, for triangular meshes the presented marked bisection algorithm is equivalent to Maubach's algorithm. □

Proposition 4.1 (Tagging with $d = 2$ after step $n-2$)**.** *Let σ be a simplex marked with the co-dimensional marking process, and consider the mesh $\mathcal{Q}^{\sigma}_{n-2}$ obtained after $n-2$ uniform refinements with marked bisection. Then, we can map a tree-simplex τ of $\mathcal{Q}^{\sigma}_{n-2}$ to a Maubach simplex μ with descendant level $l = n-2$ and tag $d = 2$.*

Proof. Let σ_0 be a simplex marked with the co-dimensional marking process and $\mathcal{Q}^{\sigma_0}_{n-1}$ be the mesh obtained after $n-2$ uniform marked bisection refinements. Let $\tau \in \mathcal{Q}^{\sigma_0}_{n-2}$ be a tree-simplex of the form $\tau = (\sigma, \bar{\kappa}, t, l = n-2)$. After applying $n-2$ uniform refinements with marked bisection, we know that σ is composed of 3 original vertices and $n-2$ multivertices. That is,

$$\sigma = \{[v_{i_0}],[v_{i_1}],[v_{i_2}],[v_{1,n-3},v_{2,n-3}],\ldots,[v_{1,0},v_{2,0}]\}.$$

Since the bisection tree t of σ is composed of vertices $[v_{i_0}]$, $[v_{i_1}]$, and $[v_{i_2}]$, that define a triangle, its bisection tree is equivalent to the bisection tree of a triangle. By Lemma 4.1, t is equivalent to the bisection tree of a tagged triangle. Thus, t is one of the bisection trees depicted in Figure 3.

If we map the tree-simplex τ corresponding to the simplex σ to a Maubach simplex $\mu = (\bar{\sigma}, l = n-2, d = 2)$, we have that there are three possible simplices $\bar{\sigma}$, illustrated in Equation (1):

$$([v_{i_1}],[v_{i_0}],[v_{i_2}],[v_{1,n-3},v_{2,n-3}],\ldots,[v_{1,0},v_{2,0}]), \quad (1a)$$

$$([v_{i_0}],[v_{i_1}],[v_{i_2}],[v_{1,n-3},v_{2,n-3}],\ldots,[v_{1,0},v_{2,0}]), \quad (1b)$$

$$([v_{i_0}],[v_{i_2}],[v_{i_1}],[v_{1,n-3},v_{2,n-3}],\ldots,[v_{1,0},v_{2,0}]). \quad (1c)$$

We recall that we sorted the vertices $[v_{i_0}]$, $[v_{i_1}]$, and $[v_{i_2}]$ according to the tagged triangles of the proof of Lemma 4.1. Thus, we have that the simplices of Equations (1a)–(1c) correspond to the bisection trees depicted in Figures 3(a)–3(c).

Then, it only remains to check that if we apply two uniform tagged-bisection steps to any simplex of Equation (1), the obtained Maubach simplices are equal to the Maubach simplex obtained after n uniform refinements with marked bisection.

For the sake of simplicity, we only perform the reasoning for the simplex in Equation (1a). Thus, let $\mu = (\bar{\sigma}, l = n-2, d = 2)$ be a Maubach simplex, where $\bar{\sigma}$ is defined in Equation (1a). Performing the first Maubach's bisection step, we obtain two children $\mu_1 = (\bar{\sigma}_1, l = n-1, d = 1)$ and $\mu_2 = (\bar{\sigma}_2, l = n-1, d = 1)$, where

$$\bar{\sigma}_1 = ([v_{i_1}],[v_{i_0}],[v_{i_1},v_{i_2}], [v_{1,n-3},v_{2,n-3}],[v_{1,n-4},v_{2,n-4}],\ldots,[v_{1,0},v_{2,0}]),$$

$$\bar{\sigma}_2 = ([v_{i_0}],[v_{i_2}],[v_{i_1},v_{i_2}], [v_{1,n-3},v_{2,n-3}],[v_{1,n-4},v_{2,n-4}],\ldots,[v_{1,0},v_{2,0}]),$$

According to Maubach's algorithm, the bisection edge of level $l = n-2$ is $([v_{i_1}],[v_{i_2}])$. This bisection edge is the same as the consistent bisection edge according to the bisection tree in Figure 3(a). Therefore, we have that $[v_{i_1},v_{i_2}] = [v_{1,n-2},v_{2,n-2}]$ and that this tagged bisection step is the same as the one performed using

marked bisection. We substitute the new multivertex in $\bar{\sigma}_1$ and $\bar{\sigma}_2$ to obtain

$$\bar{\sigma}_1 = ([v_{i_1}], [v_{i_0}], [v_{1,n-2}, v_{2,n-2}], [v_{1,n-3}, v_{2,n-3}], \ldots, [v_{1,0}, v_{2,0}]),$$

$$\bar{\sigma}_2 = ([v_{i_0}], [v_{i_2}], [v_{1,n-2}, v_{2,n-2}], [v_{1,n-3}, v_{2,n-3}], \ldots, [v_{1,0}, v_{2,0}]).$$

Again, for the sake of simplicity, we only perform a tagged bisection on simplex μ_1, since the same argument can be applied to simplex μ_2. We have that the result of Maubach's bisection on μ_1 are the simplices $\mu_{1,1} = (\bar{\sigma}_{1,1}, l = n, d = n)$ and $\mu_{1,2} = (\bar{\sigma}_{1,2}, l = n, d = n)$, where

$$\bar{\sigma}_{1,1} = ([v_{i_1}], [v_{i_0}, v_{i_1}], [v_{1,n-2}, v_{2,n-2}], \ldots, [v_{1,0}, v_{2,0}]),$$
$$\bar{\sigma}_{1,2} = ([v_{i_0}], [v_{i_0}, v_{i_1}], [v_{1,n-2}, v_{2,n-2}], \ldots, [v_{1,0}, v_{2,0}])$$

According to Maubach's tagged bisection, the bisection edge of level $l = n - 1$ is $([v_{i_0}], [v_{i_1}])$. Again, this bisection edge is the same as the consistent bisection edge according the bisection tree in Figure 3(a). Therefore, we have that $[v_{i_0}, v_{i_1}] = [v_{1,n-1}, v_{2,n-1}]$.

Analogously, the bisection edge of level $l = n - 1$ is $([v_{i_0}], [v_{i_1}])$. Therefore, by susbtituting the new multivertex in both children, we obtain

$$\bar{\sigma}_{1,1} = ([v_{i_1}], [v_{1,n-1}, v_{2,n-1}], [v_{1,n-2}, v_{2,n-2}], \ldots, [v_{1,0}, v_{2,0}]),$$

$$\bar{\sigma}_{1,2} = ([v_{i_0}], [v_{1,n-1}, v_{2,n-1}], [v_{1,n-2}, v_{2,n-2}], \ldots, [v_{1,0}, v_{2,0}]).$$

After performing the mapping to Maubach process at the end of the second stage, we obtain that $[v_{i_0}] = [v_{1,n-1}]$ and $[v_{i_1}] = [v_{2,n-1}]$. Therefore, the obtained simplices are the same that the ones obtained after n uniform refinements with marked bisection. That is, $\bar{\sigma}_{1,1}$ and $\bar{\sigma}_{1,2}$ are equal to the simplices obtain at Lines 5 and 6 of Algorithm 7, respectively.

The same argument can be applied when bisecting the simplex μ_2, and also the rest of the simplices in Equation (1). Therefore, we have proved that the simplices obtained at level $l = n - 2$ can be mapped to Maubach's simplices of tag $d = 2$. □

Now, we can state the theorem for the upper bound S_n over the similarity classes generated by the marked bisection algorithm. To do that, we consider uniform refinements in order to generate the maximum number of simplices per iteration. Thus, let $\mathcal{Q}_0^\sigma = \sigma$ and

$$\mathcal{Q}_k^\sigma = \texttt{bisectSimplices}(\mathcal{Q}_{k-1}^\sigma, \mathcal{Q}_{k-1}^\sigma)$$

the obtained mesh after performing k uniform refinements, a mesh $\mathcal{Q}_i^\sigma$ that is composed of $\#(\mathcal{Q}_i^\sigma) = 2^i$ simplices. Considering all the meshes $\mathcal{Q}_0^\sigma, \ldots \mathcal{Q}_k^\sigma$, we have at most

$$\sum_{i=0}^{k} \#(\mathcal{Q}_i^\sigma) = \sum_{i=0}^{k} 2^i = 2^{k+1} - 1 \tag{2}$$

different simplices.

Theorem 4.1 (Number of similarity classes for marked bisection)**.** *Let σ be a simplex marked with the co-dimensional marking process. Assume that from iteration k, the bisection process is equivalent to Maubach's bisection. Then, the number of similarity classes generated by the marked bisection method is at most*

$$S_n = (2^k - 1) + 2^k M_n,$$

where $M_n = nn!2^{n-2}$ is the maximum number of similarity classes of newest vertex bisection.

Proof. Let σ be a simplex marked with the co-dimensional marking process. Consider k uniform refinements with marked bisection such that further refinements of marked bisection are equivalent to Maubach's bisection. By Equation (2), the number of similarity classes generated from iteration 0 to $k-1$ is, at most, $2^k - 1$. Since the number of simplices of $\mathcal{Q}_k^\sigma$ is 2^k, the number of simplices generated using Maubach's algorithm is $2^k M_n$, where M_n is a bound of similarity classes of Maubach's algorithm, see Theorem 4.5 of [11]. Finally, summing the two values we obtain that the number of similarity classes is at most $S_n = (2^k - 1) + 2^k M_n$, as we wanted to see. □

Corollary 4.1.1. *In the presented marked bisection, k is at most $n-2$, and the number of similarity classes is at most*

$$S_n = (2^{n-2}-1)+2^{n-2}M_n = (2^{n-2}-1)+2^{n-2}nn!2^{n-2}.$$

Proof. By Proposition 4.1, at iteration $n - 2$ we can map all the simplices of $\mathcal{Q}_{n-2}^\sigma$ to Maubach simplices with descendant level $l = n-2$ and tag $d = 2$. Therefore, by Theorem 4.1, the number of similarity classes generated by the marked bisection algorithm is at most

$$S_n = (2^{n-2} - 1) + 2^{n-2} M_n.$$

□

As a consequence of Corollary 4.1.1, the additional number of similarity classes obtained with marked bisection may grow exponentially with the dimension. Therefore, marked bisection is only suitable for lower dimensions, when the additional number of similarity classes is also small.

5. NUMBER OF UNIFORM REFINEMENTS TO OBTAIN ALL THE SIMILARITY CLASSES

To understand the cyclic structure of the similarity classes, we want to calculate the minimum number of uniform refinements required to generate all the similarity classes with the proposed marked bisection. To this end, we first compute the number of uniform refinements to obtain all similarity classes with newest vertex bisection.

Newest vertex bisection generates at most $M_n = nn!2^{n-2}$ similarity classes, see Theorem 4.5 of [11]. We remark that the number of generated similarity classes is an upper bound and therefore, for some simplices we can obtain less similarity classes than M_n. Thus, in the case that the method generates M_n similarity classes, it needs to refine at least K_n times uniformly to generate all of them, where K_n holds that

$$2^{K_n+1} - 1 \geq M_n = nn!2^{n-2}.$$

Thus, K_n has to fulfill that

$$2^{K_n+1} \geq 1 + nn!2^{n-2}.$$

Applying logarithms in both sides, we obtain that

$$K_n \geq \lceil \log_2(1 + nn!2^{n-2}) \rceil - 1.$$

Theorem 5.1 (Number of uniform refinements to obtain all similarity classes)**.** *Let σ be a simplex marked with the co-dimensional marking process. Assume that from iteration k, the bisection process is equivalent to Maubach's bisection. Then, the minimum number of uniform refinements to obtain all the similarity classes in marked bisection is*

$$I_n = k + K_n.$$

Proof. The first k uniform refinements are performed using marked bisection. Then, the following refinements are performed using newest vertex bisection. Thus, from the refinement $k+1$ onward, all the simplices are bisected using newest vertex bisection. To generate all the similarity classes of this simplices we need at least K_n uniform refinements. Therefore, to generate all the similarity classes of the initial simplex we need at least $I_n = k + K_n$ uniform refinements □

Corollary 5.1.1. *In the presented marked bisection, k is at most $n-2$, and therefore, the minimum number of uniform refinements to obtain all the similarity classes is*

$$I_n = n - 2 + K_n.$$

Proof. By Proposition 4.1, at bisection level $n-2$ we can map the obtained simplices to Maubach simplices with tag $d = n$. Therefore, by Theorem 5.1, and using k is at most $n-2$, the minimum number of uniform refinements to obtain all the similarity classes is

$$I_n = n - 2 + K_n.$$

□

As a consequence of Corollary 5.1.1, we see that to obtain all the similarity classes, we first need to bisect the simplices until the bisection process is driven by newest vertex bisection. Then, we need to perform the required number of iterations to obtain all the similarity classes of newest vertex bisection. In the case of the presented marked bisection, the first stages are performed in the first $n-2$ iterations.

6. EXAMPLES

We present an example in which we compute the number of similarity classes of different simplices and compare the obtained number with our upper bound. We have computed the shape quality of the mesh using the expression

$$\frac{n \det(S)^{2/n}}{\operatorname{tr}(S^t S)},$$

where S is the Jacobian of the affine mapping between the ideal equilateral simplex and the physical simplex [16, 17].

All the results have been obtained on a MacBook Pro with one dual-core Intel Core i5 CPU, with a clock frequency of 2.7GHz, and a total memory of 16GBytes. As a proof of concept, a mesh refiner has been fully developed in Julia 1.4. The Julia prototype code is sequential (one execution thread), corresponding to the implementation of the method summarized in this work.

6.1 Number of similarity classes

In this example, we show the number of similarity classes obtained with the marked bisection algorithm for different simplices and dimension. To this end, we uniformly refine an equilateral simplex, a Cartesian simplex, a Kuhn simplex, and an irregular simplex for dimensions two, three, four, and five. The equilateral simplex has all its edges of the same length, the Cartesian simplex has vertices determined by the origin and the canonical vectors, the Kuhn simplex is one of the simplices obtained after dividing a hypercube with Coxeter-Freudenthal-Kuhn algorithm, and the irregular simplex has all of its edges with different lengths.

We numerically predict the number of similarity classes using the quality of the simplices as a proxy.

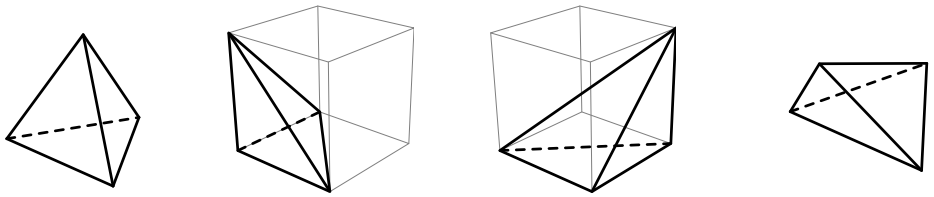


Figure 4: Three-dimensional simplices considered in example 1: (a) equilateral; (b) Cartesian; (c) Khun; and (d) irregular.

Table 1: Number of generated similarity classes by marked bisection.

Dimension	Equilateral	Cartesian	Kuhn	Irregular	S_n	M_n	S_n/M_n
2	3	1	1	4	4	4	1.00
3	17	17	3	69	73	36	2.02
4	52	45	4	1119	1539	384	4.01
5	185	301	5	32979	38407	4800	8.00

Table 2: Number of uniform refinements to generate all the similarity classes.

Dimension	Equilateral	Cartesian	Kuhn	Irregular	I_n	K_n	$I_n - K_n$
2	2	2	2	2	2	2	0
3	7	7	2	7	6	5	1
4	10	10	3	13	10	8	2
5	15	18	4	18	15	12	3

Specifically, we assign an obtained shape quality to a similarity class. With this idea, we uniformly refine the initial simplices and their descendants until the bisection process does not generate more similarity classes.

Table 1 shows the number of obtained similarity classes for each case. The equilateral and Cartesian simplex have fewer similarity classes than S_n. That is because they have geometric symmetries, and thus marked bisection generates less similarity classes than S_n. On the other hand, the Kuhn simplex is the one that generates the minimum number of similarity classes. That is because marked bisection achieves its optimal number of similarity classes with structured meshes, which are fully composed of Khun simplices. Finally, the irregular simplex generates the maximum number of similarity classes due to its lack of symmetry. That is, it generates the highest number of similarity classes in comparison with the other simplices but does not achieves the maximum number of similarity classes. As the dimension increases, the number of similarity classes also increases in all the cases.

Table 2 shows the number of uniform refinements performed to generate the similarity classes of Table 1, and the a lower bound over the number of uniform refinements to generate S_n similarity classes. We see that the equilateral, the Cartesian, and the irregular simplices are equal or exceed the number of minimum uniform refinements to generate the similarity classes of Table 1. Moreover, the number of uniform refinements of the equilateral and Cartesian simplices is smaller than the irregular simplex for 4D and 5D. For the Kuhn simplex, we can see that the number of uniform refinements to achieve the generated number of similarity classes is n, except in the 2D case. This is so because the initial simplex is the unique similarity class. Generally, when the number of similarity classes becomes larger, we need to perform more uniform refinements to generate them.

In all the cases, the estimated and the obtained number of similarity classes are similar. Note that the predicted number of similarity classes for marked bisection is larger that the number of similarity classes of newest vertex bisection. Moreover, the ratio of similarity classes between marked bisection and newest vertex bisection grows exponentially with the dimension. While we obtain the same number of similarity classes for dimension two, there is a factor of eight for dimension five. The difference between the number of bisection steps to obtain all the similarity classes in marked bisection and newest vertex bisection grows linearly with the dimension.

7. CONCLUDING REMARKS

To measure the stability of marked bisection, we have estimated in general dimensions an upper bound of the number of obtained similarity classes. Moreover, to understand the cyclic structure of similarity, we have estimated the number of uniform refinements required to generate all the similarity classes. These estimates facilitate comparing marked bisection with the newest vertex bisection.

This comparison is key because the newest vertex bisection is the optimal reference for the number of generated similarity classes. First, we compare the ratio of the number of similarity classes between marked bisection and the newest vertex bisection. This ratio grows exponentially with the dimension as $\mathcal{O}(2^{n-2})$. Second, we compare the difference between the corresponding numbers of uniform refinements required to generate all the classes. This difference grows linearly with the dimension as $n-2$.

According to the scalings, we conclude that when lower is the dimension more suitable is marked bisection for local refinement of unstructured meshes. We also conclude that marked bisection is still the right choice for unstructured meshes. The scalings seem to favor the newest vertex bisection, but it has not been yet guaranteed for unstructured meshes.

Although the two estimates are not tight, our results show that they match the magnitudes and scalings with the dimension. To tighten the bounds, we only need to improve the similarity bound because the number of iterations depends on the former bound. Accordingly, we have planned to improve the similarity bound by accounting for the possible simplicial symmetries that may arise during the first refinements of marked bisection.

In perspective, the derived scalings further motivate the need to guarantee the newest vertex bisection for local refinement on unstructured meshes. In high-dimensional applications, the reduced number of similarity classes of the newest vertex bisection will lead to higher mesh quality and quicker starts of the similarity cycles.

8. ACKNOWLEDGEMENTS

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 715546. This work has also received funding from the Generalitat de Catalunya under grant number 2017 SGR 1731. The work of Xevi Roca has been partially supported by the Spanish Ministerio de Economía y Competitividad under the personal grant agreement RYC-2015-01633.

Algorithm 9 Local refinement of a marked mesh.

input: `ConformalMarkedMesh` $\mathcal{T}$ and `SimplicesSet` $\mathcal{S} \subset \mathcal{T}$
output: `ConformalMarkedMesh` $\mathcal{T}'$

```
function localRefine(T, S)
    T̄ = bisectSimplices(T, S)
    T' = refineToConformity(T̄)
    T_0 = renumberMesh(T')
    return T_0
end function
```

Algorithm 10 Refine-to-conformity a marked mesh.

input: `MarkedMesh` $\mathcal{T}$
output: `MarkedMesh` $\mathcal{T}'$ without hanging vertices

```
function refineToConformity(T)
    S = getNonConformalSimplices(T)
    if S ≠ ∅ then
        T̄ = bisectSimplices(T, S)
        T' = refineToConformity(T̄)
    else
        T' = T
    end if
    return T'
end function
```

Algorithm 11 Bisect a set of simplices.

input: `MarkedMesh` $\mathcal{T}$, `SimplicesSet` $\mathcal{S}$
output: `MarkedMesh` $\mathcal{T}_1$

```
function bisectSimplices(T, S)
    T_1 = ∅
    for ρ ∈ T do
        if ρ ∈ S then
            ρ_1, ρ_2 = bisectSimplex(ρ)
            T_1 = T_1 ∪ ρ_1
            T_1 = T_1 ∪ ρ_2
        else
            T_1 = T_1 ∪ ρ
        end if
    end for
    return T_1
end function
```

A. ALGORITHMS

In this appendix, we declare the necessary algorithms to implement a marked bisection method, as seen in Section 2.2. Using the conformingly-marked mesh, the local refinement procedure, Algorithm 9, first refines a set of simplices, then calls a recursive refine-to-conformity strategy, and finally renumbers the mesh. The refine-to-conformity strategy, Algorithm 10, terminates when successive bisection leads to a conformal mesh. Both algorithms use marked bisection to refine a set of elements, see Algorithm 11.

References

[1] Rivara M.C. "Algorithms for refining triangular grids suitable for adaptive and multigrid techniques." *International Journal for Numerical Methods in Engineering*, vol. 20, no. 4, 745–756, 1984

[2] Rivara M.C. "Local modification of meshes for adaptive and/or multigrid finite-element methods." *Journal of Computational and Applied Mathematics*, vol. 36, no. 1, 79–89, 1991. Special Issue on Adaptive Methods

[3] Plaza A., Carey G.F. "Local refinement of simplicial grids based on the skeleton." *Applied Numerical Mathematics*, vol. 32, no. 2, 195–218, 2000

[4] Plaza A., Rivara M.C. "Mesh Refinement Based on the 8-Tetrahedra Longest-Edge Partition." *Proceedings of the 12th International Meshing Roundtable*, pp. 67–78. 2003

[5] Mitchell W.F. "Adaptive refinement for arbitrary finite-element spaces with hierarchical bases." *Journal of Computational and Applied Mathematics*, vol. 36, no. 1, 65–78, 1991. Special Issue on Adaptive Methods

[6] Kossaczký I. "A recursive approach to local mesh refinement in two and three dimensions." *Journal of Computational and Applied Mathematics*, vol. 55, no. 3, 275–288, 1994

[7] Maubach J.M. "Local Bisection Refinement for N-Simplicial Grids Generated by Reflection." *SIAM Journal on Scientific Computing*, vol. 16, no. 1, 210–227, 1995

[8] Maubach J.M. "The efficient location of neighbors for locally refined n-simplicial grids." *5th International Meshing Roundable*, vol. 4, no. 6, 137–153, 1996

[9] Traxler C.T. "An algorithm for adaptive mesh refinement in n dimensions." *Computing*, vol. 59, no. 2, 115–137, 1997

[10] Belda-Ferrín G., Ruiz-Gironés E., Roca X. "Bisecting with optimal similarity bound on 3D unstructured conformal meshes." *2022 SIAM International Meshing Roundtable (IMR), Virtual Conference*. Zenodo, 2021

[11] Arnold D.N., Mukherjee A., Pouly L. "Locally Adapted Tetrahedral Meshes Using Bisection." *SIAM Journal on Scientific Computing*, vol. 22, no. 2, 431–448, 2000

[12] Belda-Ferrín G., Gargallo-Peiró A., Roca X. "Local Bisection for Conformal Refinement of Unstructured 4D Simplicial Meshes." *27th International Meshing Roundtable*, vol. 127, pp. 229–247. Springer International Publishing, 2019

[13] Belda-Ferrín G., Ruiz-Gironés E., Gargallo-Peiró A., Roca X. "Conformal Marked Bisection for Local Refinement of n-Dimensional Unstructured Simplicial Meshes." *Computer-Aided Design*, p. 103419, 2022

[14] Gutierrez C., Gutierrez F., Rivara M.C. "Complexity of the bisection method." *Theoretical Computer Science*, vol. 382, no. 2, 131–138, 2007

[15] Aparicio G., Casado L.G., Hendrix E.M., G-Tóth B., Garcia I. "On the minimum number of simplex shapes in longest edge bisection refinement of a regular n-simplex." *Informatica*, vol. 26, no. 1, 17–32, 2015

[16] Knupp P.M. "Algebraic Mesh Quality Metrics." *SIAM Journal on Scientific Computing*, vol. 23, no. 1, 193–218, 2001

[17] Liu A., Joe B. "On the Shape of Tetrahedra from Bisection." *Mathematics of Computation*, vol. 63, no. 207, 141–154, 1994