Geometric Properties of 3D Bisection Refinement

Antti Hannukainen

Institute of Mathematics and Systems Analysis
Aalto University

Institute for analysis and scientific computing
TU Wien
Overview

Interest: Prove that the family of partitions generated by 3D bisection refinement is strongly regular (known under restrictions in 2D).
Overview

Interest: Prove that the family of partitions generated by 3D bisection refinement is strongly regular (known under restrictions in 2D).

\[ \forall T \in \mathcal{T}_h, |T| > C h^d \]

\[ |u - \pi u|_{1,K} \leq C \| A_K^{-1} \| \| A_K \|^2 |u|_{2,K}. \]
Overview

Interest: Prove that the family of partitions generated by 3D bisection refinement is strongly regular (known under restrictions in 2D).

Bisection algorithm: Bisection in 3D, efficient implementation.

Test for shape similarity in 3D: Easy method for testing when two tetrahedrons have the same shape.

Computational examples: Cube-corner, Sommerville, and regular tetrahedron.
Bisection algorithm 3D
Bisection algorithm 3D

Nodes of the tetrahedron not shared with the longest edge

Nodes of the longest edge

Children are and
Short word on anisotropic refinement

A weight function can be applied for the edge length computation: easy to produce non-uniformly refined meshes: a posteriori error control.
Bisection algorithm 3D: not so easy!

Which edge is longest and connected to which tetrahedron?

These connections change with each bisection: the size of the datastructures has to change dynamically.

Finding the longest edge: non-trivial when lot of edges are in the mesh.
Find longest edge: store edge index sorted by length (priority queue).

Find tetrahedron related to edge: keep edge index-to-tet list e2t. e.g. in array with edge index as the key.

Updating edge index-to-tet list is tricky: Size changes on each step. Some tetrahedron need to be deleted, some remain. Edge index cannot be updated.
Bisection algorithm 3D: not so easy!

Fortunately: The refinement process can be still analyzed locally!
Bisection algorithm 3D : not unique
Bisection algorithm 3D: not unique
Bisection algorithm 3D: not unique
Bisection algorithm 3D: not unique
Bisection algorithm 3D: not unique

If several edges of same length are in the mesh, the bisection may not be unique:
This is the case for regular tetrahedron.

Does this have an impact on the shapes generated in the refinement? (in my experience - yes)
How to make the refinement process well defined?
Shape test

Definition: **Tetrahadron** $A$ is similar to $B$, if $A$ can be obtained from $B$ using rotation, scaling, or mirroring.

This is, there exists an affine mapping $F$ such that

$$F(x) = Kx + b$$

$$K = cQ, \quad Q^TQ = I$$

Unitary matrix: a rotation or an improper rotation
Shape test

1. Construct affine mapping from $A$ to $B$
2. Check if $DF = cQ$, $Q^TQ = I$

$$C = DF^TDF$$

$$\left\| I - \frac{C}{C_{11}} \right\| = 0$$

First idea - divide with a small number?
Shape test

1. Construct affine mapping from \( A \) to \( B \)
2. Check if \( DF = cQ, Q^T Q = I \)

\[
C = DF^T DF
\]
\[
\left\| I - \frac{C}{C_{11}} \right\| = 0 \quad \text{First idea - divide with a small number!}
\]
\[
\|I\|\|C\| - C\|I\|\| = 0 \quad \text{Second idea - more robust}
\]
\[
C = cI \Rightarrow \|C\| = c\|I\|
\]
How to choose tolerance? \[ \| I \| \tilde{C} \| - \tilde{C} \| I \| \| \leq tol \]

Reality: Due to rounding errors etc. we never get \( C \). Instead, we are given \( \tilde{C} = C + E \), \( \| E \| \) small
How to choose tolerance? \[ \left\| I \| \tilde{C} \| - \tilde{C} \| I \| \right\| \leq tol \]

Reality: Due to rounding errors etc. we newer get \( C \).
Instead, we are given \( \tilde{C} = C + E \), \( \| E \| \) small

We obtain
\[
\left\| I \| C \| - C \| I \| \right\| = \left\| I \| C \| - I \| \tilde{C} \| + E \| I \| + I \| \tilde{C} \| - \tilde{C} \| I \| \right\|
\]
\[
\left\| I \| \tilde{C} \| - \tilde{C} \| I \| \right\| - \alpha \leq \left\| I \| C \| - C \| I \| \right\| \leq \left\| I \| \tilde{C} \| - \tilde{C} \| I \| \right\| + \alpha
\]

\[ \alpha = \left\| I \| C \| - I \| \tilde{C} \| + E \| I \| \right\| \leq (1 + \| I \|) \| E \| \]
How to choose tolerance?

Reality: Due to rounding errors etc. we never get $C$. Instead, we are given $\tilde{C} = C + E$, $\|E\|$ small.

We obtain

Tolerance can be in the range of rounding errors.

$$\alpha \leq (1 + \|I\|)\|E\|$$
Construction of affine mapping

\[ F_{A,B} : \]

Construction of affine mapping

\[ F_{A,B} = F_{\hat{K},B} \circ F_{\hat{K},A}^{-1} \]
Construction of affine mapping

Permutations: The affine mapping between tetrahedrons is not unique. There is $4! = 4 \times 3 \times 2 = 24$ permutations.

Test for each permutation is required.
Construction of affine mapping

Permutations: The affine mapping between tetrahedrons is not unique. There is \( 4! = 4 \times 3 \times 2 = 24 \) permutations.

Test for each permutation is required.
Construction of affine mapping

Permutations: The affine mapping between tetrahedrons is not unique. There is $4! = 4 \times 3 \times 2 = 24$ permutations.

Test for each permutation is required.
Construction of affine mapping

Permutations: The affine mapping between tetrahedrons is not unique. There is $4! = 4 \times 3 \times 2 = 24$ permutations.

Test for each permutation is required.
Construction of affine mapping

Permutations: The affine mapping between tetrahedrons is not unique. There is $4! = 4 \times 3 \times 2 = 24$ permutations.

Test for each permutation is required.

$$F_{A,B} = F_{K,B} \circ F_{K,A}^{-1}$$

To keep the computational cost low, do permutations here. This way, the inverse can be reused for each permutation.
The Algorithm

Keep a list of found shapes

When a tetrahedron is bisected: do a similarity for the children.

Similarity test: Construct affine mapping between each shape and children for each permutation of nodes. Do the test using affine mappings.

When new shape is found: Add to list of shapes
Numerical examples

Corner tetrahedron \((0,0,0),(1,0,0),(0,1,0),(0,0,1)\): yields 8 shapes. No new shapes after bisection step 52.

Complicated: many edges of the same length. Uniqueness.
Numerical examples - random perturbation to corner tetrahedron

Perturbation 0.1 20 samples: max. 61 / min. 30 / mean 40.5

Perturbation 0.2 20 samples: max. 56 / min. 30 / mean 39.6

Perturbation 0.3 20 samples: max. 78 / min. 33 / mean 44.75

The results were computed with 20 000 bisections (leads to about 100 000 tetrahedron, 4.5 sec / run) - results have not yet converged!
No convergence even with 200,000 bisections: something else needed!

Local nature of bisection helps!

The mesh can be omitted: Consider only single tetrahedron

Keep a list of all shapes and un-bisected shapes

Bisect un-bisected shapes: remove from list and do a similarity for the children.

When new shape is found: Add to list of shapes and to un-bisected shapes.
Numerical examples - random perturbation to corner tetrahedron

Different algorithm: random perturbation of 0.01 using 20 shape refinement steps

not converged, num of shapes 646
not converged, num of shapes 993
converged, num of shapes: 36
not converged, num of shapes 851
converged, num of shapes: 36
not converged, num of shapes 994
not converged, num of shapes 851
not converged, num of shapes 526
not converged, num of shapes 1025
not converged, num of shapes 521
not converged, num of shapes 1001
converged, num of shapes: 36
Numerical examples

Sommerfeld tetrahedron \((-1,0,0),(1,0,0),(0,-1,1),(0,1,1)\): yields 4 shapes.
Numerical examples - Regular tetrahedron

The regular tetrahedron (hard):

num of shapes:8  num of tet:4222  num bisections: 1000
num of shapes:8  num of tet:25032  num bisections: 5000
num of shapes:32 num of tet:47719  num bisections: 10 000
num of shapes:34 num of tet:97377  num bisections: 20 000
num of shapes:35 num of tet:150957 num bisections: 30 000
num of shapes:35 num of tet:266125 num bisections: 50 000
num of shapes:35 num of tet:362460 num bisections: 70 000
num of shapes:38 num of tet:503859 num bisections: 100 000
num of shapes:40 num of tet:1599551 num bisections: 300 000
Numerical examples - Regular tetrahedron

The regular tetrahedron (hard):

N=20 not converged, num of shapes 994
N=30 not converged, num of shapes 2549

(num of shapes:40  num of tet:1599551  num bisections: 300 000 ), so huge meshes!
Results not conclusive - need to improve code, now just fast :)

Bisection of similar tetrahedron leads to same shapes:
when same shape appears for the second time -
eliminate form list of tetrahedron.

This will also give conclusive stopping criterion: If all
shapes in the mesh have been divided once and no new
ones is produced, stop.

We can work only with shapes: No need for data
structures! Uniqueness is the current difficulty.