Universitat Politècnica de Catalunya
ETSECCPB

Treball Fi de Màster

Simultaneous Untangling and Smoothing of Hexahedral Meshes

by

Thomas James Wilson

Advisors: Josep Sarrate Ramos and Xevi Roca Navarro

Barcelona, June 2011

‘‘I hate meshes. I cannot believe how
hard this is. Geometry is hard.’’
--David Baraff, Senior Research Scientist,
Pixar Animation Studios

Abstract

Simultaneous Untangling and Smoothing of Hexahedral Meshes

Thomas James Wilson

A new hexahedral mesh untangler and smoother is developed using global optimisation. The method is compared with an existing local optimisation smoother and found to converge on the same improved mesh. The result validates the technique of smoothing meshes node by node, bypassing the issues associated with solving large systems of linear systems.

The Geometric Element Transform Method is compared with the new smoother as well as existing techniques. While it is not guaranteed to untangle the mesh, the GETMe method is typically able to increase the minimum and mean qualities of meshes in a time span at least an order of magnitude below that of the optimisation smoother. Converged qualities are below those of the optimisation method but above Laplacian smoothing. The mesh obtained from the Geometric Element Transform Method can be examined and further smoothed with an optimisation method only if its quality is insufficient.

Keywords: Hexahedral mesh smoothing; Hexahedral mesh untangling; Mesh optimisation.

Acknowledgments

First and foremost I must thank my tireless advisors Josep Sarrate and Xevi Roca. I have received every conceivable support from these two men. Along with laudable academic advice and insightful guidance, I have spent the last year with the use of a desk and computer within meters of their respective offices and have therefore consumed an inordinate amount of their time.

Thanks to everyone in LaCàN, but special thanks to two. Eloi Ruiz has a deep understanding of C++, ez4u, and AVS and has been kind enough to share a fraction of this knowledge with me. If any figure in this document looks good it's because Eloi showed me how to make it. If any are not it's because I didn't do as he suggested.

Abel Gargallo Peiró's area of research is surface meshes but a large amount of overlap can be found with the solid domain. Abel had been thinking about δ values and how to vectorise optimisation routines in Matlab for many months before I began work and was therefore able to give me a significant head start.

Thanks to the European Union for giving me some money, and thanks to Casa Bou in Collblanc for taking it.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

The accuracy and performance of a finite element analysis depends on the quality of the mesh on which the spatial domain has been discretised. Most three dimensional meshes are composed of tetrahedral elements as they can be automatically generated to discretise an arbitrary domain by rapid, mature algorithms (Tournois et al., 2009). While more challenging to mesh, hexahedral elements have several advantages over tetrahedra (Roca, 2009).

More Accurate For a given number of degrees of freedom, hexahedral meshes have been shown to be more accurate in empirical structural analyses.

More Robust Hexahedral elements are less likely to encounter shear locking.

Better Anisotropy As elemental aspect ratios increase well beyond unity (as is desirable, for example, in boundary layer fluid dynamics problems and composite solid problems) hexahedral elements are able to better approximate the true solution.

Unfortunately, ready access to these advantages is hampered by the significant challenges regarding the generation of the hexahedral mesh. At this stage no automated technique exists which compares to tetrahedral methods such as Delaunay refinement.

Techniques which do exist operate over a smaller domain and in a less robust manner. Meshes generated automatically can include poorly formed or even inverted elements (Ledoux, 2008). It is therefore of value to develop techniques for the smoothing and untangling of hexahedral meshes.

As available computing power increases an ever greater number of problems become tractable. One such class of problems is the smoothing of meshes using minimisation methods to find the globally optimal node configuration.

1.2 Objectives

The process of moving the nodes of a mesh to improve the overall quality of the elements is termed as *smoothing*, while the process of transforming inverted elements to correctly oriented ones is called *untangling*. Both will be addressed herein, although the connectivity of the mesh will remain unchanged. Therefore, the smoothers and untanglers will only modifiy the location of interior nodes.

The objective of this thesis is to develop and compare algorithms for the smoothing and untangling of purely hexahedral meshes in $\mathbb{R}^3$. Compared algorithms fall into two principle categories,

Heuristic Algorithms which operate by the geometrical manipulation of nodal positions. Such methods are typically quick running. Examples include Laplacian smoothing (Frey et al., 2007) and GETMe (Vartziotis and Wipper, 2011).

Minimisation Algorithms which search for a minimum of an objective function, potentially using its first and second derivatives. These methods typically generate a higher quality mesh at a greater computational expense. (Escobar et al., 2003; Knupp, 2003)

As well as comparing existing mesh smoothing methods, a new, global approach to the minimisation problem will be developed. The method will be able to smooth

and untangle hexahedral meshes simultaneously by generating a global system of equations.

1.3 Main Contributions

A global, all hexahedral combined smoother and untangler has been developed using functional optimisation. The smoother has been compared to a local smoother based on the same function as well as heuristic smoothers.

Local and Global objective function based smoothers have been shown to converge on the same nodal configuration. This validates local smoothers as a robust method of optimising the quality of hexahedral meshes.

The Geometric Element Transform Method has been shown to provide a high quality mesh as well as an effective initial guess to the optimisation procedure.

Chapter 2

Background

2.1 Quality Index

In order to improve the quality of the mesh we must define a way to measure it. The quality of a hexahedron will be based on that of the tetrahedra composed within.

2.1.1 Requirements of the Quality Index

The measure used should satisfy the definition given in (Dompierre et al., 1998), with the exception of the maximum corresponding to a regular tetrahedron. Our maximum quality corresponds to the tetrahedron forming the corner of a regular hexahedron.

> A tetrahedron shape measure is a continuous function that evaluates the quality of a tetrahedron. It must be invariant under translation, rotation, reflection and uniform scaling of the tetrahedron. It must be maximum for the regular tetrahedron and it must be minimum for a degenerate tetrahedron. There is no local maximum other than the global maximum for a regular tetrahedron and there is no local minimum other than the global minimum for a degenerate tetrahedron. For the ease of comparison, it should be scaled to the interval $[0, 1]$, and be 1 for the regular tetrahedron and 0 for a degenerate tetrahedron.

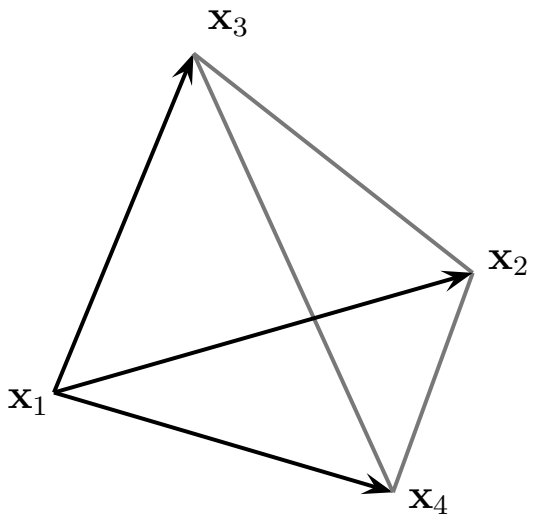

Figure 2.1: Tetrahedral Edges Defining Vector Space

2.1.2 Affine Transforms Between Elements

Throughout this section bold symbols $\mathbf{x_1}$ denote column vectors of coordinates in $\mathbb{R}^3$, bold capitals $\mathbf{A}$ denote matricies and italic symbols x_1 denote individual scalar components.

Consider an arbitrary tetrahedral element. A vector space exists defined by the edges prodruding from the first node $\mathbf{x_1}$ to each of the remaining nodes, $\mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}$. These are shown in Figure 2.1.

$$\{\mathbf{x_2} - \mathbf{x_1} \qquad \mathbf{x_3} - \mathbf{x_1} \qquad \mathbf{x_4} - \mathbf{x_1}\} \tag{2.1}$$

Linearity holds between such spaces definied on different elements. We can define an affine mapping between two elemental vector spaces using a linear transformation matrix and a translation. These affine transormations are defined in reference to three nodal configurations,

Physical Element The element defined by the nodes as they are in the mesh. Coordinates in this space are denoted by the symbol $\mathbf{x}$.

Ideal Element The desired nodal configuration. Coordinates denoted by $\mathbf{x_i}$.

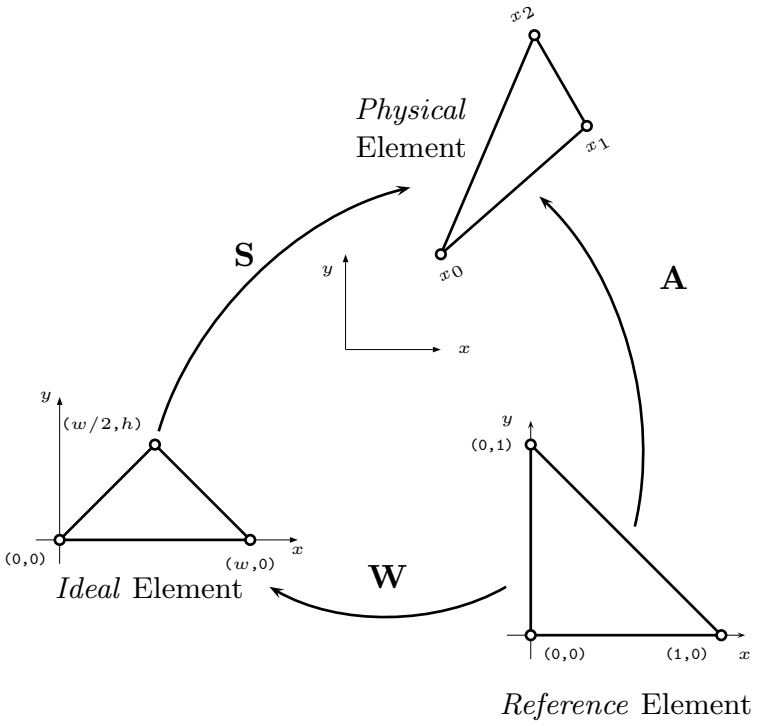

Figure 2.2: Transformations between elements (Rivas, C. A., 2010).

Reference Element The configuration shown in Figure 2.3, with unit vectors aligned with the cartesian space (x, y and z axes). Coordinates denoted by $\mathbf{x_r}$.

Transformation from the *reference* to the *physical* element is done by (2.2). Transformation from the *reference* to *ideal* element is done by (2.3), while transformation from the *ideal* to *physical* element is done with (2.4). The transformations are shown in Figure 2.2

$$\mathbf{x} = \mathbf{A}\mathbf{x_r} + \mathbf{x_1} \tag{2.2}$$

$$\mathbf{x_i} = \mathbf{W}\mathbf{x_r} \tag{2.3}$$

$$\mathbf{x} = \mathbf{S}\mathbf{x_i} + \mathbf{x_1} \tag{2.4}$$

Nodal positions of a tetrahedron make up the matrix $\mathbf{A}$ according to (2.5). The first

tetrahedral node is the base of the tetrahedron, at which all are mutually perpendicular in the ideal case. Coordinates used are those of the physical element.

$$\mathbf{A} = \begin{bmatrix} x_2 - x_1 & x_3 - x_1 & x_4 - x_1 \\ y_2 - y_1 & y_3 - y_1 & y_4 - y_1 \\ z_2 - z_1 & z_3 - z_1 & z_4 - z_1 \end{bmatrix} \tag{2.5}$$

Now we demonstrate the use of affine transformation (2.2) to transform the third node of the reference element to the physical space,

$$\begin{aligned} & \mathbf{A} \cdot (\mathbf{x_{i3}}) + \mathbf{x_1} \\ = & \begin{bmatrix} x_2 - x_1 & x_3 - x_1 & x_4 - x_1 \\ y_2 - y_1 & y_3 - y_1 & y_4 - y_1 \\ z_2 - z_1 & z_3 - z_1 & z_4 - z_1 \end{bmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \\ = & \begin{pmatrix} x_3 - x_1 \\ y_3 - y_1 \\ z_3 - z_1 \end{pmatrix} + \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \\ = & \begin{pmatrix} x_3 \\ y_3 \\ z_3 \end{pmatrix} \\ = & \mathbf{x_3} \end{aligned}$$

In this work the reference and ideal elements are the same since the ideal tetrahedron is one which makes up a corner of a cube. For tetrahedral meshes the ideal element is the regular tetrahedron. We therefore simplfy with $\mathbf{W} = \mathbf{I}$.

$$\mathbf{S} = \mathbf{A}\mathbf{W}^{-1} \tag{2.6}$$

$$\mathbf{S} = \mathbf{A} \tag{2.7}$$

2.1.3 Tetrahedral Quality Measures

We use the measure given by (Knupp, 2000).

$$q(T) = \begin{cases} \dfrac{3\sigma^{2/3}}{|\mathbf{S}|^2} & \sigma > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

$$\sigma = \det \mathbf{S} \tag{2.9}$$

2.1.4 Hexahedral Quality Measures

The quality of a hexahedron is defined by the average quality of eight tetrahedra composed of its nodes (Vartziotis and Wipper, 2011).

$$q(H) = \frac{1}{8}\sum_{i=1}^{8} q\left(T_i\right) \tag{2.10}$$

Note this differs from the quality measure used in (Knupp, 2001; Rivas, C. A., 2010). The node on each corner of the hexahedron, as well as the other three with which it shares an edge, form an element. The element's quality is determined from Section 2.1.3. The permutations used to extract these tetrahedra are given in Table 2.1.

2.1.5 Combining Elemental Qualities

We will consider two primary metrics to evaluate the quality of a whole mesh from the quality of the elements.

$$Q_{min} = \min\left(q_i\right) \tag{2.11}$$

$$Q_{mean} = \frac{1}{n}\sum_{i=1}^{n} q_i \tag{2.12}$$

The set of elements $i = \{1, ..., n\}$ is all hexahedral elements in the mesh.

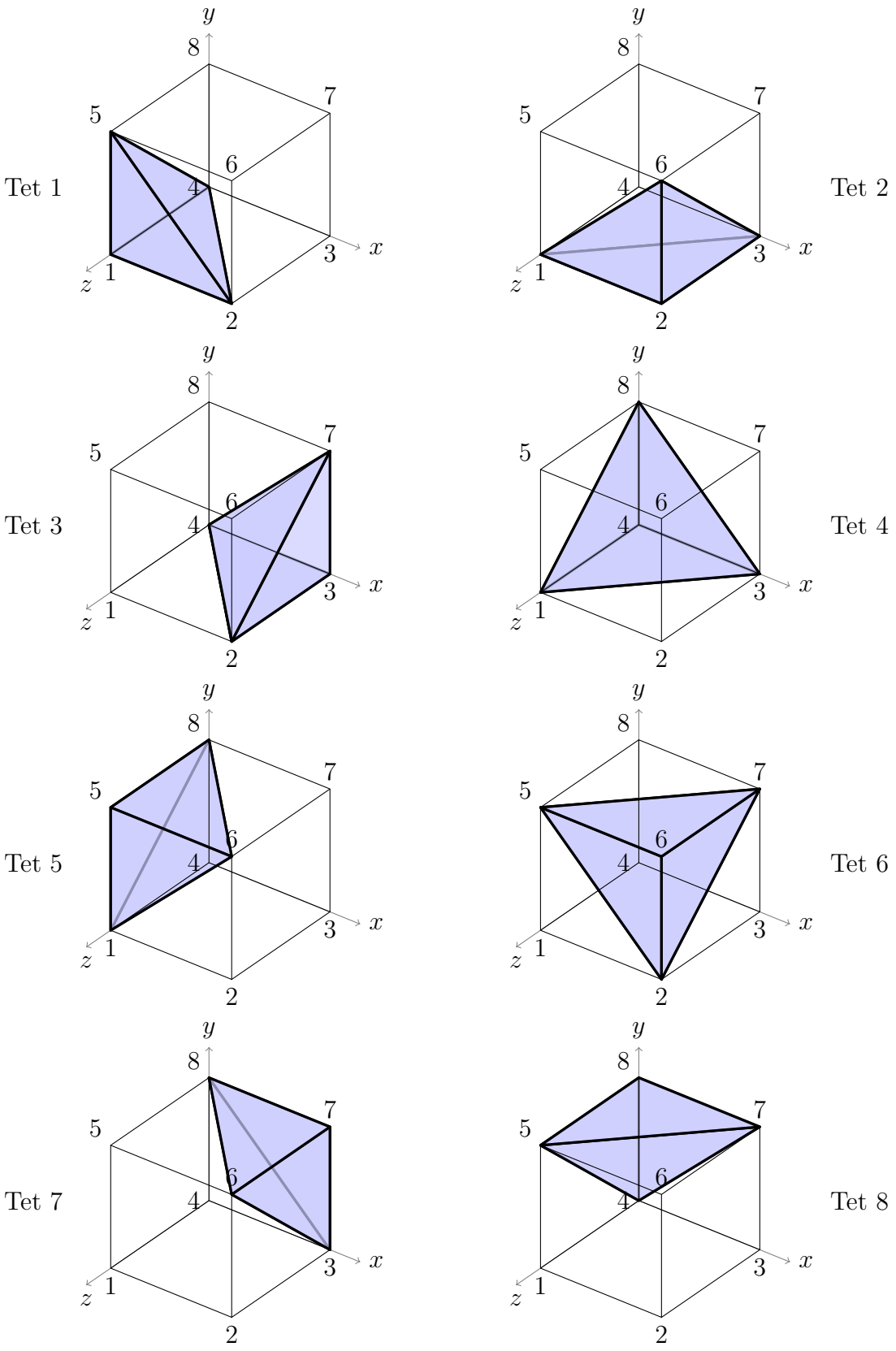

Figure 2.3: Tetrahedra used to calculate the quality of a hexahedral element. Indices from Table 2.1.

Element	Node			
	1	2	3	4
1	1	4	5	2
2	2	1	6	3
3	3	2	7	4
4	4	3	8	1
5	5	8	6	1
6	6	5	7	2
7	7	6	8	3
8	8	7	5	4

Table 2.1: Nodes of hexahedon used to form eight sub-tetrahedra as used to generate Figure 2.3

2.2 State Of The Art

Mesh smoothing and untangling is a rapidly evolving field, here we summarise some important recent results. A number of the citations given here pertain to tetrahedral meshes, on which most of the literature has been written.

2.2.1 Heuristic Methods

The Geometric Element Transform Method (Vartziotis and Wipper, 2011) regularises a hexahedron using its dual octahedron. Weighting the new node position based on the quality of the elements which contain it often results in a mesh of higher quality than would be obtained from Laplacian smoothing. In addition, the method typically approaches convergence more rapidly. The method includes a check which reverts any node movement leading to an inverted element, which can occur if elements have large aspect ratios.

2.2.2 Optimisation Methods

Optimisation methods smooth the mesh by defining an objective function inversely related to the qualities of elements. This objective function is minimised, at which point the quality is at its highest.

One method with which this can be done is to use a single objective function for the whole mesh. The movement of all nodes are considered simultaneously, as is the effect of this movement on the qualities of all elements. This approach, as used on hexahedral elements in (Knupp, 2003) has the advantage that the problem being solved is directly related to the desired outcome. If Newton based methods are called upon (making use of the second derivatives), difficulties can emerge on account of the large linear system and the fact that the Hessian matrix is not always positive definite. An example of a convex triangular mesh with just two free nodes and an indefinite Hessian can be found in (Munson, 2007), in which the global smoothing of an untangled tetrahedral mesh is investigated.

The issues of large systems are bypassed if the smoothing is done on a node by node basis, in which sequential single nodes in the mesh are considered free while all others are fixed (Sastry and Shontz, 2009). Such methods do not require the solution of large systems of equations but have other drawbacks. If a large group of nodes lie a long distance from their optimal position they require several iterations to approach the optimal value since the progress is hindered by the artificial boundary conditions of the surrounding 'fixed' nodes.

A comparison between these two broad methods as applied to tetrahedral meshes can be found in (Freitag et al., 2006). The authors report that while an optimal mesh is typically found more quickly using the global approach, good progress can be made by local methods (referred to as coordinate descent) in the time spent setting up the global one (for example, computing and allocating the sparsity pattern for the global Hessian). The authors also note the advantages of global methods are diminished as the element size heterogeneity increases.

Untangling has traditionally been done separately to smoothing (Knupp, 2000). Modification of the objective function to facilitate untangling while retaining satisfactory

local smoothing was proposed for tetrahedral meshes in (Escobar et al., 2003), and extended to hexahedral meshes by (Rivas, C. A., 2010). In this work we will extend the combined untangling and smoothing of hexahedral meshes to a global approach.

Chapter 3

Mesh Quality Improvement

The methods detailed here share the convergence criterion of Algorithm A.1. The GETMe method also requires an increase in either minimum or mean quality.

3.1 Heuristic Methods

3.1.1 Laplacian Smoothing

Laplacian Smoothing updates the position of a node to the average of the nodes directly connected to it by an edge. It is computationally cheap to perform an iteration, but it can generate inverted elements in concave domains. A *Smart Laplace* variant exists which checks all elements affected by a node relocation and reverts the change if any have become inverted.

Algorithm 3.1 Laplacian Smoothing

$nodes \leftarrow \{\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \ldots, \mathbf{x_n}\}$ Set of interior nodes
repeat
  for $\mathbf{x_i} \in nodes$ **do**
    $a \leftarrow$ nodes attached by an edge to $\mathbf{x_i}$
    $\mathbf{x_i} \leftarrow \frac{1}{N} \sum_j^N \mathbf{a_j}$
  end for
until Convergence using Algorithm A.1

3.1.2 Geometric Element Transform Method Simultaneous Smoothing

The *GETMe Simultaneous* method has ben implemented unchanged from its original presentation (Vartziotis and Wipper, 2011).

Note that the full GETMe smoothing method is a two stage process, a *simultaneous* smoothing of all nodes followed by *sequential* smoothing of the lowest quality elements. The second stage involves finding the lowest quality element and regularising it. This portion of the algorithm is not vectorisable, and also requires the maintenance of a sorted mapping of qualities to elements.

On meshes of the size examined in this work it was not feasible to run the *sequential* GETMe algorithm. We have implemented it in C++ for inclusion in the ez4u meshing framework and can confirm the algorithm improves the minimum quality of all meshes on which it has been tested. Numerical results will not be included since they would entail a comparison between MATLAB and C++.

3.2 Optimisation Methods

Meshes to be smoothed using optimisation methods can optionally be preconditioned by a heuristic method.

3.2.1 Objective Function

We base the objective function on the inverse of the elemental quality functions we wish to maximise (2.10), combined with the generalised mean. The same function is used for local and global smoothing operations. The set of elements over which the sum operates S_{elems} can be those surrounding a node or all elements in the mesh.

$$\eta = \frac{|\mathbf{S}|^2}{3h\,(\sigma)^{2/3}} \tag{3.1}$$

$$h\,(\sigma) = \frac{1}{2}\left(\sigma + \sqrt{\sigma^2 + 4\delta^2}\right) \tag{3.2}$$

$$\eta_{global} = \left(\sum_{i=1}^{n} {\eta_i}^p\right)^{1/p} \qquad \eta_i \in S_{elems} \tag{3.3}$$

Using a value of $p = 1$ in (3.3) defines the accumulated quality as the mean of the elemental contributions. In this work we will use $p = 2$, giving us the quadratic mean. Such a value causes the algorithm to more heavily weight the lowest quality elements which have the highest value of η.

The modification from $h(\sigma) = \sigma$ given in (3.2) was proposed by (Escobar et al., 2003) and allows us to untangle as well as smooth the mesh. It is important to note that the inclusion of a non-zero δ prevents the inverse of η being a quality metric as defined in Section 2.1.1 since it is no longer size independent. An explanation of its function is given in Section 3.2.4.

Methods below require first and sometimes second derivatives to operate. Refer to Appendix B where there are reported or developed where necessary.

Local Optimisation

To perform a local optimisation of a single node, the elements in S_{elems} in (3.3) are the elements attached to the node in question. All other nodes in the mesh are fixed. The steps are given in Algorithm 3.2.

Global Optimisation

Smoothing of all nodes in the mesh simultaneously is done by assigning all mesh elements to S_{elems} in (3.3). The steps are given in Algorithm 3.3.

Algorithm 3.2 Local Optimisation

```
S_nodes ← interior nodes
iters ← 0
while unconverged and iters < maxIters do
  for all x_i ∈ S_nodes do
    Apply local smoothing to node x_i
  end for
  iters ← iters + 1
end while
```

We need to ensure that the local and global methods agree on the location of the optimal point. This is done in Appendix C.

Algorithm 3.3 Global Optimisation

```
iters ← 0
while unconverged and iters < maxIters do
  Apply global smoothing to interior nodes
  iters ← iters + 1
end while
```

3.2.2 Search Direction Methods

We implement the most simple search direction methods since the primary objective is to compare the broad local and global methods rather than implementation details, which can be found in Appendix A.

First Derivative Methods

A number of first order methods exist which approximate the Hessian using successive gradient vectors. These are not used in local smoothing operations because successive iterations of search direction are interspersed with the movement of adjacent nodes, rendering past results invalid. Performing multiple iterations on a single free node was not found to be an economical technique.

Steepest Descent A new nodal position is sought in the opposite direction to the gradient at the current point. Algorithm A.2.

Second Derivative Methods

We set the derivative of the *quadratic model* to zero to obtain the search direction for Newton's Method. $\mathbf{B}$ is the Hessian or some approximation of it.

$$\begin{aligned} f(x_k + p) &\approx m_k(p) \\ m_k(p) &= f_k + p^T \nabla f_k + \frac{1}{2} p^T \mathbf{B} p \\ \frac{\partial m_k(p)}{\partial p} &= \nabla f_k + \mathbf{B} p = 0 \\ p &= -(\mathbf{B})^{-1} \nabla f_k \end{aligned} \tag{3.4}$$

Netwon Raphson By making the substitution $\mathbf{B} = \nabla^2 f_k$ in (3.4) we obtain a search direction p. Explicit steps can be found in (Nocedal and Wright, 1999) or Algorithm A.3.

Modified Newton Raphson We cannot rely on the global Hessian matrix to be positive definite, even when the mesh is in the feasible region (Munson, 2007). If the system is positive definite we have at our disposal a larger range of iterative solvers (Saad, 2003). As such, we increase the eigenvalues of the Hessian by the simple method of adding a constant to each term of the main diagonal. There are many alternative procedures for modifying the Hessian (Nocedal and Wright, 1999). For example, the one used by (Freitag et al., 2006) is the addition of a block diagonal. Our implementation is given in Algorithm A.4.

3.2.3 Step Size Selection

The selection of step size for all minimisation problems is done with a *Back Line Search* satisfying the *Armijo Condition* for descent. Specifics are listed in Algorithm A.5 in Appendix A

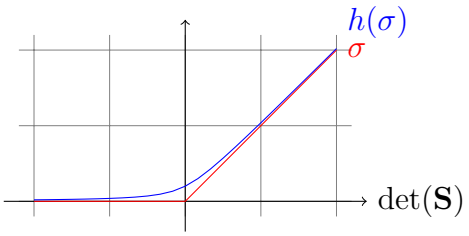

Figure 3.1: Modified σ with zero as an asymptote

3.2.4 Removing Barriers Caused by Tangled Elements

Using the inverse of the quality function (2.8) as the basis for our objective function causes problems with inverted elements. A tangled element, which has $\sigma = 0$ according to (2.8), gives an infinite objective function, preventing the method from functioning.

A modification to the objective function has been proposed (3.2) which prevents this barrier from occurring (Escobar et al., 2003). The modified function asymptotically approaches the unmodified function as $\det(\mathbf{S})$ increases and zero as it decreases. The modified and unmodified contributions can be seen in Figure 3.1. Larger values of δ entail larger deviations from the unmodified function.

δ is only assigned a non-zero value when the mesh is outside the feasible region. When all elements are untangled smoothing proceeds with an unmodified objective function.

Chapter 4

Results

4.1 Introduction

Here we present results for a three geometries of increasing complexity. Results will show the strengths and weaknesses of the smoothers.

BridgeSharp A *structured* mesh with a sharp concavity.

FiveSide A *semi-structured* convex mesh.

Plane An *unstructured* mesh of the outer domain of an aeroplane.

The smoothing was performed on the Linux Workstations with 4GB of RAM and an Intel Core2 Duo CPU E4800 running at 3.00GHZ. Smoothers were run in MATLAB 2007 on Ubuntu 8.04 LTS, which uses Linux kernel 2.6.24-26.

4.1.1 Notation

The following abeviations have been used in tables throught this chapter.

Lap Laplacian smoothing without checks for bad elements

GETMe Geometric Element Transform Method Simultanous smoothing

Init The initial mesh

SD Steepest Descent search direction, Algorithm A.2

NR Newton Raphson search direction, Algorithm A.3

MNR Modified Newton Raphson search direction, Algorithm A.4

Abbreviated terms used in Tables 4.1 to 4.13 are given below.

qMin Minimum element quality

qMean Mean elemental quality

qNorm Inverse Quadratic Norm of the elements. Equivalent to the global objective function when in feasible region.

$$qNorm = \frac{1}{\sqrt{\frac{1}{n}\sum_1^n \left(\frac{1}{q_i}\right)^2}}$$

time Smoothing Time in Seconds

iters Number of iterations to converge

timeIter Time per iteration

tIterNode Time per iteration per node

Optimisation methods which were unable to converge on the optimal solution in the allocated number of iterations are shown shown with a shaded background. The shortest remaining time is shown in **bold** face.

4.1.2 Result Caveats

There are a number of factors to bear in mind when analysing these results.

Vectorised Code The algorithms have been implemented in MATLAB, which suffers significantly in the presence of low level loops. The best performance is

obtained when the data is in large homogeneous arrays or matricies, coding in this manner led to a number of compromises. Further research is needed to determine the performance of the smoothers when implemented in a lower level language.

Local Implications Elemental node locations are duplicated across several different arrays for use in each of the local optimisation problems.

Global Implications Storage and assembly of the contributions of the global Hessian matrix consume approximately 50 times more memory than the Hessian itself, since for each global Hessian component there are aproximately 8 elements contributing 4 attached tetrahedra. Each of these 32 double float components also has a 32bit integer index consuming half the space. Assembly temporarily consumes even more.

To avoid running out of memory it was necessary to generate global Hessian chunks of 10000 to 30000 elements at a time and accumulate them.

Iterative Methods Solving the global linear system was done using MATLAB's 'minres' command to implement the *Minimum Residual Method* without preconditioning. The method was found to be the fastest suitable in the suite of MATLAB's iterative solvers. It is able to accommodate negative eigenvalues and takes into account the symmetry in the system.

It is expected that significant performance improvements could be made with a more optimised iterative solver.

4.1.3 General Comments

Some features of the optimisation smoothers are inherent to the methods and are presented ahead of numerical results.

Advantages of Global Optimisation

No Artificial Barriers Since the global smoother is able to move all the nodes at once, the convergence is not inhibited by the additional boundary conditions of surrounding nodes being stationary.

Better Derivative Component Reuse When calculating second derivatives we can reuse intermediate terms. This is not possible with the local smoother since movement occurs between the calcualtion of derivatives of each node of the element.

Advantages of Local Optimisation

No Large Linear Solutions Since the solver obtains solutions with only three degrees of freedom, solving large systems are not required. Increasing the number of nodes only increases the number of solutions required, not their size. The scaling is $\mathcal{O}(n)$ for this aspect of the solution. In addition to computational expense, solving many small systems instead of one large one requires less memory.

Second Order Search Direction Accuracy Directly solving the small linear system gives us a more accurate result than one derivied by iterative means.

Better Suited to Meshes of Variable Sized Elements Because each patch is dealt with separately, it is simple to add a step of scaling the patch. Doing so normalizes the influence of δ, which is not a function of the size of the element.

Simpler Derivative Calculations Holding all but three nodes constant simplifies the calculation of second derivatives, both in number of derivative components and the complexity of those components. See details in Appendix B.

Importance of Search Direction Accuracy

We can see that the *Global Steepest Descent* method has very slow convergence. The method only arrives at the optimal solution when smoothing the two smallest meshes

considered, *FiveSide4* and *FiveSide5* having 6119 and 14587 free nodes respectively. Search direction is very important for the global smoother. Since all the nodes are moved at the same time, any poorly selected component of the search direction will impeede the progress of the whole mesh.

Calculation of the partial first order derivatives in the global case is done by varying only the coordinate of interest. As such, the local and global first derivatives at a given point are identical. The additional freedom of having all nodes move simultaneously is not incorporated into the search direction, only it's length. To obtain a search direction which explicitly uses the movement of all nodes we must use a second order method.

When smoothing meshes locally it is not worthwhile spending time obtaining an accurate search direction. Only one step is taken before the algorithm fixes the node and moves on to another. Between iterations the surrounding nodes (and therefore the optimal position) are modified.

4.2 BridgeSharp Mesh

The fully structured *BridgeSharp* mesh has been generated with GiD. It contains a sharp concave region in which elements become inverted after Laplacian smoothing. The mesh has 32000 hexahedral elements and 28899 interior nodes. The 3D geometry is shown in Figure 4.1. The 2D mesh from the front face is shown in Figure 4.2.

We examine three initial conditions for the mesh. We only compared configurations likely to be encountered in practice, no artificial distortion have been applied.

Initial The mesh as it was received from GiD.

Laplacian Smoothed Smoothed to convergence with Laplacian smoothing. Lapacian preconditioning can be applied to the mesh before a more expensive smoothing method.

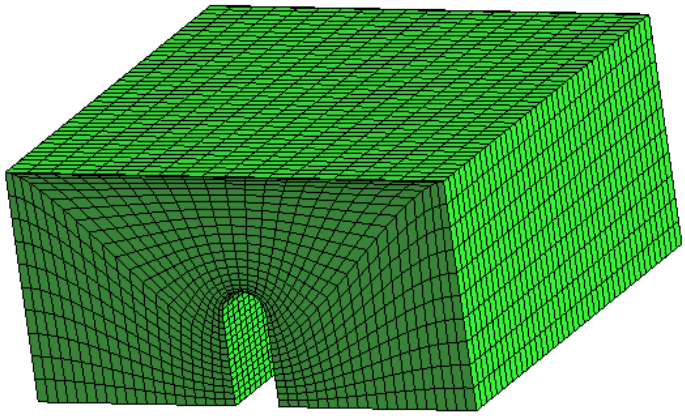

Figure 4.1: *BridgeSharp* geometry

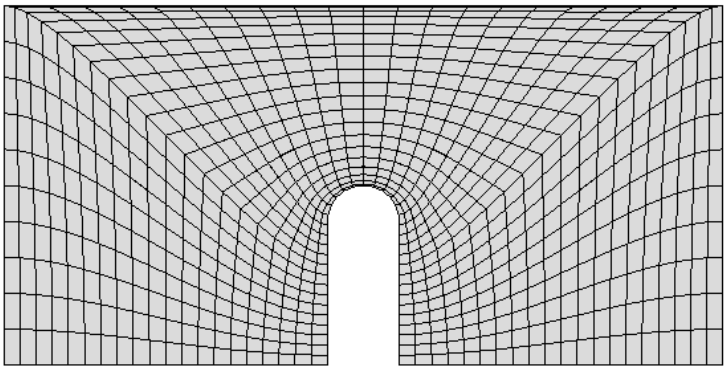

Figure 4.2: *BridgeSharp* mesh

GETMe Smoothed Smoothed to convergence with the GETMe Simultaneous method. The GETMe method is computationally more expensive per iteration than Laplacian smoothing, but can result in a higher quality final mesh. As it is cheaper to perform than the optimzation methods it is a valid preconditioner.

Mesh: bridgeSharpGiD, $\delta = N/A$

		Local Smoothing		Global Smoothing		
	Init	**SD**	**NR**	**SD**	**NR**	**MNR**
qMin	0.1679	0.3474	0.3474	0.3205	0.3475	0.3475
qMean	0.8098	0.8831	0.8831	0.8331	0.8832	0.8832
qNorm	0.6421	0.8659	0.8660	0.8029	0.8660	0.8660
time		**800.365**	2486.47	427.491	1518.79	2560.36
iters		41	35	50	9	15
timeIter		1.952e+01	7.104e+01	8.550e+00	1.688e+02	1.707e+02
tIterNode		6.755e-04	2.458e-03	2.959e-04	5.839e-03	5.906e-03

Table 4.1: Smoothing Mesh bridgeSharpGiD with 28899 free nodes. Initially 0 out of 32000 hexahedra had at least one inverted tetrahedron

Mesh: bridgeSharpGETMe, $\delta = N/A$

		Local Smoothing		Global Smoothing		
	Init	**SD**	**NR**	**SD**	**NR**	**MNR**
qMin	0.3576	0.3472	0.3474	0.3471	0.3475	0.3475
qMean	0.8586	0.8831	0.8831	0.8800	0.8832	0.8832
qNorm	0.8397	0.8660	0.8660	0.8632	0.8660	0.8660
time	202.32	650.278	1972.72	3843.5	**604**	1860.26
iters	439	39	31	501	4	11
timeIter	4.609e-01	1.667e+01	6.364e+01	7.672e+00	1.510e+02	1.691e+02
tIterNode	1.595e-05	5.770e-04	2.202e-03	2.655e-04	5.225e-03	5.852e-03

Table 4.2: Smoothing Mesh bridgeSharpGETMe with 28899 free nodes. Initially 0 out of 32000 hexahedra had at least one inverted tetrahedron

Mesh: bridgeSharpLaplace, $\delta = 0.01$

		Local Smoothing		Global Smoothing		
	Init	**SD**	**NR**	**SD**	**NR**	**MNR**
qMin	0.0000	0.3474	0.3476	0.0544	0.0641	0.0474
qMean	0.8594	0.8831	0.8831	0.8588	0.8665	0.8587
qNorm	0.0000	0.8660	0.8660	0.6293	0.8275	0.7054
time	29.418	**628.614**	2193.99	114.745	41688.5	9106.9
iters	408	35	30	11	201	65
timeIter	7.210e-02	1.796e+01	7.313e+01	1.043e+01	2.074e+02	1.401e+02
tIterNode	2.495e-06	6.215e-04	2.531e-03	3.610e-04	7.177e-03	4.848e-03

Table 4.3: Smoothing Mesh bridgeSharpLaplace with 28899 free nodes. Initially 396 out of 32000 hexahedra had at least one inverted tetrahedron

Mesh: bridgeSharpLaplace, $\delta = 0.2$

		Local Smoothing		Global Smoothing		
	Init	**SD**	**NR**	**SD**	**NR**	**MNR**
qMin	0.0000	0.0000	0.0000	0.0970	0.3475	0.3475
qMean	0.8594	0.8538	0.8531	0.8719	0.8832	0.8832
qNorm	0.0000	0.0000	0.0000	0.7799	0.8660	0.8660
time	29.418	129.752	484.966	1103.23	**1683.11**	2344.48
iters	408	14	14	157	10	14
timeIter	7.210e-02	9.268e+00	3.464e+01	7.027e+00	1.683e+02	1.675e+02
tIterNode	2.495e-06	3.207e-04	1.199e-03	2.432e-04	5.824e-03	5.795e-03

Table 4.4: Smoothing Mesh bridgeSharpLaplace with 28899 free nodes. Initially 396 out of 32000 hexahedra had at least one inverted tetrahedron

4.2.1 Results

Numerical results for *BridgeSharp* are presented in Tables 4.1 to 4.4. We see that Global Steepest descent is unable to smooth the meshes in any of the cases.

A noteworthy effect is observed in Tables 4.3 and 4.4. Selection of a small δ value prevents the *Hessian Based Global* methods from reaching convergence. This is because as $\delta \to 0$ the entries in the Hessian matricies corresponding to inverted elements get very large. These cause no problem for the direct solver used on the 3×3 Hessian, but having a large condition number in the global Hessian impeedes convergence and causes the iterative solver to abort with an inaccurate solution.

Large δ values change the objective function and therefore the location of the smoothed nodes. If δ is too large there is a chance the modified optimal configuration lies outside the feasible region and δ is never set to zero. We see this happen with the local smoother using $\delta = 0.2$ in Table 4.4.

Untangling is only needed outside the feasible region, and as such δ is immediately set to zero and has no effect on the results in Tables 4.1 and 4.2.

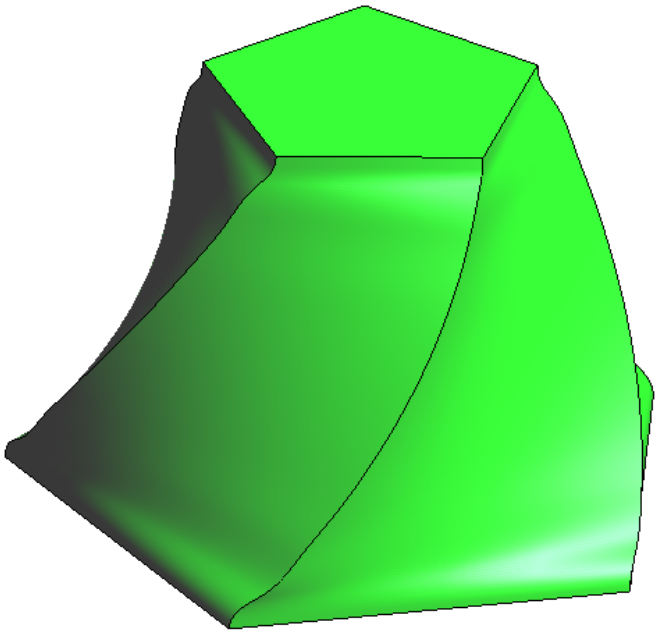

Figure 4.3: *FiveSide* geometry

Preconditioning the optimisation smoothers with a heuristc smoother lead to faster convergence for the quickest method, *Steepest Descent Local Optimisation*. This was true for both the GETMe and Laplacian smoothers even though the Laplacian method inverted several elements which were previously well oriented.

4.3 FiveSide

The *FiveSide* geometry is a regular pentagon which has been extruded, rotated and tapered. It is shown in Figure 4.3 and has been meshed using a range of discretisations to examine the scaling performance of the smoothers.

The *semi-structured* convex meshes are all used as generated by GiD. Finer discretisations had some inverted elements.

Mesh: FiveSide4

		Local Smoothing		Global Smoothing		
	Init	**SD**	**NR**	**SD**	**NR**	**MNR**
qMin	0.5038	0.5702	0.5686	0.5648	0.5709	0.5709
qMean	0.8343	0.8445	0.8447	0.8447	0.8445	0.8445
qNorm	0.8177	0.8315	0.8315	0.8315	0.8315	0.8315
time		**95.7855**	173.808	315.493	126.9	213.617
iters		32	16	281	4	6
timeIter		2.993e+00	1.086e+01	1.123e+00	3.173e+01	3.560e+01
tIterNode		4.892e-04	1.775e-03	1.835e-04	5.185e-03	5.818e-03

Table 4.5: Smoothing Mesh FiveSide4 with 6119 free nodes. Initially 0 out of 7350 hexahedra had at least one inverted tetrahedron

Mesh: FiveSide5

		Local Smoothing		Global Smoothing		
	Init	**SD**	**NR**	**SD**	**NR**	**MNR**
qMin	0.4591	0.6035	0.6036	0.6033	0.6036	0.6036
qMean	0.8697	0.8850	0.8851	0.8848	0.8850	0.8850
qNorm	0.8598	0.8790	0.8790	0.8783	0.8790	0.8790
time		**185.494**	331.551	1797.15	526.962	815.034
iters		29	14	501	7	10
timeIter		6.396e+00	2.368e+01	3.587e+00	7.528e+01	8.150e+01
tIterNode		4.385e-04	1.624e-03	2.459e-04	5.161e-03	5.587e-03

Table 4.6: Smoothing Mesh FiveSide5 with 14587 free nodes. Initially 0 out of 16650 hexahedra had at least one inverted tetrahedron

4.3.1 Results

In all of the cases examined we see that the steepest descent method with local smoothing is the fastest optimisation based smoother to converge. We also note that of the global methods, Newton Raphson is the superior one. As with the *BridgeSharp* mesh, global search direction is important.

While the *Hessian Based Global* algorithms do contain $\mathcal{O}(n^2)$ operations, we can see from Figure 4.4 that when the global methods are sucessful the scaling is not noticably worse than the local optimisation or heuristic methods. Note, however, that on

Mesh: FiveSide6

		Local Smoothing		Global Smoothing		
	Init	**SD**	**NR**	**SD**	**NR**	**MNR**
qMin	0.2681	0.3445	0.3486	0.3059	0.3490	0.3482
qMean	0.6440	0.6695	0.6710	0.6616	0.6709	0.6710
qNorm	0.5986	0.6395	0.6402	0.6300	0.6402	0.6402
time		**2314.44**	22532.2	1101.37	5828.79	3904.82
iters		62	183	96	24	17
timeIter		3.733e+01	1.231e+02	1.147e+01	2.429e+02	2.297e+02
tIterNode		9.394e-04	3.099e-03	2.887e-04	6.112e-03	5.780e-03

Table 4.7: Smoothing Mesh FiveSide6 with 39737 free nodes. Initially 0 out of 44400 hexahedra had at least one inverted tetrahedron

Mesh: FiveSide60k

		Local Smoothing		Global Smoothing		
	Init	**SD**	**NR**	**SD**	**NR**	**MNR**
qMin	0.4582	0.5964	0.6001	0.6504	0.0000	0.5959
qMean	0.8712	0.8953	0.8953	0.8756	0.8766	0.8955
qNorm	0.8654	0.8895	0.8896	0.8705	0.0000	0.8896
time		**2007.59**	7570.52	197.92	83960.9	7401.57
iters		62	64	12	201	23
timeIter		3.238e+01	1.183e+02	1.649e+01	4.177e+02	3.218e+02
tIterNode		5.624e-04	2.055e-03	2.865e-04	7.255e-03	5.589e-03

Table 4.8: Smoothing Mesh FiveSide60k with 57575 free nodes. Initially 0 out of 62350 hexahedra had at least one inverted tetrahedron. While the global steepest descent method returned a higher minimum quality than any other, the solution was worse in terms of the objective function. We can see that the qNorm value was below that of the sucessful search direction methods.

several meshes the global smoother was not able to converge. This was due to a poor search direction from either the Steepest Descent method or failure of the iterative solver to converge within 1000 iterations.

Modifying the Hessian to ensure positive definiteness is of negligible advantage in either the local or global optimization. We saw no local case in which Hessian modification reduced the number of iterations required to converge.

Mesh: FiveSide90k

		Local Smoothing		Global Smoothing		
	Init	**SD**	**NR**	**SD**	**NR**	**MNR**
qMin	0.1095	0.4057	0.4057	0.1095	0.0000	0.0000
qMean	0.7632	0.8245	0.8247	0.7632	0.8026	0.8027
qNorm	0.6903	0.8089	0.8089	0.6903	0.0000	0.0000
time		**2726.78**	10048.9	190.392	2142.19	6644.5
iters		60	53	4	5	14
timeIter		4.545e+01	1.896e+02	4.760e+01	4.284e+02	4.746e+02
tIterNode		5.760e-04	2.403e-03	6.033e-04	5.430e-03	6.016e-03

Table 4.9: Smoothing Mesh FiveSide90k with 78897 free nodes. Initially 24 out of 85400 hexahedra had at least one inverted tetrahedron

Mesh: FiveSide110k

		Local Smoothing		Global Smoothing		
	Init	**SD**	**NR1**	**SD**	**NR**	**MNR**
qMin	0.2004	0.6146	0.6210	0.2004	0.6119	0.6138
qMean	0.8238	0.8895	0.8896	0.8238	0.8898	0.8898
qNorm	0.7829	0.8857	0.8857	0.7829	0.8859	0.8859
time		**4783.04**	13757.8	259.024	5656.01	14057.6
iters		79	67	4	9	23
timeIter		6.054e+01	2.053e+02	6.476e+01	6.284e+02	6.112e+02
tIterNode		5.850e-04	1.984e-03	6.257e-04	6.073e-03	5.906e-03

Table 4.10: Smoothing Mesh FiveSide110k with 103488 free nodes. Initially 2 out of 110450 hexahedra had at least one inverted tetrahedron

	Mesh: FiveSide4			**Mesh: FiveSide5**		
	Init	**Lap**	**GETMe**	**Init**	**Lap**	**GETMe**
qMin	0.5038	0.5849	0.6684	0.4591	0.4749	0.5783
qMean	0.8343	0.8408	0.8310	0.8697	0.8777	0.8819
qNorm	0.8177	0.8299	0.8240	0.8598	0.8682	0.8759
time		2.44896	26.201		8.40326	14.2094
iters		144	319		230	73
timeIter		1.701e-02	8.213e-02		3.654e-02	1.946e-01
tIterNode		2.779e-06	1.342e-05		2.505e-06	1.334e-05

Table 4.11: Heuristic smoothing of two meshes:
FiveSide4, with 6119 free nodes (from a total of 8711).
FiveSide5, with 14587 free nodes (from a total of 18879).

	Mesh: FiveSide6			Mesh: FiveSide60k		
	Init	**Lap**	**GETMe**	**Init**	**Lap**	**GETMe**
qMin	0.2681	0.3557	0.3789	0.4582	0.4131	0.6608
qMean	0.6440	0.6662	0.6562	0.8712	0.8888	0.8851
qNorm	0.5986	0.6387	0.6343	0.8654	0.8818	0.8815
time		33.368	208.144		47.9087	55.9691
iters		342	376		345	69
timeIter		9.757e-02	5.536e-01		1.389e-01	8.111e-01
tIterNode		2.455e-06	1.393e-05		2.412e-06	1.409e-05

Table 4.12: Heuristic smoothing of two meshes:
FiveSide6GiD, with 39737 free nodes (from a total of 49329).
FiveSide60k, with 57575 free nodes (from a total of 67371).

	Mesh: FiveSide90k			Mesh: FiveSide110k		
	Init	**Lap**	**GETMe**	**Init**	**Lap**	**GETMe**
qMin	0.1095	0.1397	0.2815	0.2004	0.4214	0.6334
qMean	0.7632	0.8118	0.8150	0.8238	0.8816	0.8780
qNorm	0.6903	0.7771	0.7911	0.7829	0.8736	0.8736
time		71.4581	183.629		127.026	265.754
iters		371	148		471	160
timeIter		1.926e-01	1.241e+00		2.697e-01	1.661e+00
tIterNode		2.441e-06	1.573e-05		2.606e-06	1.605e-05

Table 4.13: Heuristic smoothing of two meshes:
FiveSide90k, with 78897 free nodes (from a total of 92209).
FiveSide110k, with 103488 free nodes (from a total of 117708).

Well conditioned Global Hessian matricies can be effectively solved by the *minres* routine without modification. If the matrix is poorly conditioned, local smoothing methods are more appropriate.

4.4 Plane

Here we present the final quality distribution of a fully unstructured mesh of the domain around a plane. It has been generated with the receding front method by Eloi Riuz-Gironés. The geometry can be seen in Figure 4.5. Since the optimisation methods are based on the same objective function they generate the same final quality.

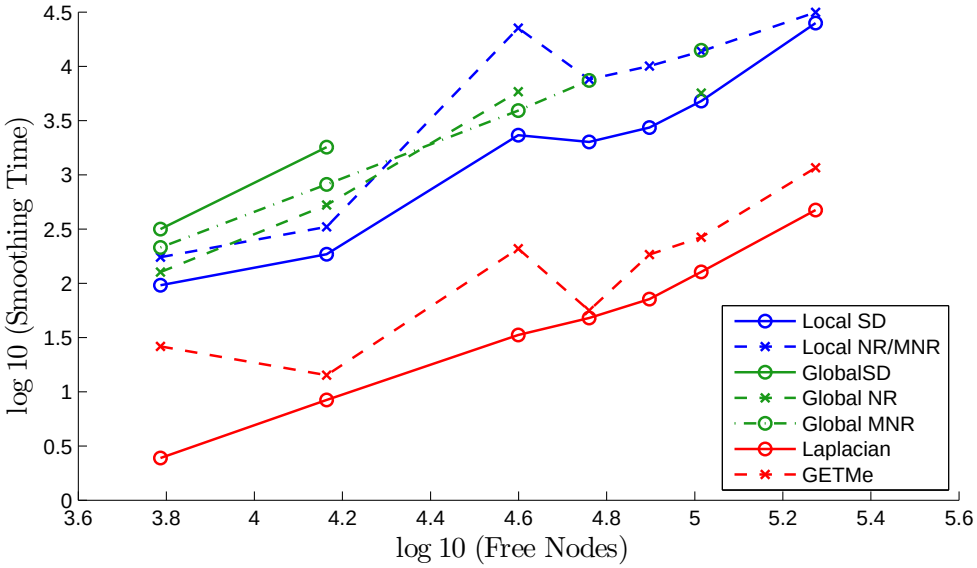

Figure 4.4: Scaling of smoothing algorithms with number of free nodes in *fiveSide* geometry

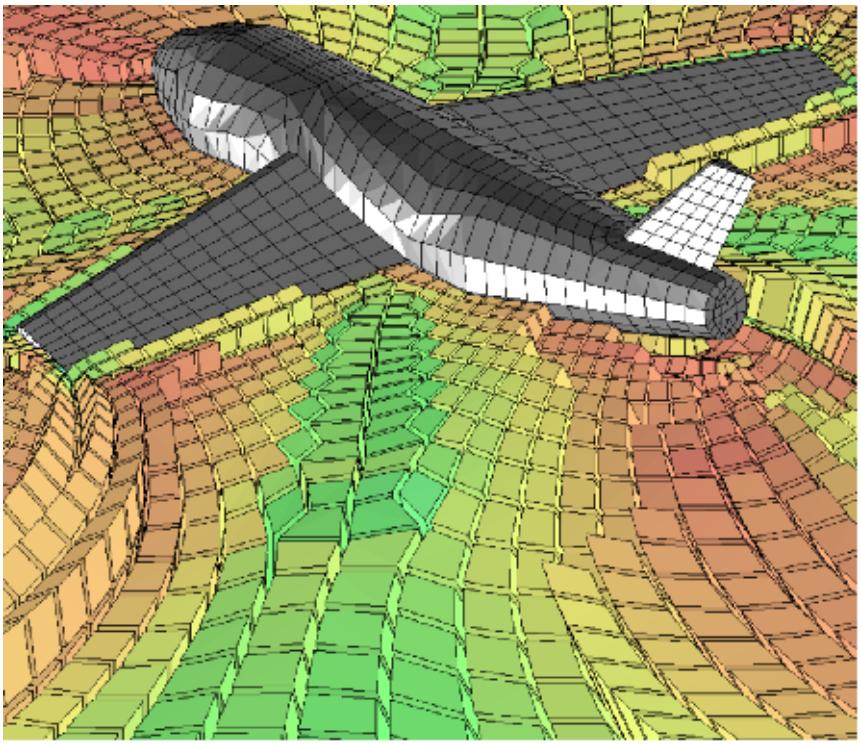

Figure 4.5: Domain around an aeroplane, courtesy Eloi Ruiz-Gironés

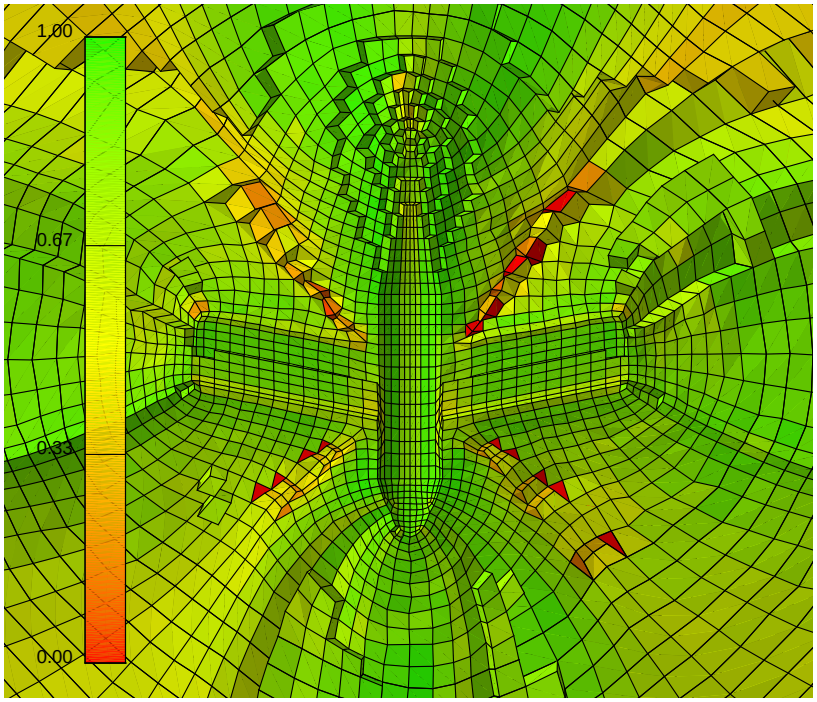

Figure 4.6: Original Plane Mesh, courtesy Eloi Ruiz-Gironés

As such, only one result is presented in Table 4.14 and Figure 4.9, it applies to all search directions for local and global optimisation in which convergence was successfully achieved.

A cross section of the original mesh is shown in Figure 4.6, the same clipping plane has been used on the smoothed meshes in Figures 4.7 and 4.8. Final qualities are given in Table 4.14, and their distribution is shown in Figure 4.9.

4.4.1 Results

We can see from Table 4.14 that optimization methods are able to converge on a higher quality final mesh than any other. For this mesh Laplacian smoothing lowers the mean quality and inverts a number of elements which were previously in the feasible region. GETMe Simultaneous smoothing increases the minimum and mean

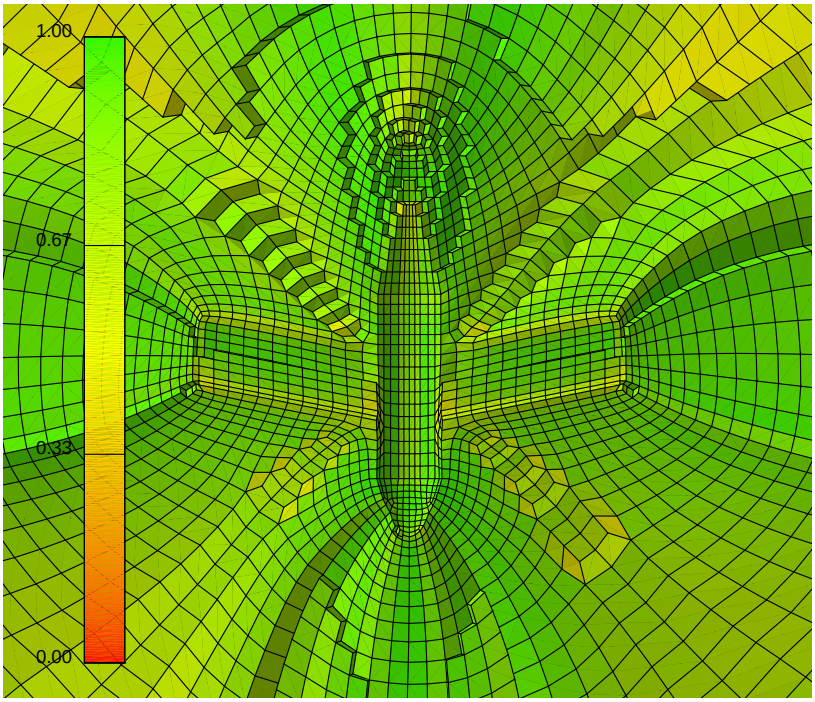

Figure 4.7: Plane mesh smoothed using the GETMe Simultaneous smoother

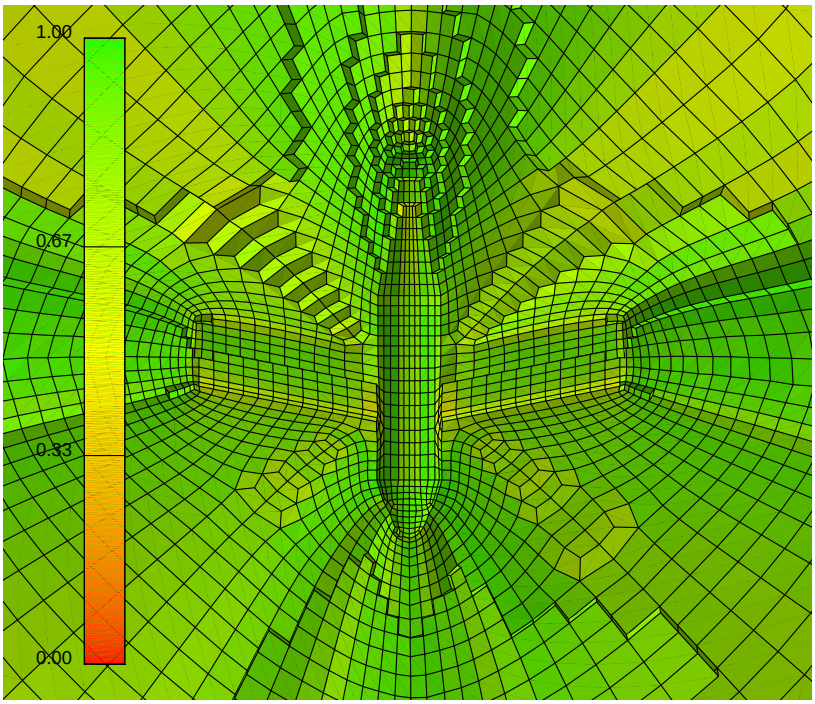

Figure 4.8: Plane mesh smoothed using optimization smoothers

Smoothing Method	Min	Mean
Initial	0.0945	0.8164
Laplacian	0.0000	0.6902
GETMe Simultaneous	0.5037	0.8792
Optimization	0.6057	0.9129

Table 4.14: Converged Qualities of smoothers on Plane mesh

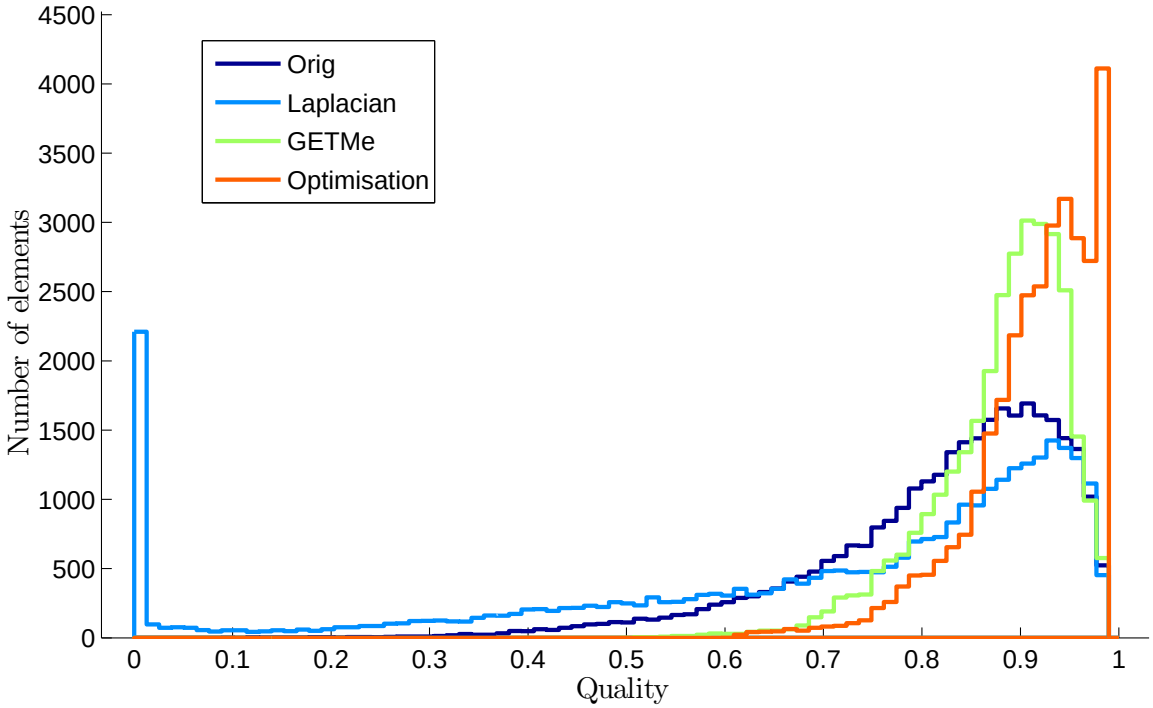

Figure 4.9: Quality distributions for *Plane* mesh

qualities, but not as much as optimisation smoothing.

Table 4.15 lists the smoothing times of the various methods. Optimisation smoothers have been run on both the initial mesh and the converged *GETMe* mesh.

On this mesh the global optimisation method is not suitable. The complex geometry and wide range of element sizes leads to a poorly selected search direction in the case of *Global Steepest Descent*, and an ill-conditioned Hessian for the *Global Newton Raphson* method. Further discussion can be found in Section 4.1.3.

Method	Time (s) / Iterations / Time per Iteration (s)	Qualities		
		Min	Mean ± Std Dev	Max
Initial Mesh		0.0945	0.8164 ± 0.1276	0.9968
	Heuristic Smoothing of **Initial** Plane			
Lap	43.507278 **(609)** 0.071441	0.0000	0.6902 ± 0.2829	0.9938
GETMeSim	86.945900 **(204)** 0.426205	0.5037	0.8792 ± 0.0689	0.9959
	Local Smoothing of **Initial** Plane			
SD	1028.099 **(71)** 14.480	0.6057	0.9129 ± 0.0636	0.9973
NR	2719.213 **(45)** 60.427	0.6022	0.9131 ± 0.0639	0.9974
	Global Smoothing of **Initial** Plane			
SD	515.061 **(4)** 128.765	0.0945	0.8164 ± 0.1276	0.9968
NR	41978.557 **(201)** 208.849	0.0843	0.8654 ± 0.0969	0.9984
MNR	13477.851 **(88)** 153.157	0.6031	0.9127 ± 0.0645	0.9968
	Local Smoothing of **GETMe Smoothed** Plane			
SD	1073.170 **(69)** 15.553	0.6054	0.9129 ± 0.0636	0.9973
NR	2349.306 **(39)** 60.239	0.6031	0.9130 ± 0.0638	0.9974
	Global Smoothing of **GETMe Smoothed** Plane			
SD	517.673 **(4)** 129.418	0.5037	0.8792 ± 0.0689	0.9959
NR	42742.141 **(201)** 212.648	0.0040	0.9038 ± 0.0873	0.9979
MNR	19177.808 **(141)** 136.013	0.1100	0.8785 ± 0.0721	0.9959

Table 4.15: Using mesh Plane, with 30389 free nodes (from a total of 33411). In the Initial configuration 163 of the 31760 elements have at least one inverted tetrahedron. In the GETMe Smoothed configuration, 4 of the 31760 elements have at least one inverted tetrahedron. In all optimisation methods $\delta = 0.01$.

The heuristic smoothers are able to converge much more rapidly than optimisation based smoothers. The fastest optimisation method, *Local Steepest Descent*, was more than an order of magnitude slower than the slowest heuristic method, *GETMe*.

When smoothing the *BridgeSharp* mesh we noted that time spent preconditioning the mesh heuristically was compensated for by a faster optimisation stage. This was not

the case with the *Plane* mesh, *Local Steepest Descent* smoothing was slightly faster from the mesh's initial state.

Chapter 5

Conclusions

5.1 Conclusions

We have implemented the *Laplacian* and *GETMe* heuristic smoothers, as well as a local optimisation smoother and untangler of hexahedral meshes. We have compared these methods with a new, global smoother and untangler developed in this work.

We have shown several examples in which local and global approaches to untangling and smoothing converge to the same optimum. Global smoothers have been shown to be less robust and are competitive only when using second order search direction methods with a well conditioned Hessian. Practical issues arise on larger meshes, meshes with a large range of element sizes and meshes which contain heavily inverted elements. These issues make local optimisation the most appealing method of maximising the quality of a hexahedral mesh.

We have shown that local smoothing of single nodes is best performed with the steepest descent search direction. Since the information used to determine the direction is temporary, a more accurate solution is unnecessary.

Furthermore, we have shown that heuristic smoothing methods, while not resulting

in optimal node locations, can serve as an effective initial guess to the optimisation methods. A general procedure for improving the quality of a hexahedral mesh using the results of this work is given in Algorithm 5.1.

Algorithm 5.1 General Smoothing Method

```
Given initial mesh X_init
X_GETMe ← GETMe smoothed X_init
if Quality of X_GETMe is sufficient then
  Return X_GETMe
else
  if Mesh contains similarly sized, untangled elements then
    Return Global NR smoothed X_GETMe
  else
    Return Local SD smoothed X_GETMe
  end if
end if
```

5.2 Future Work

5.2.1 Boundary Node Movement

The global mesh quality could be improved if nodes were able to move on the boundary. For example, nodes on an edge could be confined to just this edge. Nodes a surface could be free to move on that surface, and nodes on an edge could move along the edge.

Since optimisation methods have been used, an objective function could be defined which takes into account the quality of surface elements as well as the three dimensional volume elements. An example of surface mesh smoothing can be found in (Gargallo, A., 2011).

5.2.2 Iterative Solver Improvements

The size of the linear system generated by the global approach requires an iterative method to solve it. Only the most superficial research has been put into optimising this part of the solver and it is expected that efficiency gains could be made by further exploring this major area of research.

5.2.3 Coding in a Lower Level Language

Results presented in this work are based on code written in MATLAB. Reimplementation using the ez4u framework in C++ has been done for the heuristic methods as well as the local optimiser, but not the global optimiser. A comparison of all methods using C++ could result in different conclusions as the MATLAB overhead may vary between methods.

5.2.4 Use of Multiple δ Values

We have seen that large values of δ make the system easier to solve, but in some cases can lead to an optimal solution outside the feasible region. The possibility exists to reduce δ as the system approaches the feasible region.

5.2.5 Smoothing Large Patches Sequentially

In this work we have examined smoothing all nodes simultaneously as well as one at a time. It is possible there is a more efficient middle ground in which the sequential smoothing of groups of connected nodes of a fixed size could be performed.

This would have the some of the advantages of global smoothing such as derivative component reuse and would enable more nodal movement at each step. The number of elements could be selected such that the direct solution of the system was feasible, or that indirect methods would not encounter prohibitive scaling.

5.2.6 Intelligent Smoothing Order

Since the mesh boundary is fixed, information propagates from it to the mesh interior. By smoothing levels of nodes starting at the boundary it could be possible to take advantage of this inherent directionality.

Appendices

Appendix A

Algorithms

A.1 Convergence

Algorithm A.1 Convergence Criterion

$X \leftarrow$ Original Coordinates
$X' \leftarrow$ Updated Coordinates
$tol \leftarrow$ Tolerace
$dXG \leftarrow 0$
for $x_i \in X$, $x'_i \in X'$ **do**
  $dX \leftarrow$abs$(x'_i - x_i)$
  if $dX > dXG$ **then**
    $dXG \leftarrow dX$
  end if
  $x_i \leftarrow x'_i$
end for
if $dXG < tol$ **then**
  Converged $\leftarrow$ True
else
  Converged $\leftarrow$ False
end if

A.2 Search Direction

Algorithms A.2, A.3 and A.4 are used to select the direction for the line search. They can be found in (Nocedal and Wright, 1999).

With regard to the Modified Newton Raphson Algorithm A.4, in the local case where we have a 3×3 matrix the eigenvalues λ_{min} and λ_{max} are computed directly. When smoothing the global case they must be approximated, which is an expensive process if the ratio $\frac{\lambda_{max}}{\lambda_{min}}$ is large. If the approximation does not converge within 50 iterations the minimum eigenvale is taken to be zero and the algorithm increases the eigenvalues.

Algorithm A.2 Steepest Descent

given initial x_0
$k \leftarrow 0$
repeat
  obtain $\nabla f(x_k)$
  set $p_k = -\nabla f(x_k)$
  set $x_{k+1} \leftarrow x_k + \alpha_k p_k$ {using Line Search}
  $k \leftarrow k + 1$
until convergence

Algorithm A.3 Newton Raphson

given initial x_0
$k \leftarrow 0$
repeat
  $\mathbf{H} \leftarrow \nabla^2 f(x_k)$
  solve $\mathbf{H}p_k = -\nabla f(x_k)$ for search direction p_k
  set $x_{k+1} \leftarrow x_k + \alpha_k p_k$ {using Line Search}
  $k \leftarrow k + 1$
until convergence

A.3 Step Length

Algorithm A.4 Modified Newton Raphson.

given initial x_0
$k \leftarrow 0$
repeat
  $\mathbf{H} \leftarrow \nabla^2 f(x_k)$
  obtain eigenvectors $\lambda_{min}, \lambda_{max}$ from $\mathbf{H}$
  if $\lambda_{min} < 0.001\lambda_{max}$ **then**
    $\mathbf{H} \leftarrow \mathbf{H} + (0.001\lambda_{max} - \lambda_{min})\,\mathbf{I}$
  end if
  solve $\mathbf{H}p_k = -\nabla f(x_k)$ for search direction p_k
  set $x_{k+1} \leftarrow x_k + \alpha_k p_k$ {using Line Search}
  $k \leftarrow k + 1$
until convergence

Algorithm A.5 Back Line Search with Armijo condition

$\lambda \leftarrow 0.7$
$c_1 \leftarrow 10^{-4}$
$\alpha \leftarrow 1$
while $f(x_k + \alpha p_k) \geq f(x_k) + c_1 \alpha \nabla f_k^T p_k$ **do**
  $\alpha \leftarrow \lambda\alpha$
end while

Appendix B

Second Derivative of the Shape Metric η

The shape metric given by (Escobar et al., 2003) is

$$\eta = \frac{|\mathbf{S}|^2}{nh(\sigma)^{2/n}} \tag{B.1}$$

$$h(\sigma) = \frac{1}{2}\left(\sigma + \sqrt{\sigma^2 + 4\delta^2}\right) \tag{B.2}$$

where n is the number of dimensions. Throughout this work $n = 3$ but the coefficient is left in for generality.

The development of the first derivative given in (Rivas, C. A., 2010) is valid in the global case, and is given by

$$\frac{\partial \eta}{\partial \alpha} = 2\eta \left[\left(\frac{\partial \mathbf{S}}{\partial \alpha}, \mathbf{S} \right) \frac{1}{|\mathbf{S}|^2} - \frac{1}{n} \frac{\partial \sigma}{\partial \alpha} \frac{1}{\sqrt{\sigma^2 + 4\delta^2}} \right] \tag{B.3}$$

To obtain the second derivative we will consider the following relations, given in (Petersen and Pedersen, 2008). α and β are arbitrary degrees of freedom, any single

coordinate direction of a node of a tetrahedron. $\mathbf{S}$ and $\mathbf{T}$ are matricies.

$$\begin{aligned}
\sigma &= \det \mathbf{S} && \text{(B.4)}\\
\frac{\partial \sigma}{\partial \alpha} &= \sigma \ \mathrm{Tr}\left[\mathbf{S}^{-1}\frac{\partial \mathbf{S}}{\partial \alpha}\right] && \text{(B.5)}\\
\frac{\partial \ \mathrm{Tr}\ \mathbf{S}}{\partial \alpha} &= \mathrm{Tr}\left[\frac{\partial \mathbf{S}}{\partial \alpha}\right] && \text{(B.6)}\\
\frac{\partial \mathbf{S}\mathbf{T}}{\partial \alpha} &= \frac{\partial \mathbf{S}}{\partial \alpha}\mathbf{T} + \mathbf{S}\frac{\partial \mathbf{T}}{\partial \alpha} && \text{(B.7)}\\
\frac{\partial \mathbf{S}^{-1}}{\partial \alpha} &= -\mathbf{S}^{-1}\frac{\partial \mathbf{S}}{\partial \alpha}\mathbf{S}^{-1} && \text{(B.8)}
\end{aligned}$$

and in our case

$$\frac{\partial^2 \mathbf{S}}{\partial \alpha \partial \beta} = 0 \qquad \text{(B.9)}$$

B.1 Second Derivative of σ

Using (B.6) and (B.7) on the first σ derivative (B.5) we obtain the following

$$\begin{aligned}
\frac{\partial}{\partial \beta}\left(\frac{\partial \sigma}{\partial \alpha}\right) &= \sigma \ \mathrm{Tr}\left[\mathbf{S}^{-1}\frac{\partial \mathbf{S}}{\partial \beta}\right] \mathrm{Tr}\left[\mathbf{S}^{-1}\frac{\partial \mathbf{S}}{\partial \alpha}\right] + \\
&\quad \sigma \ \mathrm{Tr}\left[\frac{\partial \mathbf{S}^{-1}}{\partial \beta}\frac{\partial \mathbf{S}}{\partial \alpha}\right] + \sigma \ \mathrm{Tr}\left[\mathbf{S}^{-1}\frac{\partial^2 \mathbf{S}}{\partial \alpha \partial \beta}\right] \qquad \text{(B.10)}
\end{aligned}$$

Now using (B.8) and (B.9) we obtain the expression

$$\frac{\partial^2 \sigma}{\partial \alpha \partial \beta} = \sigma \ \mathrm{Tr}\left[\mathbf{S}^{-1}\frac{\partial \mathbf{S}}{\partial \beta}\right] \mathrm{Tr}\left[\mathbf{S}^{-1}\frac{\partial \mathbf{S}}{\partial \alpha}\right] - \sigma \ \mathrm{Tr}\left[\mathbf{S}^{-1}\frac{\partial \mathbf{S}}{\partial \beta}\mathbf{S}^{-1}\frac{\partial \mathbf{S}}{\partial \alpha}\right] \qquad \text{(B.11)}$$

While the expression given by (B.11) is nonzero in the general global case and must be included, it goes to zero in following special cases, the first of which applies when performing element by element local optimization.

1. Both of the degrees of freedom α and β are on the same node.

2. Both of the degrees of freedom α and β are in the same direction (x, y or z).

B.2 Second Derivative of Tetrahedral η

We follow a similar procedure to that in (Rivas, C. A., 2010), however no simplifications are made which depend on both of the components of second derivative being taken at the same node. We begin by taking the derivative of (B.3) with respect to a new degree of freedom β,

$$\begin{aligned}\frac{\partial^2\eta}{\partial\alpha\partial\beta} &= 2\frac{\partial\eta}{\partial\beta}\left[\left(\frac{\partial\mathbf{S}}{\partial\alpha},\mathbf{S}\right)\frac{1}{|\mathbf{S}|^2}-\frac{1}{n}\frac{\partial\sigma}{\partial\alpha}\frac{1}{\sqrt{\sigma^2+4\delta^2}}\right]+ \\ &\quad 2\eta\frac{\partial}{\partial\beta}\left[\left(\frac{\partial\mathbf{S}}{\partial\alpha},\mathbf{S}\right)\frac{1}{|\mathbf{S}|^2}-\frac{1}{n}\frac{\partial\sigma}{\partial\alpha}\frac{1}{\sqrt{\sigma^2+4\delta^2}}\right] \qquad \text{(B.12)}\end{aligned}$$

We split (B.12) into several terms as follow:

$$2\frac{\partial\eta}{\partial\beta}\left[\left(\frac{\partial\mathbf{S}}{\partial\alpha},\mathbf{S}\right)\frac{1}{|\mathbf{S}|^2}-\frac{1}{n}\frac{\partial\sigma}{\partial\beta}\frac{1}{\sqrt{\sigma^2+4\delta^2}}\right]=A\times B \qquad \text{(B.13)}$$

where A and B are defined as

$$\begin{aligned}A &= 2\eta\left[\left(\frac{\partial\mathbf{S}}{\partial\beta},\mathbf{S}\right)\frac{1}{|\mathbf{S}|^2}-\frac{1}{n}\frac{\partial\sigma}{\partial\beta}\frac{1}{\sqrt{\sigma^2+4\delta^2}}\right] \\ &= \frac{\partial\eta}{\partial\beta} \qquad \text{(B.14)} \\ B &= 2\left[\left(\frac{\partial\mathbf{S}}{\partial\alpha},\mathbf{S}\right)\frac{1}{|\mathbf{S}|^2}-\frac{1}{n}\frac{\partial\sigma}{\partial\alpha}\frac{1}{\sqrt{\sigma^2+4\delta^2}}\right] \\ &= \frac{1}{\eta}\frac{\partial\eta}{\partial\alpha} \qquad \text{(B.15)}\end{aligned}$$

Combining (B.14) and (B.15) with (B.13) gives us

$$2\frac{\partial\eta}{\partial\beta}\left[\left(\frac{\partial\mathbf{S}}{\partial\alpha},\mathbf{S}\right)\frac{1}{|\mathbf{S}|^2}-\frac{1}{n}\frac{\partial\sigma}{\partial\alpha}\frac{1}{\sqrt{\sigma^2+4\delta^2}}\right]=\frac{1}{\eta}\frac{\partial\eta}{\partial\alpha}\frac{\partial\eta}{\partial\beta} \qquad \text{(B.16)}$$

The second part of (B.12) will be considered as two parts in the following manner:

$$
\begin{aligned}
& 2\eta\frac{\partial}{\partial\beta}\left[\left(\frac{\partial\mathbf{S}}{\partial\alpha},\mathbf{S}\right)\frac{1}{|\mathbf{S}|^2}\right] \\
&= 2\eta\left[\left(\frac{\partial^2\mathbf{S}}{\partial\alpha\partial\beta},\mathbf{S}\right)\frac{1}{|\mathbf{S}|^2}+\left(\frac{\partial\mathbf{S}}{\partial\alpha},\frac{\partial\mathbf{S}}{\partial\beta}\right)\frac{1}{|\mathbf{S}|^2}+\left(\frac{\partial\mathbf{S}}{\partial\alpha},\mathbf{S}\right)\frac{\partial}{\partial\beta}\left(|\mathbf{S}|^{-2}\right)\right] \\
&= 2\eta\left[\left(\frac{\partial^2\mathbf{S}}{\partial\alpha\partial\beta},\mathbf{S}\right)\frac{1}{|\mathbf{S}|^2}+\left(\frac{\partial\mathbf{S}}{\partial\alpha},\frac{\partial\mathbf{S}}{\partial\beta}\right)\frac{1}{|\mathbf{S}|^2}-2\left(\frac{\partial\mathbf{S}}{\partial\alpha},\mathbf{S}\right)\frac{1}{|\mathbf{S}|^4}\left(\frac{\partial\mathbf{S}}{\partial\beta},\mathbf{S}\right)\right] \qquad \text{(B.17)}
\end{aligned}
$$

Considering the second part of the second term of (B.12) gives us the following

$$
\begin{aligned}
& 2\eta\frac{\partial}{\partial\beta}\left(-\frac{1}{n}\frac{\partial\sigma}{\partial\alpha}\frac{1}{\sqrt{\sigma^2+4\delta^2}}\right) \\
&= -\frac{2\eta}{n}\left(\frac{\partial^2\sigma}{\partial\alpha\partial\beta}\frac{1}{\sqrt{\sigma^2+4\delta^2}}+\frac{\partial\sigma}{\partial\alpha}\frac{\partial}{\partial\beta}\frac{1}{\sqrt{\sigma^2+4\delta^2}}\right) \\
&= -\frac{2\eta}{n}\left(\frac{\partial^2\sigma}{\partial\alpha\partial\beta}\frac{1}{\sqrt{\sigma^2+4\delta^2}}-\sigma\frac{\partial\sigma}{\partial\alpha}\frac{\partial\sigma}{\partial\beta}\left(\sigma^2+4\delta^2\right)^{\frac{-3}{2}}\right) \qquad \text{(B.18)}
\end{aligned}
$$

By incorporating (B.9) and (B.11) into (B.18) then substituting this along with (B.16) and (B.17) into (B.12), we obtain the global second derivative of a tetradehron component.

$$
\begin{aligned}
\frac{\partial^2\eta}{\partial\alpha\partial\beta} &= \frac{1}{\eta}\frac{\partial\eta}{\partial\alpha}\frac{\partial\eta}{\partial\beta}+2\eta\left[\left(\frac{\partial\mathbf{S}}{\partial\alpha},\frac{\partial\mathbf{S}}{\partial\beta}\right)\frac{1}{|\mathbf{S}|^2}-2\frac{1}{|\mathbf{S}|^4}\left(\frac{\partial\mathbf{S}}{\partial\alpha},\mathbf{S}\right)\left(\frac{\partial\mathbf{S}}{\partial\beta},\mathbf{S}\right)-\right. \\
&\quad \sigma\ \mathrm{Tr}\left[\mathbf{S}^{-1}\frac{\partial\mathbf{S}}{\partial\beta}\right]\mathrm{Tr}\left[\mathbf{S}^{-1}\frac{\partial\mathbf{S}}{\partial\alpha}\right]-\sigma\ \mathrm{Tr}\left[\mathbf{S}^{-1}\frac{\partial\mathbf{S}}{\partial\beta}\mathbf{S}^{-1}\frac{\partial\mathbf{S}}{\partial\alpha}\right]+ \\
&\quad \left.\frac{\sigma}{n}\frac{\partial\sigma}{\partial\alpha}\frac{\partial\sigma}{\partial\beta}\left(\sigma^2+4\delta^2\right)^{\frac{-3}{2}}\right] \qquad \text{(B.19)}
\end{aligned}
$$

B.3 Derivatives of Hexahedral η

The shape metric of a hexahedron is defined as the mean value of the tetrahedra which compose it, trasposed such that the minimum (a perfect cube element) is zero (Escobar et al., 2003).

$$\eta_{hex} = \frac{1}{8}\sum \eta_i - 1 \tag{B.20}$$

$$\frac{\partial \eta_{hex}}{\partial \alpha} = \frac{1}{8}\sum \frac{\partial \eta_i}{\partial \alpha} \tag{B.21}$$

$$\frac{\partial^2 \eta_{hex}}{\partial \alpha \partial \beta} = \frac{1}{8}\sum \frac{\partial^2 \eta_i}{\partial \alpha \partial \beta} \tag{B.22}$$

In Equations (B.20) to (B.22) i subscripts (for example, η_i) denote the sub-elements in the hexahedra. It is important to note that the indicies of tetrahedral and hexahedral α and β are not the same, an assembley needs to be performed based on the entries in Table 2.1.

Appendix C

Local and Global Agreement of Mesh Optimum Point

We aim to assert that a nodal configuration (x1, x2, . . . , xn) is a candidate of the global minimization if and only if is a candidate point for the local minimization.
We aim to assert that a nodal configuration $(\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n})$ is a candidate point for the *global* minimisation if and only if if is a candidate point for the *local* minimisation.

We start with some nomenclature,

i	index for nodes $\mathbf{x_i}$
j	index for elements
η_j	objective function of element j
S_i	indicies j of elements η_j in which node $\mathbf{x_i}$ can be found
n_x	number of free nodes in the mesh
n_e	number of elements in the mesh

The elemental objective function derivative is $\dfrac{\partial \eta_j}{\partial \mathbf{x_i}}$, therefore

$$\frac{\partial \eta_j^p}{\partial \mathbf{x_i}} = \begin{cases} p \cdot \dfrac{\partial \eta_j}{\partial \mathbf{x_i}} \cdot \eta_j^{p-1} & \text{if} \quad j \in S_i \\ 0 & \text{otherwise} \end{cases} \tag{C.1}$$

We now condsider the derivative of the global objective function (3.3),

$$\begin{aligned} \eta &= \left(\sum_{j=1}^{n_e} \eta_j^p \right)^{\frac{1}{p}} \\ \frac{\partial \eta}{\partial \mathbf{x_i}} &= \sum_{j=1}^{n_e} \frac{\partial \eta_j^p}{\partial \mathbf{x_i}} \cdot \frac{1}{p} \cdot \left(\sum_{j=1}^{n_e} \eta_j^p \right)^{\frac{1}{p}-1} \end{aligned} \tag{C.2}$$

By removing zero terms with (C.1), (C.2) becomes

$$\frac{\partial \eta}{\partial \mathbf{x_i}} = \sum_{j \in S_i} \frac{\partial \eta_j^p}{\partial \mathbf{x_i}} \cdot \frac{1}{p} \cdot \left(\sum_{j \in S_i} \eta_j^p \right)^{\frac{1}{p}-1} \tag{C.3}$$

We now obtain the derivative of the local objective function at node $\mathbf{x_i}$ by setting S_{elems} from (3.3) to S_i. The function η_i is the combination of objective functions of elements η_j in S_i.

$$\eta_i = \left(\sum_{j \in S_i} \eta_j^p \right)^{\frac{1}{p}}$$

Making use of (C.1),

$$\frac{\partial \eta_i}{\partial \mathbf{x_i}} = \sum_{j \in S_i} \frac{\partial \eta_j^p}{\partial \mathbf{x_i}} \cdot \frac{1}{p} \cdot \left(\sum_{j \in S_i} \eta_j^p \right)^{\frac{1}{p}-1} \tag{C.4}$$

We see from (C.3) and (C.4) that the derivatives of the global and local objective function are the same at any node $\mathbf{x_i}$, allowing us to trivially prove the assertation.

If $(\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n})$ is a singular point of global η then

$$\begin{aligned} \frac{\partial \eta}{\partial \mathbf{x_i}} &= 0 \qquad \text{for all } i = 1, 2, \ldots, n_x \\ 0 &= \frac{\partial \eta \left(\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\right)}{\partial \mathbf{x_i}} \\ \frac{\partial \eta_i \left(\mathbf{x_i}\right)}{\partial \mathbf{x_i}} &= 0 \qquad \text{for all } i = 1, 2, \ldots, n_x \end{aligned}$$

therefore $(\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n})$ is also a singular point of local η_i around node $\mathbf{x_i}$.

Bibliography

Dompierre, Labbé, Guibault and Camarero (1998). Proposal of Benchmarks for 3D Unstructured Tetrahedral Mesh Optimization. *Proceedings of the 7th International Meshing RoundTable'98*, 459–478.

Escobar, J. M., E. Rodríguez, R. Montenegro, G. Montero, and J. M. González-Yuste (2003). Simultaneous untangling and smoothing of tetrahedral meshes. *Computer Methods in Applied Mechanics and Engineering 192*(25), 2775–2787.

Freitag, Knupp, Munson and Shontz (2006). A comparison of two optimization methods for mesh quality improvement *Engineering with Computers 22*(2), 61–74.

Frey, P. J. and George, P. (2007). Mesh Generation: Application to Finite Elements *ISTE* ISBN 1-903-39800-2

Gargallo Peiró, A. (2011). Smoothing and untangling of meshes on parameterized surfaces *Masters Thesis*, Facultat de Matemàtiques i Estadística. Universitat Politècnica de Catalunya.

Knupp, P. (2000). Hexahedral Mesh Untangling & Algebraic Mesh Quality Metrics *Proceedings of the 9th International Meshing Roundtable*, 173–183.

Knupp, P (2001). Hexahedral and Tetrahedral Mesh Untangling *Engineering with Computers 17*(3), 261–268.

Knupp, P (2003). A method for hexahedral mesh shape optimization *International Journal for Numerical Methods in Engineering 58*(2), 319–332.

Ledoux, F. and Weill, J. (2008). An Extension of the Reliable Whisker Weaving Algorithm *Proceedings of the 16th International Meshing Roundtable*, 215–232.

Munson, T. (2007). Mesh shape-quality optimization using the inverse mean-ratio metric *Mathematical Programming 110*(3), 561–590.

Nocedal, J. and Wright, S. (1999). Numerical Optimization. *Springer (New York)* ISBN 0-387-98793-2

Petersen, K. B. and Pedersen, M. S. (2008). The Matrix Cookbook `http://www2.imm.dtu.dk/pubdb/p.php?3274`

Rivas Guerra, C. A. (2010). Simultaneous Untangling and Smoothing of Hexahedral Meshes. *Masters Thesis*, Facultat de Matemàtiques i Estadística. Universitat Politècnica de Catalunya.

Roca, X. (2009, July). Paving the Path Towards Automatic Hexahedral Mesh Generation. *Ph. D. thesis*, Facultat de Matemàtiques i Estadística. Universitat Politècnica de Catalunya.

Saad, Y. (2003). Iterative Methods for Sparse Linear Systems *Society for Industrial and Applied Mathematics (Philadelphia)* ISBN 0-898-71534-2

Sastry, S. and Shontz, S. (2009) A Comparison of Gradient- and Hessian-Based Optimization Methods for Tetrahedral Mesh Quality Improvement *Proceedings of the 18th International Meshing Roundtable* 631–648.

Tournois, Wormser, Alliez and Desbrun (2009). Interleaving Delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. *ACM Trans. Graph. 28*(3), 75:1–75:9.

Vartziotis, D. and Wipper, J. (2011). A dual element based geometric element transformation method for all-hexahedral mesh smoothing *Computer Methods in Applied Mechanics and Engineering 200*(9-12), 1186–1203.

(Petersen and Pedersen, 2008; Rivas, C. A., 2010; Gargallo, A., 2011; Escobar et al., 2003; Dompierre et al., 1998; Tournois et al., 2009; Roca, 2009; Vartziotis and Wipper, 2011; Freitag et al., 2006; Knupp, 2003, 2001; Nocedal and Wright, 1999; Sastry and Shontz, 2009; Munson, 2007; Knupp, 2000; Saad, 2003; Ledoux, 2008; Frey et al., 2007)

DO NOT INCLUDE THIS PAGE!