The Optimisation of Finite Element Meshes

Alan Kelly

Infrastructure & Environment Research Division
School of Engineering

University of Glasgow

Submitted in fulfilment of the requirements for the Degree of Doctor of Philosophy

November 2014

Declaration

Abstract

Among the several numerical methods[1] which are available for solving complex problems in many areas of engineering and science such as structural analysis, fluid flow and bio-mechanics, the Finite Element Method (FEM) is the most prominent. In the context of these methods, high quality meshes can be crucial to obtaining accurate results. Finite Element meshes are composed of elements and the quality of an element can be described as a numerical measure which estimates the effect that the size/shape of an element will have on the accuracy of an analysis. In this thesis, the strong link between mesh geometry and the accuracy and efficiency of a simulation is explored and it is shown that poor quality elements cause both interpolation errors and poor conditioning of the global stiffness matrix.

Numerical optimisation is the process of maximising or minimising an objective function, subject to constraints on the solution. When this is applied to a finite element mesh it is referred to as mesh optimisation, where the quality of the mesh is the objective function and the constraints include, for example, the domain geometry, maximum element size, etc. A mesh optimisation strategy is developed with a particular focus on optimising the quality of the worst elements in a mesh. Using both two and three dimensional examples, the most efficient and effective combination of element quality measure and objective function is found.

Many of the problems under consideration are characterised by very complex geometries. The nodes lying on the surfaces of such meshes are typically treated as unmovable by most mesh optimisation software. Techniques exist for moving such nodes as part of the mesh optimisation process, however, the resulting mesh geometry and area/volume is often not conserved. This means that the optimised mesh is no longer an accurate discretisation of the original domain. Therefore, a method is developed and demonstrated which optimises the positions of surface nodes while respecting the geometry and area/volume of a domain.

At the heart of many of the problems being considered is the Arbitrary Lagrangian Eulerian (ALE) formulation where the need to ensure mesh quality in an evolving mesh is very important. In such a formulation, a method of determining the updated nodal positions is required. Such a method is developed using mesh optimisation techniques as part of the FE solution process and this is demonstrated using a two-dimensional, axisymmetric simulation of a micro-fluid droplet subject to external excitation. While better quality meshes were observed using this method, the time step collapsed resulting in simulations requiring significantly more time to complete. The extension of this method to incorporate adaptive re-meshing is also discussed.

[1]e.g. The Finite Element Method (FEM), the Finite Difference Method (FDM), the Boundary Element Method (BEM), the Discrete Element Method (DEM) and the Finite Volume Method (FVM)

Acknowledgments

I would like to thank both of my supervisors Professor Chris Pearce and Doctor Łukasz Kaczmarczyk for their help and support throughout my PhD. They were a constant source of guidance, ideas, motivation and support throughout this project. I would also like to thank them for their belief in me during tough times. Their experience and judgement also proved invaluable at many times over the course of this research.

I would also like to thank my parents Colm and Patricia Kelly for their continued support throughout this project and indeed throughout my education in general.

My colleagues and friends, Caroline, Ross, Dimitrios X., Ignatios, Graeme, Michael, Dimitrios K., Xue, Ali, Euan M., Julien and James all deserve many thanks for their help with my research and for the many fun times we shared in our office.

I would also like to thank my girlfriend Jeanne for her support and patience, especially during the final stages of this project.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The Finite Element Method (FEM) and its many variants (Boundary Element Method, Finite Volume Method etc) are the methods most often used for solving complex problems in many areas of science and engineering. These methods involve discretising the domain into a mesh which is composed of elements. The choosen mesh can have a significant impact on the accuracy of the solution [1], therefore it is crucial that the best possible mesh is used. The quality of a mesh is a function of the quality of its elements and the quality of an element can be described as a numerical measure which estimates the effect that the size/shape of an element will have on the accuracy of an analysis [1]. There is a strong link between mesh geometry and the accuracy and efficiency of a simulation and it can be shown that poor quality elements can result in both interpolation errors and poor conditioning of the stiffness matrix. In the extreme, a single poor element can render a problem intractable. Therefore, a high quality mesh is crucial to performing an analysis.

Mesh optimisation is the process of relocating the nodes of a mesh to increase its quality. It is based on the techniques of numerical optimisation which is the process of maximising or minimising an objective function, subject to constraints on the solution, for example, the boundary of the mesh or the maximum element size. The field of mesh optimisation is complex and has now become an area of research in its own right. It is important to recognise that an analyst will not generally be an expert in mesh optimisation. Therefore, in order to make the process of mesh optimisation more straightforward, this project aims to create a set of tools which make it possible for an analyst to improve a complex meshes used in actual simulations, in as simple a manner as possible. This involves simplifying a very complex process; in this thesis the problems encountered are explained and proposed solutions to these problems are presented.

The motivation for this project has come from the problems encountered by colleagues in my research group. This group is concerned with the computational modeling of materials and structures, including fracture, microfluids, surface tension and biological materials. These problems are characterised by evolving surfaces which consequently require an evolving mesh that will enable the problems to be solved accurately and efficiently. The initial mesh in such cases may be adequate for its intended purpose, however, during deformation its quality can quickly deteriorate to such an extent that numerical errors become unacceptably large. Additionally, for complex three-dimensional domains, automatic mesh generators do not always create meshes of sufficient quality to ensure a satisfactory level of accuracy in the solution. In the problems mentioned above, many numerical issues have been traced back to poor quality meshes and, although there are a number of tools already available for improving mesh quality, none of them have matched the needs of the group. Therefore, the decision was made to develop a new set of tools, either via the modification of existing open-source software or through the development of new tools.

In FE simulations, the mesh is required to conform to the shape of the domain, even in cases where this may be quite complex. In fact such are the problems associated with generating meshes of complex domains, the mesh generation process is the bottleneck in many FE simulations [2], thus, mesh generators often struggle to produce suitable meshes. Such meshes must therefore be optimised before being used in simulations. In other simulations, the initial shape may be relatively straightforward, however, as the simulation progresses, the domain undergoes large deformations. The mesh must therefore adapt to the deformed domain which can rapidly lead to a deterioration in the quality of the mesh, meaning that it must be optimised.

The optimisation of meshes of complex domains often presents an additional difficulty - the worst elements in the mesh often have nodes on the boundary of the domain. The relocation of these nodes, for the purposes of optimisation, must not alter the domain shape or volume as doing so would adversely affect the accuracy of the simulation. For example, in the modelling of free surface problems in micro-fluids (e.g. droplet of water), surface tension must also be modelled. The positions of the surface nodes are determined by the physics of the problem and therefore their movement is constrained, i.e. the shape and the volume of the domain must be preserved in any mesh optimisation procedure. Therefore a method of improving mesh quality by moving surface nodes but without changing the shape or volume of the domain is necessary. Such a method has been developed and is described in Chapter 6.

The movement of mesh nodes in many of the problems under consideration is determined using an Arbitrary Lagrangian Eulerian (ALE) formulation, where the need to ensure mesh quality in an evolving mesh is very important. This is a form of finite element analysis where the positions of the mesh nodes may be determined in a Lagrangian manner, whereby they track a material point or an Eulerian manner where the mesh is fixed and the continuum moves with respect to the mesh or in some arbitrary combination of these [3]. As the mesh evolves, elements may become distorted, leading to numerical issues. Therefore, the mesh must be improved regularly during the analysis to limit the numerical errors associated with it. The problems associated with re-meshing are discussed in Chapter 2, and it is shown why it is desirable to limit re-meshing. However, this conflicts with the need to ensure that the adapted mesh is of adequate quality. Therefore, a method of ensuring that the adapted mesh is both compatible with physical equilibrium and is of adequate quality is required. The development of such a method is described in Chapter 7. This is an example of a situation where mesh optimisation and the solution of the physical problem must be considered as a holistic process, and not as separate processes.

Many FE simulations involving the approximation of highly anisotropic functions. In Chapter 2 the relationship between a function and the mesh used to discretise it is explored. A quality measure suitable for assessing anisotropic elements is presented in Chapter 3 as well as a method of adapting isotropic quality measures for use with anisotropic elements. Although the focus of this research is on isotropic meshes, all of the algorithms and techniques presented are suitable for use with anisotropic meshes.

Chapter 2

Motivation

The primary goal of this project is to develop a methodology to reduce the numerical errors associated with poor quality meshes used in finite element analyses. In order to achieve this, it is necessary to first understand the source of these errors and also evaluate the strength and deficiencies of current mesh optimisation techniques and software in order to focus the research in the correct direction. The next section will show how the chosen mesh geometry directly affects the magnitude of interpolation errors and the conditioning of the stiffness matrix assembled from the mesh.

2.1 Errors induced by poor meshes

2.1.1 Interpolation errors

The seminal paper of Shewchuk [1] derives bounds for both interpolation and gradient interpolation errors. In this important paper, large angles are found to be the source of both interpolation and gradient interpolation errors. Following Shewchuk [1], a simple numerical experiment is presented here which demonstrates why large angles are so detrimental to the accuracy of an interpolated function.

Let T be a triangular mesh with a continuous scalar function $f(x, y)$, defined in the domain of the mesh and let $g(x, y)$ be a piecewise linear approximation of $f(x, y)$ with $g(x_i, y_i) = f(x_i, y_i)$, where (x_i, y_i) is the Cartesian coordinate of node i. Therefore, $\nabla g(x, y)$ is piecewise constant. Three different meshes, Figure 2.1, are used to approximate the function $f(x, y) = x^2 + \frac{1}{2}y^2$ in the domain $-0.5 \leq x \leq 0.5$ and

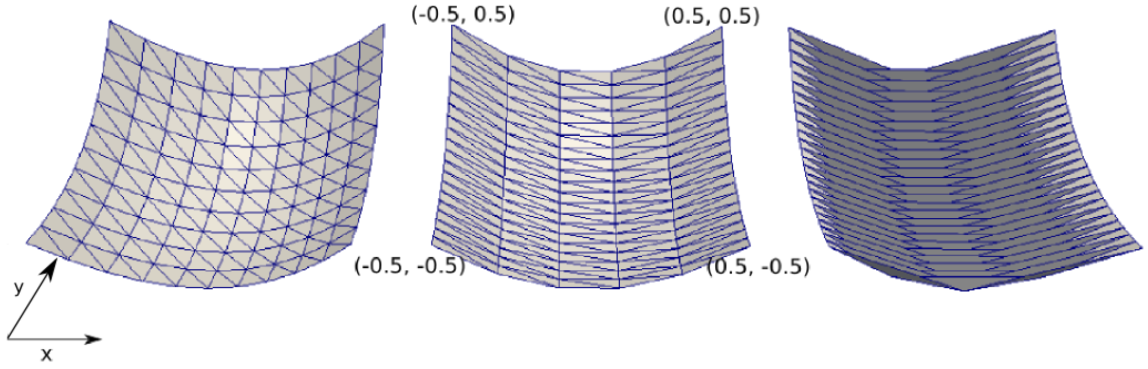


Figure 2.1: The three meshes used to approximate the function $f(x, y) = x^2 + \frac{1}{2}y^2$. Each mesh is of the same domain which is indicated on the central mesh, $-0.5 \leq x \leq 0.5$ and $-0.5 \leq y \leq 0.5$, and has 200 triangles.

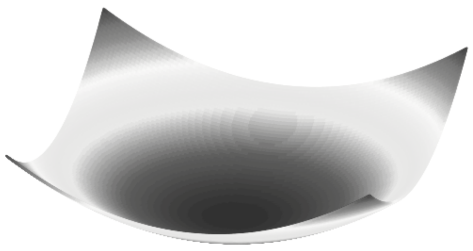


Figure 2.2: Plot of analytical function, $f(x, y) = x^2 + \frac{1}{2}y^2$

$-0.5 \leq y \leq 0.5$. Each mesh contains 200 triangles. The mesh on the left consists of isosceles triangles with no extreme angles, the middle mesh has small angles but no angle greater than 90° and the mesh on the right has both large and small angles. The triangles in the central mesh are stretched in the x direction and squeezed in the y direction. This effect is enhanced for mesh 3. The function being approximated is a smooth, strictly convex function and it can be seen that mesh 1 provides a very good approximation of the actual function, which is shown in Figure 2.2. Mesh 2 performs only slightly worse than the first; when compared with Figure 2.2 it can be seen that it provides a reasonable approximation of $f(x, y)$, although not as good as that provided by mesh 1. This is due to the presence of longer edges in mesh 2. However, it can clearly be seen that mesh 3 performs very poorly. This is due to the presence of large angles.

To quantify these interpolation errors, it is useful to compare the analytical function with the interpolated function at the centroid of each triangle for all three meshes. Similarly, it is helpful to compare the analytical gradient with its interpolated counterpart. The greatest interpolation error and gradient interpolation error for each mesh

	Mesh 1	Mesh 2	Mesh 3
$\%\lvert\frac{f-g}{f}\rvert_{max}$	1.31	3.77	39.5
$\%\frac{\lvert\nabla f-\nabla g\rvert}{\lvert\nabla f\rvert}_{max}$	4.55	6.72	613.73

Table 2.1: The greatest percentage interpolation and gradient interpolation error for each of the three meshes in Figure 2.1.

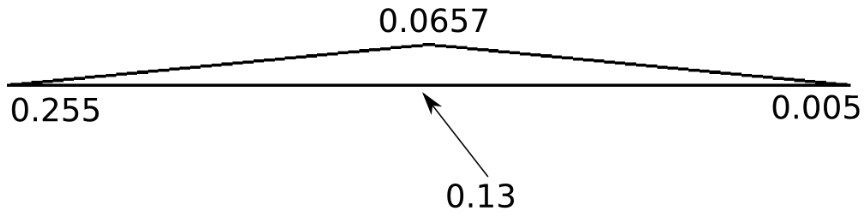


Figure 2.3: The element with the greatest gradient interpolation errors in Mesh 3. Function values are indicated at the vertices and the interpolated function value is shown.

is shown in Table 2.1. The largest interpolation error for mesh 1 is 1.3%, mesh 2 performs slightly worse at 3.77% whereas the error for mesh 3 is almost 40%. The gradient interpolation errors tell a similar story. Mesh 1 approximates the gradient within 5% of the analytical value and mesh 2 is within 7% of the analytical value. However, the greatest gradient interpolation error on mesh 3 is over 600%, rendering the approximation of this function and its gradient worthless. This simple example demonstrates just how great an effect large angles can have on the accuracy of interpolated functions and on their gradients. In many problems, for example, deformation of materials, the gradient of the primary function (i.e. strain) is more important than the function itself. Therefore the accuracy of the interpolated gradients are also of primary interest.

The source of these errors can be investigated by examining the element of Mesh 3 which gave the greatest gradient interpolation errors, Figure 2.3. The value of the function is shown at each node and the value shown at the midpoint of the bottom edge is the interpolated value, $\frac{0.255+0.005}{2}$. This interpolated value is independent of the value of the top node. As the angle at the top node approaches 180°, it becomes closer and closer to the interpolated point. Therefore the vertical component of ∇g rapidly becomes very large making it a very poor approximation of ∇f even though $f = g$ at each node.

Consider now the numerical integration of the function f over the domain. The effect of

	Mesh 1	Mesh 2	Mesh 3
% error	-2.0	-5.5	-20.874

Table 2.2: Numerical integration errors for each mesh

extreme internal angles is now demonstrated. A three point integration rule is adopted which is sufficient to accurately integrate the function that is approximated using linear interpolation functions. Thus any errors will be the result of an interpolation error rather than insufficient integration points.

The larger an angle, the longer the edge opposite it, thus the larger the interpolation error. It is trivial to analytically integrate the chosen function to get an exact answer. Table 2.2 compares exact integration of the function with the numerical result. The solutions for both Meshes 1 and 2 are reasonably accurate whereas the solution obtained for Mesh 3 is very inaccurate. In the context of the Finite Element method, this simple experiment clearly shows the need for high quality meshes as the quality of the solution deteriorates rapidly as the quality of the mesh deteriorates. It is worth noting that, in this very simple case, the use of quadratic elements instead of linear elements would have yielded the correct solution. However in practice the function being approximated can be orders of magnitude greater than the interpolation function or not a polynomial.

These simple numerical experiments clearly demonstrate how the presence of large angles in meshes can lead to errors in the FE solution.

2.1.2 The approximation of functions on anisotropic meshes

Many physical problems which are analysed using FEA involve functions whose solutions vary significantly more in one direction than in the others [4]. Equilateral elements are well suited to solution fields that vary equally in every direction however, the use of isotropic elements in problems where the solution is highly anisotropic will result in meshes which are prohibitedly large [5]. For example, problems involving boundary layers and shock waves or flow problems with high Reynold's number often exhibit significant anisotropy. It may be advantageous to the accuracy and efficiency of the numerical solution of such problems to use an anisotropic mesh. Furthermore, the degree of anisotropy may vary over the entire mesh, meaning the definition of the ideal element may be a function of the position of the element in the mesh. This is

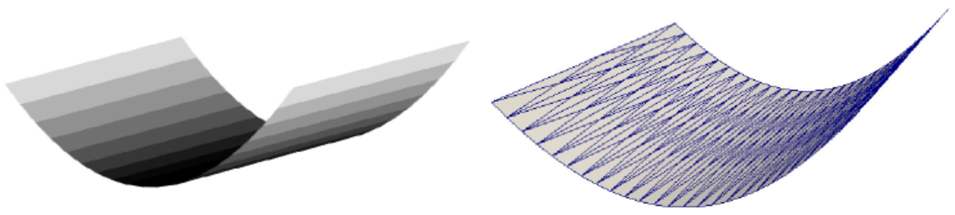


Figure 2.4: (a) Plot of the function $f(x, y) = x^2$. and (b) its discretisation using Mesh 3.

also true for functions which vary equally in each direction, as the magnitude of this variation may change from point to point, meaning that although equilateral elements may be suitable throughout the mesh, the size of the ideal element may vary according to the position in the mesh.

In the previous section, the interpolation errors for three different meshes used to approximate a function, $f(x, y) = x^2 + \frac{y^2}{2}$, were quantified. The curvature of this function is both a function of position and direction. It was concluded in the previous section that large angles lead to interpolation errors, and thus errors in the FE solution. This is because the edge opposite a large angle is much longer than the other two edges of the element. However, if the function being approximated is constant or varies linearly over this edge, the presence of the large angle does not introduce errors.

This concept may be demonstrated by examining the function $f(x, y) = x^2$, Figure 2.4. This function is constant in the y direction, therefore the approximation of this function using Mesh 3 is in fact quite accurate, despite Mesh 3 between completely unsuitable in the previous example. This means that the choice of mesh is heavily influenced by the function it must approximate. If the curvature of the function is isotropic, then isotropic elements should be used. For functions whose curvature is anisotropic, anisotropic elements, when correctly orientated, may be the most suitable. This is because elements can be stretched to adapt to features which are of equal or lower dimensionality than the element shape functions [6]. This then leads to the concept of curvature adaptive meshing where the size, shape and orientation of elements is chosen so as to account for the curvature of the function being approximated on the mesh.

2.1.2.1 Curvature adaptive meshing

In the previous section it was shown that the suitability of a mesh for interpolating a function is intrinsically linked to the curvature of the function that the mesh is approximating. This means that the mesh generation and optimisation processes should, where possible, take this into account. The curvature of the underlying function may be accounted for by defining a metric field, which is discussed in detail in the Section 2.1.3, based on a computed solution on an existing mesh of a domain [7]. This is then used as an input to a suitable mesh generation or optimisation package [5]. For example, in a steady-state problem, where the solution does not change with time, a solution may be obtained using an isotropic mesh. Using this solution, a metric field is then defined and used to either re-mesh all or part of the domain or to optimise the mesh. The problem is then solved again using the new mesh. This process may be repeated until a satisfactory solution is obtained. Such a technique is suitable for problems where the computational cost is relatively low thus allowing for multiple iterations of the solution process on continuously improving meshes. However, the computational cost of solving many interesting problems is such that this technique would be too expensive.

Schoen [2] proposes an alternative technique where the problem is solved on a coarse isotropic mesh to obtain a rough estimate of the solution. Using metrics derived from this mesh, a finer anisotropic mesh may be generated and a much more accurate solution obtained. However, important features of the solution may not be captured by the coarse mesh [6], thus they would not be accounted for by this process. This method may be quite effective for repetitive steady-state calculations but it is not at all suitable for transient problems [6] where the solution changes with time, such as the problem studied in Chapter 7. This is both for cost and accuracy reasons. In such a problem, the solution is calculated at intervals, or time-steps, using the solution at the previous time-step. The solution at the start time is calculated using the initial conditions. A metric field is calculated using the computed solution and the mesh is adapted. At this stage, either the solution is re-calculated using the adapted mesh, or the solution at the following time-step is calculated. In either case, a solution must be transferred from an existing mesh to the new mesh. As shown in Section 2.1.1, the interpolation of a solution from one mesh to another is a source of errors, thus, this would lead to an inaccurate solution. At the starting time, the inital conditions may be applied to an adapted mesh, such as in a steady-state problem, without any errors. This is because the intial conditions are generally defined on the undiscretised domain, and not on the discretised domain as is the case at intermediate time-steps. This is also true for some steady-state problems where the load may be applied in stages so that the entire solution path may be traced and to limit the accumulation of solution errors [8]. In the following sections, the process of calculating a metric field is explained and several possible methods proposed in the literature of adapting a mesh based on the metric field are described.

2.1.3 The calculation of a metric field

A metric tensor is a function for calculating the distance between any two points in a space [9]. Therefore, the explanation of a metric tensor requires the formal definition of length [10]. The definition of length in a metric space requires both the metric used to define the space and a suitable definition of the dot product. For any point P in $\mathcal{R}^d$, a metric tensor is a $d \times d$ symmetric positive definite matrix $\mathcal{M}(P)$. In two dimensions, $\mathcal{M}(P)$ takes the following form:

$$\mathcal{M}(P) = \begin{pmatrix} a & b \\ b & c \end{pmatrix} \tag{2.1}$$

where $a > 0$ and $c > 0$ and its determinant $ac - b^2 > 0$. This tensor must be symmetric positive definite so that its geometric representation is an ellipse [5]. The coefficients of the metric depend on the position P. This induces a Riemannian structure over $\mathcal{R}^d$. If the coefficients of this matrix do not depend on P then the classical Euclidean case is defined, where the metric is not a function of position [10].

Using equation 2.1, the dot product for a given metric $\mathcal{M}(P)$ in Euclidean space for two vectors $\mathbf{u}$ and $\mathbf{v}$ is defined as:

$$\langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{M}(P)} = \mathbf{u}^T \mathcal{M}(P) \mathbf{v} \tag{2.2}$$

Using the standard Euclidean norm, the norm of a vector is defined as:

$$\|\mathbf{u}\| = \sqrt{\langle \mathbf{u}, \mathbf{u} \rangle_{\mathcal{M}(P)}} = \sqrt{\mathbf{u}^T \mathcal{M}(P) \mathbf{u}} \tag{2.3}$$

The calculation of the length of a vector is more complicated when the metric is a function of the point P. For example, take a vector $\mathbf{u}$ connecting points a and b. To calculate the length of this vector, the metrics at both a and b and each intermediate metric must be considered [11]. This is achieved by using a parameterised segment, $\gamma(t) = a + t\mathbf{u}$. Therefore the length of a vector connecting two points, where the metric is a function of position is defined in equation 2.4.

$$l_{\mathcal{M}}(\mathbf{u}) = \int_0^1 \sqrt{\mathbf{u}^T \mathcal{M}(t) \mathbf{u}} \tag{2.4}$$

The is in contrast to the method presented by Thompson et al [7] where the length of a vector is calculated using the average of the metric at both nodes.

2.1.4 Construction of a metric tensor using error indicators

The most common method of calculating metric tensors for use on finite element meshes involves analysing error indicators. The simplest error indicator is obtained by evaluating the jump or absolute difference of the variable being interpolated along an edge [6]. A much more common method involves estimating the approximation error of the discretised function. Assuming the function is smooth, the error may be estimated to one order greater than the order of the element shape functions [6]. The aim of this process is to construct a metric which gives a homogeneous distribution of the interpolation error over the entire mesh. This will be done in one dimension using a regular function u which is defined on the segment $[a, b]$. Let h be the length of segment $[a, b]$ which is not necessarily small and $\Pi_h u$ be the P_1 interpolation of the function u on $[a, b]$ meaning that it is linear and piecewise continuous [12].

The following derivation is adapted from Frey and George [10]. As the interpolation scheme is of P_1 type, the interpolation error is related to the variation of the various variables being approximated on $[a, b]$ and in particular, to their gradients and Hessians. The function and its approximation are equal at a and b meaning that $\Pi_h u(a) = u(a)$ and $\Pi_h u(b) = u(b)$. The approximation error is defined as $e(x) = u(x) - \Pi_h u(x)$.

Consider the function $\Pi_h u$ on the segment $[\Pi_h(a), \Pi_h(b)]]$ which is the linear approximation of the function u on the segment between a and b. It is assumed that the computed solution is quite close to the actual solution. This analysis is based on a variation of Taylor's formula. Since h is not necessarily small, a formula of the type shown in equation 2.5 is preferable to a standard Taylor series expansion [10].

$$f(x) = f(a) + (a-x)f'(x) + \frac{(a-x)^2}{2} f''(x + t(a-x)) \tag{2.5}$$

where t, defined on the segment $[0, 1]$, is a function of both x and a. For this analysis, f is defined as $(u - \Pi_h u)(x) = u(x) - u_h(x)$ on the segment $[a, b]$. Therefore the error at point a is approximated using equation 2.6.

$$e(a) = (u-\Pi_h u)(a) = (u-\Pi_h u)(x) + (a-x)(u-\Pi_h u)'(x) + \frac{(a-x)^2}{2}(u-\Pi_h u)''(x+t_1(a-x)) \tag{2.6}$$

However, as previously stated, the function u and its approximation $\Pi_h u$ are equal at a meaning that the left hand side is zero. This yields the following simplified expression:

$$0 = (u - \Pi_h u)(x) + (a-x)(u - \Pi_h u)'(x) + \frac{(a-x)^2}{2}(u - \Pi_h u)''(x + t_1(a-x)) \tag{2.7}$$

As the largest interpolation error is of interest here, the extremum x where $(u - \Pi_h u)'(x) = 0$ is sought. Therefore this expression may be further simplified.

$$0 = (u - \Pi_h u)(x) + \frac{(a-x)^2}{2}(u - \Pi_h u)''(x + t_1(a-x)) \tag{2.8}$$

A similar expression for b may also be constructed:

$$0 = (u - \Pi_h u)(x) + \frac{(b-x)^2}{2}(u - \Pi_h u)''(x + t_2(b-x)) \tag{2.9}$$

Adding equations 2.8 and 2.9 gives:

$$0 = 2(u - \Pi_h u)(x) + \frac{(a-x)^2}{2}(u - \Pi_h u)''(x + t_1(a-x)) + \frac{(b-x)^2}{2}(u - \Pi_h u)''(x + t_2(b-x)) \tag{2.10}$$

Let M be the majorant of u'' on the segment $[a, b]$. Therefore:

$$|(u - \Pi_h u)(x)| \leq \frac{1}{2}\left(\frac{(a-x)^2}{2} + \frac{(b-x)^2}{2}\right) M \tag{2.11}$$

Then:

$$|(u - \Pi_h u)(x)| \leq \frac{1}{4} \max_{x \in [a,b]} \left((a-x)^2 + (b-x)^2\right) M \tag{2.12}$$

The maximum is reached for $\frac{a+b}{2}$, implying that, $\forall x \in [a, b]$:

$$|e(x)| = |(u - \Pi_h u)(x)| \leq \frac{(b-a)^2}{8} M \tag{2.13}$$

The value of $\frac{(b-a)^2}{8}M$ is compared with a given value ϵ, which is the maximum allowable error or the maximum allowed gap between the function u and its linear approximation $\Pi_h u$. In many transient problems this error tolerance may be required to vary over the domain [6]. The calculation of M via the recovery of the Hessian on the segment $[a, b]$ is described in the next section.

Hessian recovery

The next step in developing a metric tensor based on error analysis involves recovering the Hessian of the function being approximated. The following derivation is adapted from Löhner [6]. This method involves the assumption that the Hessian may be expressed using shape functions $\widetilde{\mathbf{N}}$. Therefore:

$$\frac{\partial^2 u}{\partial s^2} \approx \widetilde{\mathbf{N}}\mathbf{u}'' \tag{2.14}$$

where $\mathbf{u}$ is the nodal values of u and $\mathbf{u}''$ is the nodal values of the Hessian of u. The direction s at which the derivative is taken with respect to is arbitrary and could refer to x, y or z. The second assumption is that the function u may be expressed using shape functions. Therefore:

$$u \approx \mathbf{N}\mathbf{u} \tag{2.15a}$$

$$\frac{\partial u}{\partial s} \approx \frac{\partial \mathbf{N}}{\partial s}\mathbf{u} \tag{2.15b}$$

$$\frac{\partial^2 u}{\partial s^2} \approx \frac{\partial^2 \mathbf{N}}{\partial s^2}\mathbf{u} \tag{2.15c}$$

Combining equations 2.14 and 2.15(c) gives:

$$\widetilde{\mathbf{N}}\mathbf{u}'' \approx \frac{\partial^2 \mathbf{N}}{\partial s^2}\mathbf{u} \tag{2.16}$$

This is then weighted with shape functions $\mathbf{w}$:

$$\int_\Omega \mathbf{w}^T \widetilde{\mathbf{N}} d\Omega \mathbf{u}'' = \int_\Omega \mathbf{w}^T \frac{\partial^2 \mathbf{N}}{\partial s^2} d\Omega \mathbf{u} \tag{2.17}$$

The right hand side is then integrated by parts to give:

$$\int_\Omega \mathbf{w}^T \widetilde{\mathbf{N}} d\Omega \mathbf{u}'' = -\int_\Omega \frac{\partial \mathbf{w}^T}{\partial s}\frac{\partial \mathbf{N}}{\partial s} d\Omega \mathbf{u} + \int_\Gamma \mathbf{w}^T \mathbf{n} \frac{\partial \mathbf{N}}{\partial s} d\Gamma \mathbf{u} \tag{2.18}$$

where $\mathbf{n}$ is the unit normal. For the special case corresponding to the Galerkin weighted residual method, where $\mathbf{W} = \widetilde{\mathbf{N}} = \mathbf{N}$:

$$\mathbf{M}\mathbf{u}'' = \int_\Omega \mathbf{N}^T \mathbf{N} d\Omega \mathbf{u}'' = -\int_\Omega \frac{\partial \mathbf{N}^T}{\partial s}\frac{\partial \mathbf{N}}{\partial s} d\Omega \mathbf{u} + \int_\Gamma \mathbf{N}^T \mathbf{n} \frac{\partial \mathbf{N}}{\partial s} d\Gamma \mathbf{u} \tag{2.19}$$

This derivation is valid in one, two or three dimensions.

Calculating a metric tensor using the Hessian

With the Hessian recovered on the segment $[a, b]$, the error may now be calculated using equation 2.13. If the error is below the threshold, ϵ, then the segment $[a, b]$ provides a suitable discretisation of the function u between a and b. If the calculated error is greater than the threshold, then the target edge length h may be calculated using equation 2.20.

$$h = \sqrt{\frac{8\epsilon}{M}} \tag{2.20}$$

where M is the majorant of u'' on $[a, b]$. The required metric is then calculated as follows:

$$\mathcal{M} = \frac{I_d}{h^2} \tag{2.21}$$

2.1.4.1 Calculation of the metric tensor in multiple dimensions

The metric tensor may also be calculated in two or three dimensions [10] in the same manner as for the one-dimensional case shown in the previous section. In two dimensions the metric is a 2×2 matrix which may be diagonalised as follows:

$$\mathcal{M} = \mathcal{R} \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \mathcal{R}^{-1} \tag{2.22}$$

where $\mathcal{R}$ is a rotation matrix and λ_1 and λ_2 are the eigenvalues of $\mathcal{M}$. This corresponds to an ellipse in the coordinate system defined by $\mathcal{R}$:

$$\frac{\lambda_1}{\epsilon} x^2 + \frac{\lambda_2}{\epsilon} y^2 = 1 \tag{2.23}$$

where ϵ is the acceptable error. The interpolation error of the function u on any triangle either lying on or within this ellipse is less than or equal to ϵ in any direction.

2.1.4.2 Combining multiple metrics

Many FE simulations involve solving for several variables on the same mesh. Therefore, the mesh must be suitable for the interpolation of each variable, thus it must account for the different anisotropy that each variable may exhibit. A metric must be calculated for each variable and these must be combined into a unique metric which reflects the nature of its constituent metrics [10].

Consider the intersection of two metrics in two-dimensions and their respective ellipses. This may be extended to three dimensions using ellipsoids. The desired solution is a metric associated with the two original metrics. In general, the result is not an ellipse, therefore an ellipse which fits inside the intersection area is used [10]. This is referred to as an intersection metric. Different solutions exist, depending on the choice of ellipse within the intersection region. The largest ellipse fitting within this region may be used or the direction of one of the initial ellipses may be preserved.

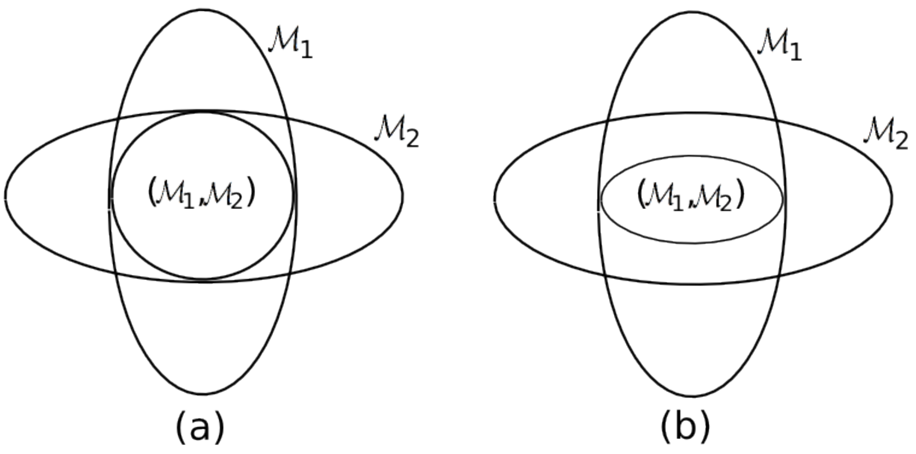


Figure 2.5: Intersection of two metrics, $\mathcal{M}_1 \cap \mathcal{M}_2$. (a) The simultaneous reduction of two metrics and (b) the reduction of two metrics preserving the directions of $\mathcal{M}_2$, adapted from [10].

Simultaneous reduction of two metrics:

Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be the two metrics and their respective ellipses may be expressed as:

$$\mathbf{u}^T\mathcal{M}_1\mathbf{u} = \lambda_1 x^2 + \lambda_2 y^2 = 1 \qquad \mathbf{u}^T\mathcal{M}_2\mathbf{u} = \mu_1 x^2 + \mu_2 y^2 = 1 \tag{2.24}$$

where λ_i and μ_i are the eigenvalues of $\mathcal{M}_1$ and $\mathcal{M}_2$ respectively. The intersection metric, $(\mathcal{M}_1 \cap \mathcal{M}_2)$, is defined as:

$$(\mathcal{M}_1 \cap \mathcal{M}_2) = \mathbf{P}^{-T} \begin{pmatrix} max(\lambda_1, \mu_1) & 0 \\ 0 & max(\lambda_2, \mu_2) \end{pmatrix} \mathbf{P}^{-1} \tag{2.25}$$

As the two metrics may have different canonical bases, they must be mapped to that associated with the simultaneous reduction of the two metrics using the matrix $\mathbf{P}$. This is illustrated in Figure 2.5(a). When more then two metrics are defined at a point, their simultaneous reduction is defined as:

$$(\mathcal{M}_1 \cap \cdots \cap \mathcal{M}_q) = ((\cdots((\mathcal{M}_1 \cap \mathcal{M}_2) \cap \mathcal{M}_3) \cap \cdots) \cap \mathcal{M}_q) \tag{2.26}$$

Metric intersection preserving specific directions:

The simultaneous reduction of two metrics described above does not preserve the directions of either of the initial metrics. If this is of interest, the directions of one of the initial metrics may be preserved by finding the maximal ellipse with the required directions within the intersection region. For example, in Figure 2.5(b) the directions of $\mathcal{M}_2$ are preserved. The intersection metric is defined as:

$$(\mathcal{M}_1 \cap \mathcal{M}_2) = \omega \mathcal{M}_1 \tag{2.27}$$

where $\omega = \max(\frac{\mu_1}{\lambda_1}, \frac{\mu_2}{\lambda_2}, 1)$.

Modifying a mesh based on a metric field

In this section several methods of modifying a mesh based on a metric field are described. The simplest method involves re-meshing the domain using the metric field and a suitable mesh generator. Adaptive mesh refinement and coarsening may also be used where nodes are added to zones of high curvature and removed from zones of low curvature. A new triangulation is then calculated. Both of these methods involve large amounts of interpolation from the original mesh to the new mesh.

A more sophisticated method is mesh adaptation which is the process of modifying a mesh based on a computed solution so as to more accurately capture features of the solution, without significantly increasing the number of degrees of freedom. Li et al [5] describes a mesh adaptation procedure involving node insertions and removals and edge and face swapping operations based on a calculated metric field. This algorithm takes as input a minimum and maximum allowed edge length and iteratively corrects each edge in the mesh until the length of as many edges as possible in the transformed space is within the required range. The anisotropic mesh smoothing procedure described by Li et al [13] is then applied. This mesh smoothing algorithm takes as input the piecewise metric field defined on each node and a threshold value. It adjusts the position of each node connected to an edge whose length is outside the required range. For the purposes of mesh smoothing, several alternatives methods have been proposed in the literature for calculating the length of an edge in a metric space to that presented in equation 2.4. Jiao et al [14] propose taking the average value of the metric at every neighbour of a node where as Buscaglia et al [15] propose taking the metric with the greatest determinant.

In the previous sections the interpolation errors associated with poor quality meshes were explained and the notion that the defintion of an ideal element, in terms of interpolation errors, can vary from point to point and may not be isotropic. In the following section, the effect that poor quality elements can have on the conditioning of a finite element stiffness matrix is discussed.

2.1.5 Stiffness matrix conditioning

Shewchuk [1] also examines the mathematical relationship between both element shape and size and the condition number of a FE stiffness matrix. In the FEM, the stiffness matrix represents the system of linear equations that must be solved to obtain an approximate solution to the governing partial differential equation:

$$\mathbf{K}\mathbf{u} = -\mathbf{f} \tag{2.28}$$

where $\mathbf{K}$ is the global of system stiffness matrix. $\mathbf{K}$ is assembled from the stiffness matrix of each element $\mathbf{k}^i$ in the mesh.

$$\mathbf{K} = \underset{i=1}{\overset{n}{\mathsf{A}}} \mathbf{k}^i \tag{2.29}$$

where A is the standard assembly operator and n is the number of elements in the mesh. Shewchuk concludes that both large and small angles can cause the stiffness matrix to be poorly conditioned. The condition number of a matrix, κ, is defined as:

$$\kappa = \frac{\lambda_{max}}{\lambda_{min}} \tag{2.30}$$

where λ_{max} and λ_{min} are the largest and smallest eigenvalues of the matrix. A matrix with a large condition number means that the linear system is harder to solve. This is because if $\kappa = 10^k$, then one can expect to lose k digits of precision in solving the linear system [16] for the type of sparse matrices generally encountered in FE simulations. If the condition number is greater than machine precision, the system is unsolvable. As it approaches machine precision, accuracy is lost by a direct solver and the time taken for an iterative solver to obtain a solution increases.

λ_{min} is tied to the properties of the physical system being modelled and to the sizes of the elements with its lower and upper bounds proportional to the area/volume of smallest/largest elements [1]. This suggests that mesh generators should aim to create uniform mesh. However, this is not always possible as element size is often dictated by physical and computational requirements. This is further complicated by the need to require a mesh to reflect an evolving problem, such as the development of a large strain. For efficiency and computational reasons it is not sensible to refine an entire mesh to the level of local refinement required near these large gradients. Therefore, the analyst typically has very limited control over the size of elements (and so λ_{min}) during the mesh generation and refinement stages. For this reason it is prudent to focus on minimising λ_{max}, which is largely influenced by element shape. λ_{max} has been shown to be limited to a scalar multiple of the greatest element stiffness matrix eigenvalue by the expression, [1]:

$$\max_i \lambda^i_{max} \leq \lambda^K_{max} \leq m \;\max_i \lambda^i_{max} \tag{2.31}$$

Where λ^i_{max} is the largest eigenvalue associated with i^{th} element's stiffness matrix, λ^K_{max} is the largest global stiffness matrix eigenvalue and m is the maximum number of elements which correspond with one node. Therefore λ_{max} can be made arbitrarily large by one poorly shaped element. The mesh optimisation strategy adopted in this thesis is drawn from this statement, to optimise a mesh so that the quality of the *worst* element reaches an acceptable level.

Shewchuk [1] derives a quality measure which directly limits the magnitude of the largest eigenvalue to a narrow bound. This is done for an element stiffness matrix assembled from the Poisson equation which in two dimensions is 3×3 for triangular elements and in three dimensions is 4×4 for a tetrahedral element. For the two dimensional case, an expression for the roots of the characteristic polynomial, i.e. the eigenvalues, is derived. In the three dimensional case, the characteristic polynomial is too complex for the roots to be found. Furthermore, the first derivative and, depending on the type of solver being used, the second derivative, of a function are required to optimise a mesh based on that function. It will be shown in Chapter 3 how complex the derivatives of multi-dimensional functions can be. A typical FE stiffness matrix for a two dimensional element used in stress analysis is a 6×6 matrix, for a three-dimensional tetrahedron it is a 12×12 matrix. The composition of this matrix strongly depends on the physical equations being solved, thus making it impossible to derive a general expression. Instead it would be necessary to derive a separate expression for every type of analysis being performed. Also, the calculation of an expression for the eigenvalues of such a matrix would be extremely complicated and such an expression, if successfully derived, would be unsuitable for use as part of the mesh optimisation process due to its complexity. Therefore, it is neither practicable nor realistic to optimise a mesh based on an expression for its eigenvalues, although it is an interesting theoretical exercise.

It is only possible to optimise a function which closely correlates to the maximum eigenvalue of the element stiffness matrix. Many such measures exist and these are discussed in the next chapter.

2.1.5.1 Conclusions and recommendations

It is clear from the previous two sections that poor quality meshes can lead to many problems with FEA. Re-meshing is often used during an analysis to eliminate poor quality elements; however, the data must be transferred using interpolation from the poor mesh to the new mesh, thus introducing errors. This is also true if a poor mesh is optimised, although the amount of interpolation will be less. Regardless of which technique is used to heal poor quality meshes, numerical errors will be introduced. Therefore it is desirable to begin an analysis with the highest quality mesh possible, through a combination of a high quality mesh generation algorithm and optimisation. In the case that a mesh is forced to evolve during an analysis, the mesh quality must be maintained as much as possible throughout the evolution process.

2.2 A review of current mesh optimisation software

Perhaps the two state of the art mesh optimisation packages available today are Mesquite [17] and Stellar [18] and this thesis will focus attention on them and compare them with the algorithms and numerical tools developed herein. In this section, the very different mesh optimisation strategies adopted by each of these packages will be discussed and the reasons behind the decision to develop new tools will be explained. The terms mesh optimisation, mesh improvement and mesh smoothing are often used interchangeably. To ensure clarity in the rest of this thesis each of these terms shall be defined using the definitions supplied by Knupp [19]:

- Mesh Smoothing: The term smoothing refers to methods for relocating a mesh's nodes which are not based on optimisation techniques.
- Mesh Optimisation: The process of finding the optimal position of mesh nodes without changing the mesh connectivity.
- Mesh Improvement: The process of improving the quality of a mesh. This includes mesh optimisation and topological transformations.

Other terms which are used throughout this thesis are defined below:

- Connectivity: An element's connectivity is the list of nodes which define it.
- Topological transformation: The process of modifying the topology of a mesh. This includes modifying the connectivity of elements and node insertion or removal.
- Inverted element: An inverted element is an element whose Jacobian matrix has a negative determinant.
- Solution space: All the elements which are affected by moving a node.

2.2.1 A comparison between Mesquite and Stellar

Overview

Stellar was written by Bryan Klingner as part of his PhD thesis, [18]. The goal of Stellar is to bring the worst element in a three dimensional tetrahedral mesh to an acceptable quality, regardless of the time taken to achieve this. Stellar uses both mesh optimisation and topological transformations to improve a mesh. Stellar does not have two dimensional functionality, although two dimensional analogues exist for most of its algorithms.

The Mesh Quality Improvement Toolkit (Mesquite), is a software library which provides many mesh optimisation algorithms to improve the quality of meshes. Mesquite is part of the Interoperable Tools for Advanced Petascale Simulations framework (ITAPS) which aims to provide high quality, reusable, robust and reliable tools to manage the complexities associated with sophisticated simulations [20]. Being part of this framework allows for easy integration with any of the other components of the framework, for example the Mesh Oriented Database (MOAB), [21], which provides a powerful interface for accessing and modifying meshes. This eliminates many of the complexities and inefficiencies associated with working with multiple libraries and greatly reduces the time taken to build a simulation tool. Mesquite is compatible with many different element types, both linear and quadratic, in two and three dimensions. Mesquite's design fully exploits the principles of object oriented programming making it easy to add new components. For this reason it is the ideal platform for the testing and development of new mesh optimisation algorithms.

Quality measures

Quality measures are discussed in depth in Chapter 3, however, a brief overview of the quality measures included in each package is given here. Stellar provides the user with a choice of three quality measures, the volume-length ratio, the minimum sine measure, which are discussed in detail in the following chapter, and the radius ratio which is defined as the ratio of a tetrahedron's circumsphere to to its insphere. Klingner [18] concludes the radius ratio is not very effective and does not recommend its use. Good results are demonstrated using the other two measures.

Mesquite incorporates a large selection of quality measures, for many element types in both two and three dimensions. Different quality measures aim to optimise different criteria depending on the user's requirements. The user can easily add their own quality measure if the existing ones do not adequately meet their needs.

Objective functions

An *objective function* is a means of expressing the quality of an entire mesh or a group of elements as one number, a scalar. The choice of objective function is crucial to obtaining high quality results.

Stellar uses the non-smooth optimisation algorithm developed by Freitag, Jones and Plassmann [22]. Numerical optimisation is the process of maximising or minimising an objective function. When this is applied to a mesh, it is referred to as mesh optimisation and the objective function is the quality of the mesh. A typical objective function is composed of the sum of the quality of every element in the mesh. If such an objective function is optimised, the average element quality will be increased. However, there is no guarantee that the worst element will be improved, it may even be made worse, as one poor element will have little impact on the value of an objective function where the average value dominates. In an extreme case, some elements may even be inverted. Non-smooth optimisation was developed to counter this deficiency of traditional objective functions.

Non-smooth optimisation defines the quality of a mesh, hence its objective function, as the quality of its worst element. There may be more than one element with the same worst quality. Such an algorithm attempts to improve the worst element until it is no longer the worst. The is done by optimising the position of each of its connected nodes, one by one. Figure 2.6 shows a 2D situation with 5 triangular elements, where

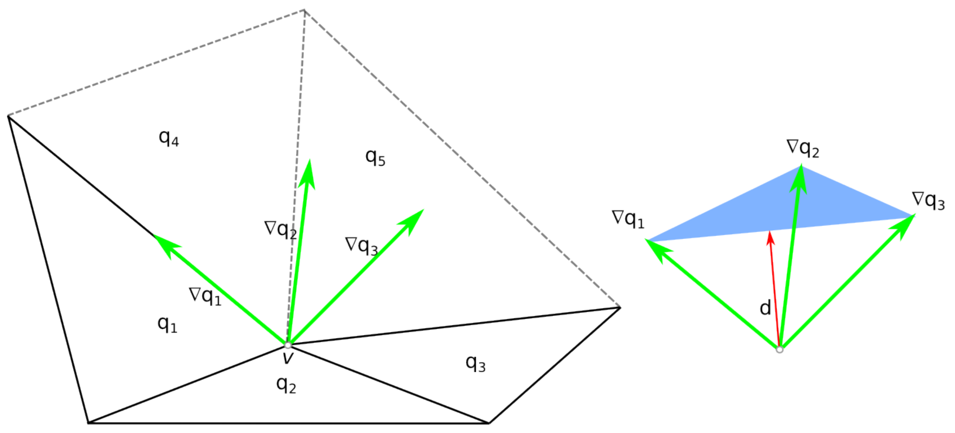


Figure 2.6: Non-Smooth Optimisation

more than one element has the same worst quality connected to the free node. In this case, the quality of elements 1,2 and 3 are equal, $q_1 = q_2 = q_3$. Elements 4 and 5 are of higher quality. The green arrows are vectors illustrating the relative gradient of the quality of elements 1-3 for the free node. A convex hull (shown in blue) is constructed using these vectors. The search direction, **d**, is the line connecting the free node and the closest point on the boundary of the convex hull. At this stage, how the quality of the elements is defined is not important. If the free node lies within the convex hull, it is not possible to improve the elements [22]. As the objective function does not depend on the quality of element's 4 and 5, a method of determining when element's 1,2 or 3 are no longer the worst in the solution space is needed. This is achieved using a Taylor Series expansion to approximate how the quality of every element is affected as the free node is moved along **d**. This gives an estimate of how far the free node may be moved along **d** without reducing the quality of one of elements 1,2 or 3 or the quality of element 4 or 5 becomes the worst in the solution space. This technique has been shown to be very powerful by Freitag [23].

Mesquite provides the user with a choice of objective functions, including non-smooth optimisation. However, objective functions studied in this thesis are restricted to those which are compatible with Newton-type solvers, i.e. twice differentiable functions of nodal positions, as algorithms which may be integrated into the FE solution process are required. Non-smooth optimisation, although proven to be very effective, unfortunately does not fall into this category. This means that Mesquite has only one objective function which is suitable, the L^P objective function:

$$Q = \sum_{i=1}^{n} q_i^p \tag{2.32}$$

where q_i is the quality of the i^{th} element, n is the number of elements and p is a user defined integer power which is greater than or equal to one. This measure is a smooth, twice differentiable function, as required. Extensive experimentation by the author has shown that the use of a higher power than unity ($p > 1$) with the L^P objective function is of no benefit in terms of maximising the quality of the worst element in a mesh. It has also been found by the author that using $p > 1$ is significantly more computationally expensive. This may be due to the additional expense involved in calculating the objective function, its gradient and its Hessian. This is demonstrated in equations 2.33c(a-c) where expressions for the L^p objective function, its gradient and Hessian are given. In the case where $p = 1$, these are significantly simplified. However, when $p > 1$, it is clear that the calculation is much more complicated.

$$Q = \sum_{i=1}^{n} q_i^p \tag{2.33a}$$

$$\nabla Q = \sum_{i=1}^{n} p q^{p-1} \nabla q_i \tag{2.33b}$$

$$\nabla^2 Q = \sum_{i=1}^{n} p(p-1) q^{p-2} \nabla q_i \nabla q_i^T + p q^{p-1} \nabla^2 q \tag{2.33c}$$

Boundary nodes

One of the most difficult aspects of mesh optimisation is the movement of boundary nodes. This topic is covered in detail in Chapter 5. At present it is sufficient to say that Stellar and Mesquite attempt to deal with boundary nodes in very different ways. The movement of nodes on simple boundaries such as straight edges or planar surfaces is straight forward and is explained in Chapter 5. However, the movement of nodes on curved surfaces is much more difficult. Stellar moves boundary nodes by allowing them to deviate from the original surfaces by a user defined tolerance. Stellar makes no attempt to preserve domain volume when moving such nodes, it assumes that small changes are acceptable. Mesquite, on the other hand, cannot move nodes which lie on complex curved surfaces unless geometrical information relating to the continuous domain is supplied. If no such information is available, these boundary nodes are deemed unmovable.

User control and ease of integration with existing projects

A configuration file is supplied with Stellar giving the user basic control over aspects of the improvement process such as the choice of quality measure, which improvement operations are used and element size control. However, the user has very limited control over the optimisation process. To properly control many aspects of the mesh optimisation process, the user must make modifications to the source code. Conversely, Mesquite gives the user full control over the entire mesh optimisation process and when it should terminate. Default options are preselected and wrapper classes are supplied for novice users but more experienced users can easily control the entire process.

Many mesh generation packages (e.g. Cubit [24]) allow for data such as boundary conditions to be added to the mesh as part of the generation process. This data will be lost when a mesh is added to Stellar due to its inability to store such data. While it is possible to overcome this disadvantage via modification of Stellar's source code, Stellar also makes changes to the mesh topology and adds/removes nodes (operations the user has very limited control over) in a manner which is very difficult to track. The use of computational techniques, such as a k-dimensional tree [25], to transfer the data from the original mesh to the optimised mesh is theoretically possible, but in reality it is both demanding in terms of time and user effort.

Mesquite is part of the ITAPS framework and thus is fully compatible with the ITAPS common interface. This makes it simple to integrate efficiently into existing projects built around this framework whilst minimising duplication of data stored in memory. This is ideal if the user wishes to perform mesh optimisation outside of an existing analysis, however, in the case where a user wishes to optimise a mesh whilst simultaneously solving a finite element problem is not so straightforward, as Mesquite uses its own custom data types to store the optimisation data. From an efficiency standpoint, the use of custom data types can be very beneficial, however, from a flexibility and compatibility standpoint, they are problematic. These data types cannot simply be *cast* to standard data types due to the particular intricacies of the chosen storage pattern. This requires the user to extract the required data and manually convert it. Although a competent programmer can comfortably do this, it is both laborious and inefficient.

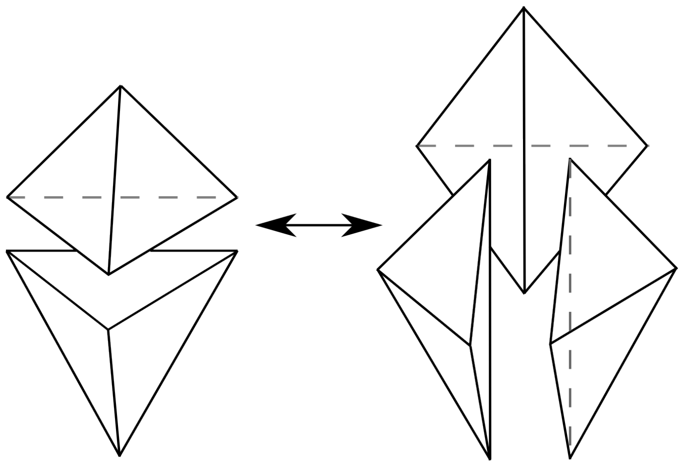


Figure 2.7: 2-3 flip and 3-2 flip, adapted from [26].

Topological transformations

This is arguably Stellar's defining feature. Stellar has very powerful topological transformation functionality and it is the benchmark in this regard. There are two categories of topological transformation: those that preserve the number of nodes in the mesh and those that don't. For example, a 2-3 or 3-2 flip, Figure 2.7, are performed when two tetrahedrons share a face or three tetrahedrons share an edge. The numbers refer to the number of original tetrahedrons and the number of tetrahedrons after the operation. In the case of a 2-3 flip, two tetrahedrons which meet at a face are replaced by three tetrahedrons sharing an edge. If the new configuration results in a better mesh, than it is kept, otherwise the mesh is returned to its former state. Stellar employs many other sophisticated topological transformations which are described in [26] and [18].

Mesquite does not currently support topological transformations although its architecture does allow for the future addition of such features. Mesquite documentation states that this is a future goal but the developers are currently focused on further development of the mesh optimisation algorithms. If one wishes to improve a mesh independent of a finite element solution process, the use of topological transformations may be beneficial once care is taken to limit the change in nodal density and the number of elements. However, if topological transformations are used as part of a finite element solution process, non-linearities are introduced which may severely hinder or even render the entire solution process intractable.

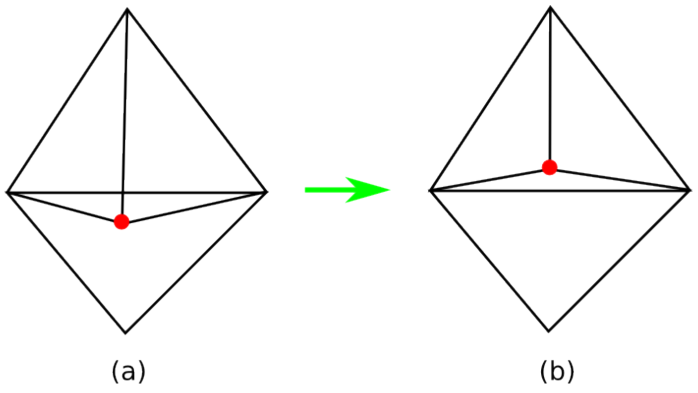


Figure 2.8: Mesh untangling

Mesh untangling

Mesh untangling is required when the initial mesh contains inverted elements, for example the two-dimensional case shown in Figure 2.8(a). Mesquite contains a quality measure which aims to untangle meshes. It must be noted that after untangling a mesh using this quality measure, the mesh may not be optimal, Figure 2.8(b) as this quality measure only aims to untangle the mesh. The resulting mesh should then be optimised using a standard quality measure. In contrast, it is a requirement of Stellar that the initial mesh be untangled.

Summary

Although Stellar and Mesquite have the same end goal, they adopt very different approaches. Stellar is very powerful and produces very high quality meshes. However, it is not at all suited to the needs of a finite element analyst. Nevertheless, as an academic resource to demonstrate the possibility of what may be achieved by mesh improvement operations and as a comparison tool to appraise the quality of a mesh improvement operation it is invaluable. It also worth noting that Stellar is the work of one PhD student, Dr. Bryan Klingner, whereas Mesquite is developed by a team of researchers and developers.

2.2.2 Other mesh optimisation software

CGAL: Computational Geometry Algorithms Library (CGAL) [27] is an open-source computational geometry library developed by a partnership between several universities and industry. This library includes mesh generation and optimisation functionality.

Its optimisation routines are based on Lloyd's algorithm which involves constructing the Voronoi tessellation of the mesh nodes, computing their mass centroids and relocating each node to the centre of its corresponding Voronoi cell [28]. The Delaunay triangulation of the relocated nodes is then calculated.

2.3 Summary

In this section the sources of many errors associated with Finite Element meshes were introduced and discussed. The use of anisotropic elements to reduce interpolation errors was also discussed. In the following chapter, the quality of elements, meshes and and the mesh optimisation process are described. Isotropic element quality measures are focussed on, however, an anisotropic quality measure is proposed. All of the mesh optimisation techniques proposed in this thesis are compatible with anisotropic quality measures.

Chapter 3

The Quality of Finite Elements, Meshes and the Optimisation Process

3.1 Introduction

Finite element meshes represent the discretisation of a domain and are composed of elements, often triangles (2D) or tetrahedra (3D), and are used to form approximations of functions. The quality of an element is a measure of how suitable it is for approximating a particular function. It is shown in Chapter 2 that, of all the factors influencing the quality of the approximation of a function by an element, the internal angles (triangular elements) and dihedral angles (tetrahedral elements) are of most importance. A dihedral angle is the angle formed between two faces of a tetrahedron, Figure 3.1. Examples of elements with dihedral angles ranging from very small to very large are shown in Figure 3.2. Red elements have extreme dihedral angles (either very small or very large) whereas green elements contain no extreme dihedral angles.

3.1.1 Numerical optimisation

Constrained optimisation is the process of maximising or minimising an objective function subject to some constraints. In the context of a mesh, optimisation is the process of finding the best possible position for each of its nodes. In order to optimise a mesh it is necessary to be able to quantify its quality in order to make an absolute and relative assessment. This also allows the ideal element to be mathematically defined. Therefore two things are required: first a means of expressing the quality of an element as a

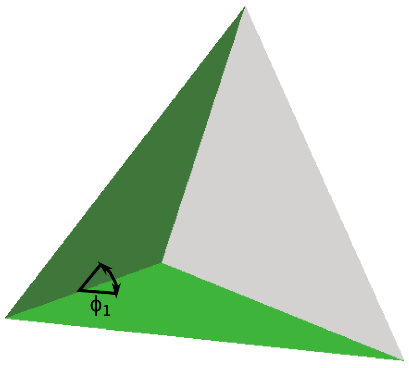


Figure 3.1: There are six dihedral angles in a tetrahedron. These are the angles formed between the faces, for example, ϕ_1.

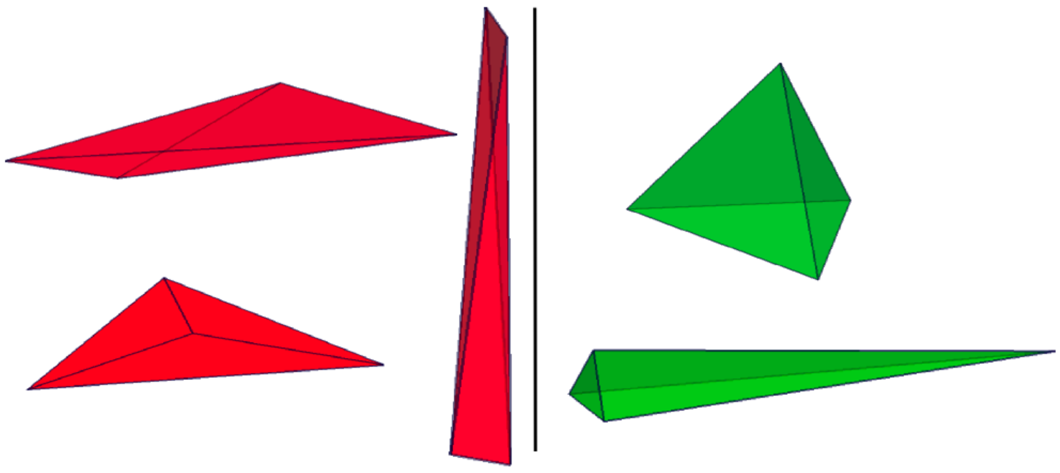


Figure 3.2: Examples of elements with good and poor dihedral angles. The elements on the left all contain either small or large dihedral angles. The elements on the right contain no extreme dihedral angles.

function of its nodal positions and second, a means of expressing the quality of a group of elements using one number - the objective function. It is this objective function which must be maximised or minimised, depending on the nature of the quality measure. In the following two chapters, various quality measures and objective functions will be investigated in order to find the combination which best meets the author's needs.

3.1.2 Problem definition

Mesh optimisation:

In order to choose the best combination of quality measure and objective function, it is necessary to define the aims of the optimisation process:

- To develop a general mesh optimisation framework which can be applied to a wide variety of problems with little user intervention.
- To simplify as much as possible the mesh optimisation process to enable non-expert users to quickly and reliably optimise meshes. For example, mesh generation packages often produce poor quality meshes when presented with complex domains typical of many engineering problems.
- To address the limitations of the current state of the art mesh optimisation packages discussed in the previous chapter, thus making advancements and contributing to the field of mesh optimisation.

Mesh optimisation as part of the FE solution process:

An additional goal of this project is to optimise meshes as part of the FE solution process. In particular, this thesis is concerned with physically non-linear problems where the domain, and therefore also the mesh, evolves during the solution process. This involves coupling the mesh optimisation equations with the physical equations. The equations governing the physical problem and the optimisation process are solved monolithically so as to maintain the quality of meshes during the analysis process, in order to minimise the numerical errors induced by poor meshes. The range of optimisation procedures available is greatly limited in this case due to the requirement that the chosen optimisation strategy be compatible with the solution process for the physical equations. For example, topology modification and node insertion/deletions operations should generally be avoided as these can adversely affect the solution of the physical equations by introducing large non-linearities which the solver may not be able to overcome. Secondly, the chosen quality measure and objective function must both be smooth, twice differentiable functions of the nodal positions in order to be compatible with the Newton solver typically used to solve the non-linear systems of equations governing the physical problem. Although this thesis aims to contribute to the field of mesh optimisation as a separate process, the focus is on the limited range of operations that are compatible with monolithic mesh optimisation. Although mesh optimisation as a separate process is immensely useful, the eventual goal is to apply the knowledge gained in implementing this, thus making it possible to perform mesh optimisation as part of a monolithic solution process.

3.2 Quality measures

Finding a suitable quality measure that provides an accurate estimate of how well an element approximates a function in terms of minimising the discretisation/interpolation errors and stiffness matrix conditioning is challenging and is a very active area of research in itself. The aim of this thesis dictates that any quality measures used must be smooth, twice differentiable and reasonably cheap to calculate so that it may be used as part of a monolithic FE simulation. In the following sections, several quality measures will be examined. To aid visualisation, their 2D versions are studied, although the conclusions drawn are applicable to 3D.

3.2.1 Area-length and volume-length quality measures

The area-length quality measure (AL), and its three-dimensional analogue the volume-length quality measure (VL), is a smooth function of an element's nodal positions, it ranges between -1 and 1, is positive for non-inverted elements and is unity for an equilateral element. The contour plot of the AL quality measure in Figure 3.4a is created by fixing two of a triangle's three nodes and moving the third in the $x - y$ plane. It reaches a maximum when the triangle is equilateral. This function is both smooth and convex in the neighbourhood of its maximum. This means that its matrix of second derivatives, or its Hessian, will be positive definite thus making it ideal for use with an implicit solver.

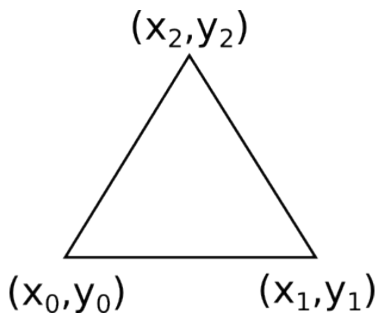


Figure 3.3: Quantities for calculating the element Jacobian matrix.

3.2.2 Ideal weight inverse mean ratio quality measure

Mesquite provides several quality measures which are based on the element Jacobian matrix are these are discussed in detail by Knupp [29]. The Jacobian is defined in equation 3.1 for a 2D triangular element. The variables used in this equation are illustrated in Figure 3.3.

$$\mathbf{J} = \begin{pmatrix} x_0 - x_2 & x_1 - x_2 \\ y_0 - y_2 & y_1 - y_2 \end{pmatrix} \tag{3.1}$$

The Jacobian based quality measure most discussed in the literature is the **Ideal Weight Inverse Mean Ratio** (IMR). It has been shown to be effective by Munson [30] and is recommended in Mesquite's User Guide [17]. For these reasons this measure is studied here in further detail as an example of a Jacobian based measure. The IMR quality measure is defined as:

$$q = \frac{\|\mathbf{AW}^{-1}\|_F^2}{2|det(\mathbf{AW}^{-1})|} \tag{3.2}$$

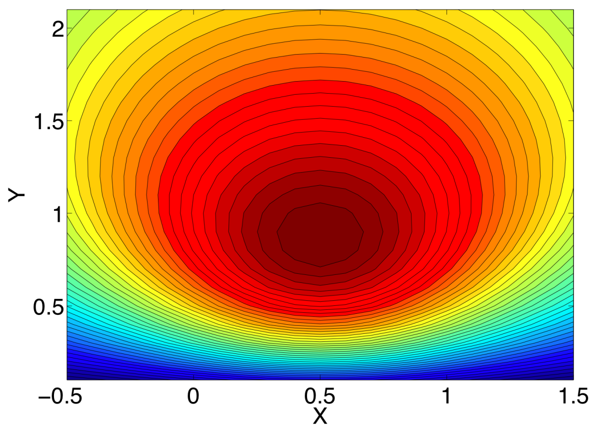
where **W** is the Jacobian matrix of an ideal element, as defined by the user, and **A** is the element Jacobian matrix. The norm used in the numerator of this equation is a Frobenius norm which is defined for a **matrix** as $(\sum_{ij} |a_{ij}|^2)^{\frac{1}{2}}$. In general the ideal element is taken to be equilateral. However, it is worth noting that this is not essential if the application requires that elements have a particular bias. The IMR quality measure varies from unity, when the original element is the same shape as the ideal element, to infinity. In the contour plot of this measure, Figure 3.4g, we can see that this function is very steep in the periphery of the ideal position and flattens rapidly in proximity to the equilateral position. This means that it has very small gradients when its quality is in the range 1-1.75, therefore only very poor quality elements are punished. In a similar manner to the AL, this function is convex in the neighbourhood of its minimum thus, making it ideal for use with implicit solvers.

3.2.3 Sine and Cosine quality measures

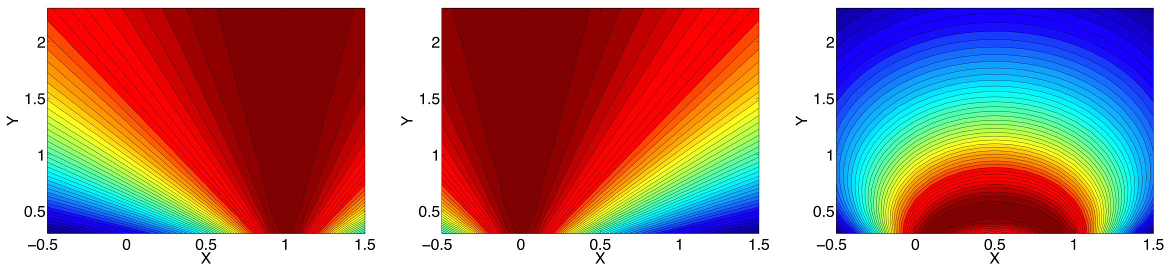
The two measures introduced so far do not directly attack poor internal or dihedral angles, although it has been shown by [18] and [30] that these measures are effective. Several quality measures exist which express the quality of an element in terms of its internal or dihedral angles, for example the sines and cosines of these angles. In these cases, each triangular element has three quality measures and each tetrahedral element has six, corresponding to the number of internal or dihedral angles. This clearly represents a large increase in the computational cost associated with optimising meshes based on such measures. While the quality measures themselves are not hugely expensive to calculate, this is not the case for their first and second derivatives. Both sine and cosine measures vary between -1 and $+1$. With non-inverted elements, only angles in the range $0° - 180°$ occur, so when using a sine measure, only values in the range of 0 to 1 are encountered, whereas with the cosine measure the range is -1 to $+1$. The latter has the advantage that of distinguishing between acute and obtuse angles, whereas the former does not discriminate. This may be useful in some cases, for example if one wishes to eliminate obtuse angles more than acute angles. However, this may be achieved indirectly using a sine measure by applying relative weights to acute/obtuse angles. Despite this disadvantage of the sine measure, it is significantly easier to optimise meshes based on this measure as the sine of an angle is always positive in the range $0 - 180°$, which is the range of interest, thus reserving negative qualities for inverted elements. This means inverted elements are easily identified. The sine of each of a triangle's three angles is plotted in Figure 3.4(b)-(d). The sum of these is shown in Figure 3.4e. Similar to the IMR quality measure, the magnitude of the gradient of this function is relatively small in the neighbourhood of its minimum, in contrast to the AL quality measure which is much larger.

3.2.4 Spire tetrahedra

It is well known that the sum of the angles in a triangle is always 180°. However, no such equivalent exist for tetrahedrons; instead, the sum of the dihedral angles of a

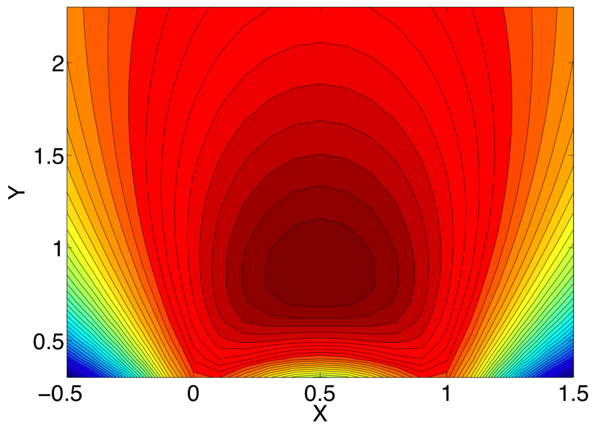


(a) Contour plot of the 2D area-length ratio for a triangle with vertices (0,0), (1,0) and (x,y).

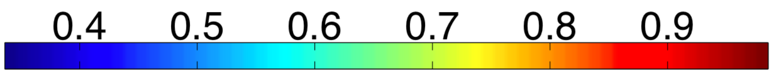


(b) θ_0 (c) θ_1 (d) θ_2

Contour plot of the sine of each of a triangle's three angles, θ_0, θ_1 and θ_2, for a triangle with vertices (0,0), (1,0) and (x,y).

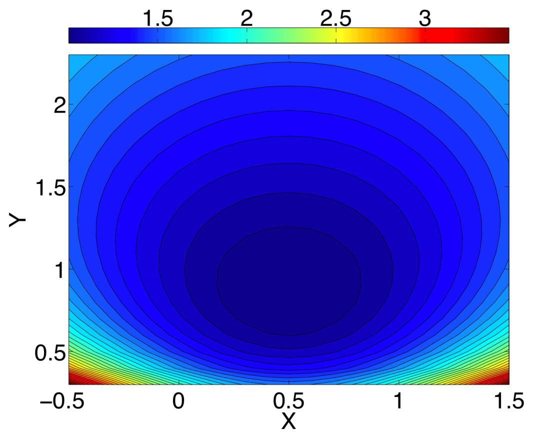


(e) Contour plot of the sum of the three sines for a triangle with vertices (0,0), (1,0) and (x,y). This function reaches a maximum when the triangle is equilateral and has been divided by three to keep it in the range $0 - 1$.



(f) Element quality

Figure 3.4



(g) Contour plot of the ideal weight inverse mean ratio for a triangle with vertices (0,0), (1,0) and (x,y). This function reaches a minimum when the triangle is equilateral.

Figure 3.4: Contour plot of element quality measures

tetrahedron lies in the range $2\pi - 3\pi$. This means that poor quality spire tetrahedra can form in locations where a node is free to move a very large distance without affecting the boundary of the domain or the quality of neighbouring elements. Spire tetrahedra are long and skinny and have very good dihedral angles. An example of a spire tetrahedra is shown in Figure 3.2. The use of a quality measure which only considers the dihedral angles, such as the sine quality measure, encourages their formation whereas the VL quality measure and the IMR quality measure will severely punish them. In certain applications spire tetrahedra do not cause any problems, [18], but generally their presence causes severe problems. In practice, spire tetrahedra are rarely encountered when using the sine measure as the quality of neighbouring elements is usually very poor. They are generally only found when the surface of a mesh is unconstrained and a surface node is associated with only one element. This means that the node may move an arbitrarily long distance resulting in a spire tetrahedron.

3.2.5 The first and second derivatives of quality measures

In order to optimise a mesh using the three quality measures introduced in the previous section, the first and second derivatives (i.e. their gradients and Hessians) of these functions must be derived. Although these may be approximated using finite differences, calculating them analytically is much more accurate, especially in the case of the second derivative, and generally much cheaper. The first derivative of the VL quality measure and the sine quality measure in 3D are given in [18], however it was necessary to derive the remaining first derivatives and all the second derivatives in

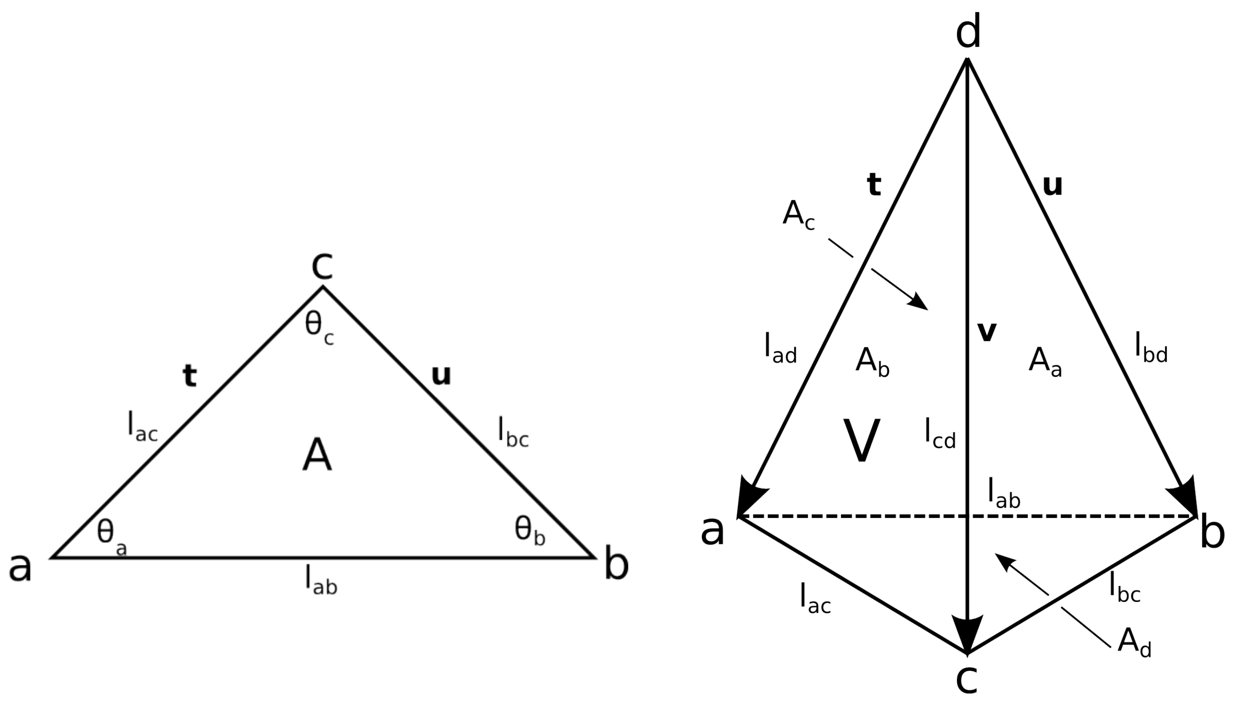


Figure 3.5: Quantities associated with triangles/tetrahedra. a, b, c and d are the nodal coordinates, l_i is the length of edge i, A_i is the area of face i, $\mathbf{t} = a$ - d, $\mathbf{u} = b$ - d, and $\mathbf{v} = c$ - d are the edge vectors, A is the area of a triangular element and V is the volume of a tetrahedral element and $\vec{\mathbf{n}}$(not shown) are the outward pointing normals from each face. Formulas for computing these quantities are given Table 3.1

order to complete this work. Mesquite is adopted as a platform for testing the quality measures as its modular architecture allows for the easy integration of additional quality measures, as described in the previous chapter. The IMR quality measure is included with Mesquite, so it is not necessary to calculate its derivatives.

Two dimensions

The sine of the angle formed between edges i and j can be expressed as:

$$q_{sine}^{i,j} = \frac{2A}{l_i l_j} \tag{3.3}$$

	Triangle		Tetrahedron
$\mathbf{t}$	$a-c$	$\mathbf{t}$	$a-d$
$\mathbf{u}$	$b-c$	$\mathbf{u}$	$b-d$
		$\mathbf{v}$	$c-d$
l_{ac}	$\lvert\mathbf{t}\rvert$	l_{ad}	$\lvert\mathbf{t}\rvert$
l_{bc}	$\lvert\mathbf{u}\rvert$	l_{bd}	$\lvert\mathbf{u}\rvert$
l_{ab}	$\lvert a-b\rvert$	l_{cd}	$\lvert\mathbf{v}\rvert$
		l_{ab}	$\lvert a-b\rvert$
		l_{ac}	$\lvert a-c\rvert$
		l_{bc}	$\lvert b-c\rvert$
l_{rms}	$\sqrt{\frac{1}{3}(l_{ac}^2+l_{bc}^2+l_{ac}^2)}$	l_{rms}	$\sqrt{\frac{1}{6}(l_{ad}^2+l_{bd}^2+l_{cd}^2+l_{ab}^2+l_{ac}^2+l_{bc}^2)}$
		$\vec{\mathbf{n}}_a$	$\mathbf{u}\times\mathbf{v}$
		$\vec{\mathbf{n}}_b$	$\mathbf{v}\times\mathbf{t}$
		$\vec{\mathbf{n}}_c$	$\mathbf{t}\times\mathbf{u}$
		$\vec{\mathbf{n}}_d$	$\mathbf{(u\text{-}v)}\times\mathbf{(t\text{-}v)}$
		A_a	$\frac{\lvert\vec{\mathbf{n}}_a\rvert}{2}$
		A_b	$\frac{\lvert\vec{\mathbf{n}}_b\rvert}{2}$
		A_c	$\frac{\lvert\vec{\mathbf{n}}_c\rvert}{2}$
		A_d	$\frac{\lvert\vec{\mathbf{n}}_d\rvert}{2}$
A	$\frac{\lvert\mathbf{t}\times\mathbf{u}\rvert}{2}$	V	$\frac{\det(\mathbf{tuv})}{6}$

Table 3.1: Formulae to calculate quantities associated with triangles/tetrahedra

Its gradient, where the gradient operator (∇) of a function q is given as $\nabla q = \left(\frac{\partial q}{\partial x}, \frac{\partial q}{\partial y}\right)^T$, can be expressed as:

$$\nabla q_{sine}^{i,j} = 2\left(\frac{\nabla A}{l_i l_j} - \frac{A\nabla l_j}{l_i l_j^2} - \frac{A\nabla l_i}{l_i^2 l_j}\right) \tag{3.4}$$

and its Hessian can be expressed as:

$$\begin{aligned}\nabla^2 q_{sine}^{i,j} = 2\Bigg(&\frac{\nabla^2 A}{l_i l_i} - \frac{\nabla A \nabla l_j^T}{l_i l_j^2} - \frac{\nabla A \nabla l_i^T}{l_i^2 l_j} \\ &- \frac{\nabla A \nabla l_j^T}{l_i l_j^2} - \frac{A\nabla^2 l_j}{l_i l_j^2} + \frac{A\nabla l_j \nabla l_i^T}{l_i^2 l_j^2} + \frac{2A\nabla l_j \nabla l_j^T}{l_i l_j^3} \\ &- \frac{\nabla A \nabla l_i^T}{l_i^2 l_j} - \frac{A\nabla^2 l_i}{l_i^2 l_j} + \frac{A\nabla l_i \nabla l_j^T}{l_i^2 l_j^2} + \frac{2A\nabla l_i \nabla l_i^T}{l_i^3 l_j}\Bigg)\end{aligned} \tag{3.5}$$

where: $\nabla^2 q = \begin{bmatrix} \frac{\partial^2 q}{\partial x^2} & \frac{\partial^2 q}{\partial x \partial y} \\ \frac{\partial^2 q}{\partial x \partial y} & \frac{\partial^2 q}{\partial y^2} \end{bmatrix}$

The Area-Length (AL) quality measure can be expressed as:

$$q_{AL} = \frac{4}{\sqrt{3}} \frac{A}{l_{rms}^2} \tag{3.6}$$

Its gradient can be expressed as:

$$\nabla q_{AL} = \frac{4}{\sqrt{3}}\left(\frac{\nabla A}{l_{rms}^2} - \frac{2A\nabla l_{rms}}{l_{rms}^3}\right) \tag{3.7}$$

and its Hessian can be expressed as:

$$\begin{aligned}\nabla^2 q_{AL} = &\frac{\nabla^2 A}{l_{rms}^2} - \frac{2\nabla A \nabla l_{rms}^T}{l_{rms}^3} \\ &- \frac{2\nabla A \nabla l_{rms}^T}{l_{rms}^3} - \frac{2A\nabla^2 l_{rms}}{l_{rms}^3} + \frac{6A\nabla l_{rms} \nabla l_{rms}^T}{l_{rms}^4}\end{aligned} \tag{3.8}$$

Three Dimensions

The sine of the dihedral angle formed between faces i and j of a tetrahedron can be expressed as:

$$q^{i,j}_{sine} = \frac{3}{2}\frac{Vl_{ij}}{A_iA_j} \tag{3.9}$$

where $l_{i,j}$ is the length of the edge connecting node i and node j. Its gradient can be expressed as:

$$\nabla q^{i,j}_{sine} = \frac{3}{2}\left(\frac{l_{kl}\nabla V}{A_aA_b} + \frac{V\nabla l}{A_aA_b} - \frac{Vl_{kl}\nabla A_b}{A_aA_b^2} - \frac{Vl_{kl}\nabla A_a}{A_a^2A_b}\right) \tag{3.10}$$

and its Hessian can be expressed as:

$$\begin{aligned}\nabla^2 q^{i,j}_{sine} = \frac{3}{2}\Bigg(&\frac{\nabla l_{kl}\nabla V^T}{A_aA_b} + \frac{l_{kl}\nabla^2 V}{A_aA_b} - \frac{l_{kl}\nabla V\nabla^T A_b}{A_aA_b^2} - \frac{l_{kl}\nabla V\nabla^T A_a}{A_a^2A_b}\\ &+\frac{\nabla V^T\nabla l_{kl}}{A_aA_b} + \frac{V\nabla^2 l_{kl}}{A_aA_b} - \frac{V\nabla l_{kl}\nabla^T A_b}{A_aA_b^2} - \frac{V\nabla l_{kl}\nabla^T A_a}{A_a^2A_b}\\ &-\frac{l\nabla V\nabla A_b^T}{A_aA_b^2} - \frac{V\nabla l_{kl}\nabla A_b^T}{A_aA_b^2} - \frac{Vl_{kl}\nabla^2 A_b}{A_aA_b^2} + \frac{Vl_{kl}\nabla A_b\nabla A_a^T}{A_a^2A_b^2} + \frac{2Vl_{kl}\nabla A_b\nabla A_b^T}{A_aA_b^3}\\ &-\frac{l\nabla V\nabla A_a^T}{A_a^2A_b} - \frac{V\nabla l_{kl}\nabla A_a^T}{A_a^2A_b} - \frac{Vl_{kl}\nabla^2 A_a}{A_a^2A_b} + \frac{Vl_{kl}\nabla A_a\nabla A_b^T}{A_a^2A_b^2} + \frac{2Vl_{kl}\nabla A_a\nabla A_a^T}{A_a^3A_b}\Bigg)\end{aligned} \tag{3.11}$$

The Volume-Length (VL) quality measure can be expressed as:

$$q_{VL} = 6\sqrt{2}\frac{V}{l^3_{rms}} \tag{3.12}$$

Its gradient can be expressed as:

$$\nabla q_{VL} = 6\sqrt{2}\left(\frac{\nabla V}{l^3_{rms}} - \frac{3V\nabla l_{rms}}{l^4_{rms}}\right) \tag{3.13}$$

and its Hessian can be expressed as:

$$\begin{aligned}\nabla^2 q_{VL} = 6\sqrt{2}\Bigg(\frac{\nabla^2 V}{l_{rms}^3} - \frac{3\nabla A \nabla l_{rms}^T}{l_{rms}^4} \\ -\frac{3\nabla V \nabla l_{rms}^T}{l_{rms}^4} - \frac{3V\nabla^2 l_{rms}}{l_{rms}^4} \\ +\frac{12V\nabla l_{rms}\nabla l_{rms}^T}{l_{rms}^5}\Bigg)\end{aligned} \tag{3.14}$$

It can be seen that the sine quality measure and its derivatives in both two and three-dimensions is much more expensive to compute than the AL or VL quality measure. It must also be calculated for each angle, thus in two-dimensions it must be calculated three times and in three dimensions, six times. In the following chapter it is determined if the additional computational expense associated with optimising a mesh using the sine measure results in higher quality meshes.

3.2.5.1 Implementation of quality measures using standard FE procedures

Given the focus of this work on improving meshes for the Finite Element Method, it is useful to draw a comparison between the mechanics of large deformations and the process of modifying a mesh in order to improve its quality. This allows mesh optimisation to be implemented using standard finite element procedures that are already used in the context of solid mechanics. This represents a novel contribution of this thesis.

In solid mechanics, the deformation gradient $\mathbf{H}$ is a tensor that quantifies the 2D and 3D shape change of an element, as well as overall rotation. It is defined such that a material line segment $d\boldsymbol{\chi}$ deforms into the new line segment $d\mathbf{X}$, so that:

$$\mathbf{H} = \frac{\partial \mathbf{X}}{\partial \boldsymbol{\chi}} \tag{3.15}$$

This is shown graphically in Figure 3.6. Using standard isoparametric elements (with n nodes per element) to discretise the domain, the geometry of the original elements

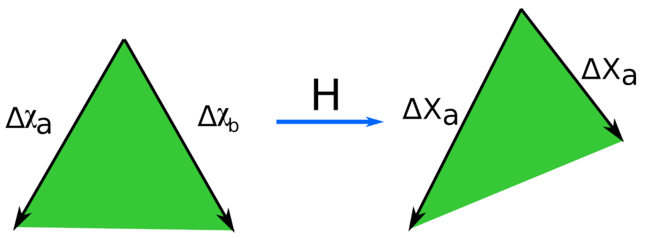


Figure 3.6: The gradient of deformation $\mathbf{H}$, which is used to map from the original mesh to the improved mesh is applied to each of the edge vectors. χ refers to the edge vector of the original element and $\mathbf{X}$ to the edge vector of the improved element.

and the improved elements are defined as:

$$\boldsymbol{\chi} = \sum_{\alpha=1}^{n} N_\alpha \chi_\alpha \tag{3.16a}$$

$$\boldsymbol{X} = \sum_{\alpha=1}^{n} N_\alpha X_\alpha \tag{3.16b}$$

where χ_a are the nodal coordinates of the original element, X_a the nodal coordinates of the improved element and N_a are the standard finite element shape functions which satisfy the condition $\sum_{\alpha=1}^{n} N_a = 1$. The shape functions are typically defined with respect to a local coordinate system (ξ). For example, for a three noded triangular element, N_a are linear:

$$N_1 = 1 - \xi_1 - \xi_2$$
$$N_2 = \xi_1$$
$$N_3 = \xi_2$$

Given 3.16b, the deformation gradient is therefore:

$$\mathbf{H} = \frac{\partial \mathbf{X}}{\partial \boldsymbol{\chi}} = \sum_{\alpha=1}^{n} X_\alpha \frac{\partial N_\alpha}{\partial \boldsymbol{\chi}} \tag{3.18}$$

or more conveniently:

$$\mathbf{H}_{i,J} = \sum_{\alpha=1}^{n} X_{\alpha,i} \frac{\partial N_\alpha}{\partial \chi_J} \tag{3.19}$$

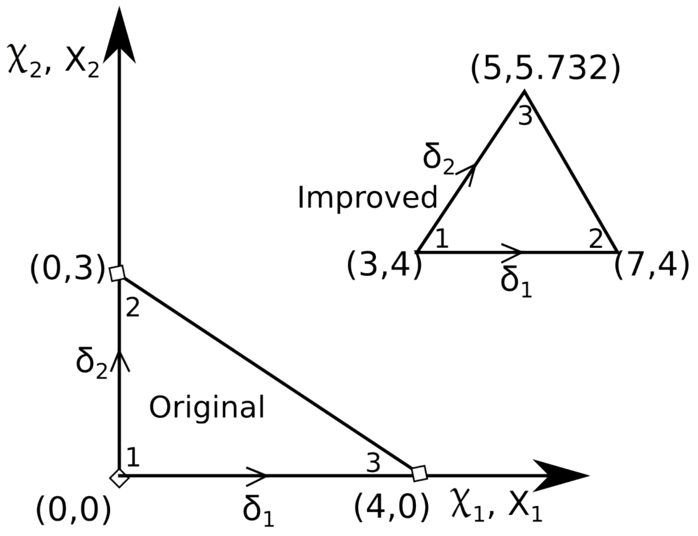


Figure 3.7: Initial and improved element

Consider the following two-dimensional example where **H** is calculated.

The derivative of the shape functions, N_α, with respect to the initial coordinates, $\boldsymbol{\chi}$, may be expressed as:

$$\frac{\partial N_a}{\partial \boldsymbol{\chi}} = \left(\frac{\partial \boldsymbol{\chi}}{\partial \boldsymbol{\xi}}\right)^{-T} \frac{\partial N_a}{\partial \boldsymbol{\xi}} \tag{3.20}$$

Given 3.16a,

$$\frac{\partial \boldsymbol{\chi}}{\partial \boldsymbol{\xi}} = \begin{bmatrix} \frac{\partial \chi_1}{\partial \xi_1} & \frac{\partial \chi_1}{\partial \xi_2} \\ \frac{\partial \chi_2}{\partial \xi_1} & \frac{\partial \chi_2}{\partial \xi_2} \end{bmatrix}; \quad \frac{\partial \chi_I}{\partial \xi_a} = \sum_{\alpha=1}^{n} \chi_{\alpha,I} \frac{\partial N_\alpha}{\partial \xi_a} \tag{3.21}$$

For this example,

$$\begin{matrix} \frac{\partial N_1}{\partial \xi_1} = -1 & \frac{\partial N_2}{\partial \xi_1} = 1 & \frac{\partial N_3}{\partial \xi_1} = 0 \\ \frac{\partial N_1}{\partial \xi_2} = -1 & \frac{\partial N_2}{\partial \xi_2} = 0 & \frac{\partial N_3}{\partial \xi_2} = 1 \end{matrix} \tag{3.22}$$

Given the problem shown in Figure 3.7,

$$\frac{\partial \boldsymbol{\chi}}{\partial \boldsymbol{\xi}} = \begin{bmatrix} 0\times(-1)+4\times 1+0\times 0 & 0\times(-1)+4\times 0+0\times 1 \\ 0\times(-1)+0\times 1+3\times 0 & 0\times(-1)+0\times 0+3\times 1 \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 0 \\ 0 & 3 \end{bmatrix} \tag{3.23}$$

From 3.20,

$$\begin{aligned} \frac{\partial N_1}{\partial \boldsymbol{\chi}} &= \frac{1}{12}\begin{bmatrix} 3 & 0 \\ 0 & 4 \end{bmatrix}\begin{pmatrix} -1 \\ -1 \end{pmatrix} &= \frac{-1}{12}\begin{pmatrix} 3 \\ 4 \end{pmatrix} \\ \frac{\partial N_2}{\partial \boldsymbol{\chi}} &= \frac{1}{12}\begin{bmatrix} 3 & 0 \\ 0 & 4 \end{bmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} &= \frac{1}{12}\begin{pmatrix} 3 \\ 0 \end{pmatrix} \\ \frac{\partial N_3}{\partial \boldsymbol{\chi}} &= \frac{1}{12}\begin{bmatrix} 3 & 0 \\ 0 & 4 \end{bmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} &= \frac{1}{12}\begin{pmatrix} 0 \\ 4 \end{pmatrix} \end{aligned} \tag{3.24}$$

Therefore, from 3.19 and 3.24,

$$\mathbf{H} = \frac{1}{12}\begin{bmatrix} 3\times(-3)+7\times 3+5\times 0 & 3\times(-4)+7\times 0+5\times 4 \\ 4\times(-3)+4\times 3+5.752\times 0 & 4\times(-4)+4\times 0+5.732\times 4 \end{bmatrix}$$

$$= \frac{1}{12}\begin{bmatrix} 12 & 8 \\ 0 & 6.928 \end{bmatrix} = \begin{bmatrix} 1 & \frac{2}{3} \\ 0 & 0.5773 \end{bmatrix} \tag{3.25}$$

This second order tensor can now be used to transform a vector from the initial mesh to the improved mesh.

3.2.5.2 Expressing quality measures as a function of the gradient of deformation

In this section, expressions are developed for the AL and VL quality measures and their derivatives as functions of the gradient of deformation. Consider the AL quality measure, recalling equation 3.6:

$$q = \frac{4}{\sqrt{3}} \frac{A}{l_{rms}^2}$$

The area of the improved element may be expressed with respect to the area of the original element using the gradient of deformation tensor:

$$A = A_0 \det(\mathbf{H}) \tag{3.26}$$

Therefore, by combining equations 3.6 and 3.26, the AL quality measure can be expressed as:

$$q = \frac{4}{\sqrt{3}} \frac{A_0 \det(\mathbf{H})}{l_{rms}^2} \tag{3.27}$$

The first and second derivatives of this equation are now required so that a mesh may be optimised using this quality measure. Therefore, the derivatives of $\det(\mathbf{H})$ and l_{rms}^2 with respect to the gradient of deformation must be developed.

The following derivation is adapted from Bonet and Wood [31]. Here, the rate of change of a function, f, in the direction of a vector $\mathbf{v}$, is determined. This is the so-called directional derivative, denoted as $\mathcal{D}f[\mathbf{v}]$.

$$\begin{aligned} \mathcal{D}det(\mathbf{H})[\delta\mathbf{X}] &= \frac{\partial}{\partial\epsilon}\Big|_{\epsilon=0} det\left(\mathbf{H} + \epsilon\frac{\partial\delta\mathbf{X}}{\partial\boldsymbol{\chi}}\right) \\ &= \frac{\partial}{\partial\epsilon}\Big|_{\epsilon=0} \det\left[\mathbf{H}\left(\mathbf{I} + \epsilon\mathbf{H}^{-1}\frac{\partial\delta\mathbf{X}}{\partial\boldsymbol{\chi}}\right)\right] \\ &= \det(\mathbf{H})\frac{\partial}{\partial\epsilon}\Big|_{\epsilon=0} \det\left(\mathbf{I} + \epsilon\mathbf{H}^{-1}\frac{\partial\delta\mathbf{X}}{\partial\boldsymbol{\chi}}\right) \end{aligned} \tag{3.28}$$

where $\mathbf{I}$ is the identity tensor, $\dfrac{\partial\delta\mathbf{X}}{\partial\boldsymbol{\chi}}$ is the derivative of Equation 3.15 and ϵ represents an infinitely small change in the function [31]. The characteristic polynomial of a 3×3

matrix $\mathbf{B}$ with eigenvalues λ_1^B, λ_2^B and λ_3^B is

$$\det(\mathbf{B} - \lambda\mathbf{I}) = (\lambda_1^B - \lambda)(\lambda_2^B - \lambda)(\lambda_3^B - \lambda) \tag{3.29}$$

Taking λ as -1 and $\mathbf{B} = \epsilon\mathbf{H}^{-1}\frac{\partial\delta\mathbf{X}}{\partial\boldsymbol{\chi}}$ gives

$$\mathcal{D}\det(\mathbf{H})[\delta\mathbf{X}] = \det(\mathbf{H})\frac{\partial}{\partial\epsilon}\bigg|_{\epsilon=0}(1+\epsilon\lambda_1)(1+\epsilon\lambda_2)(1+\epsilon\lambda_3) \tag{3.30}$$

where λ_1, λ_2 and λ_2 are the eigenvalues of $\epsilon\mathbf{H}^{-1}\delta\mathbf{X}$. Using the product rule gives

$$\mathcal{D}\det(\mathbf{H})[\delta\mathbf{X}] = \det(\mathbf{H})(\lambda_1 + \lambda_2 + \lambda_3) \tag{3.31}$$

As the trace of a matrix is equal to the sum of its eigenvalues,

$$\begin{aligned}\mathcal{D}\det(\mathbf{H})[\delta\mathbf{X}] &= \det(\mathbf{H})\mathrm{tr}\left(\mathbf{H}^{\text{-}1}\frac{\partial\delta\mathbf{X}}{\partial\boldsymbol{\chi}}\right)\\ &= \det(\mathbf{H})\left(\mathbf{H}^{\text{-T}} : \frac{\partial\delta\mathbf{X}}{\partial\boldsymbol{\chi}}\right)\end{aligned} \tag{3.32}$$

The double contraction operator (:) introduced above is defined as

$$\mathbf{A} : \mathbf{B} = \mathrm{tr}(\mathbf{A}^{\mathrm{T}}\mathbf{B})$$

In order to derive the derivative of l_{rms}^2 with respect to the gradient of deformation, it must first be possible to express it as a function of $\mathbf{H}$. In two-dimensions, l_{rms} may be expressed as:

$$l_{\mathrm{rms}} = \sqrt{\frac{1}{3}\sum_{i=1}^{3} l_i^2} \tag{3.33}$$

As shown in Table 3.1, l_i is simply the norm of the edge vector. A vector norm may also be expressed using the square root of the dot product of the vector with itself. Using the same notation as in Figure 3.6, the length of edge $\Delta\mathbf{X}_i$ is

$$l_i = \sqrt{\Delta\mathbf{X}_i^{\mathrm{T}}\Delta\mathbf{X}_i} \tag{3.34}$$

From equation 3.15, $\Delta\mathbf{X}_i = \mathbf{H}\Delta\chi$. Therefore, l_i may be expressed as:

$$l_i = \sqrt{\Delta\boldsymbol{\chi}_i^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{H}\Delta\boldsymbol{\chi}_i} \tag{3.35}$$

and l_i^2 as

$$l_i^2 = \Delta\boldsymbol{\chi}_i^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{H}\Delta\boldsymbol{\chi}_i \tag{3.36}$$

Using the standard product rule of differentiation the following expression is developed:

$$\mathcal{D}(\mathbf{H}^{\mathrm{T}}\mathbf{H})(\delta\mathbf{X}) = \left(\frac{\partial\delta\mathbf{X}}{\partial\boldsymbol{\chi}}\right)^{\mathrm{T}}\mathbf{H} + \mathbf{H}^{\mathrm{T}}\left(\frac{\partial\delta\mathbf{X}}{\partial\boldsymbol{\chi}}\right) \tag{3.37}$$

Using this expression, a derivative for l_i can now be developed.

$$\begin{aligned}
\mathcal{D}(l_i)[\delta X] &= \frac{1}{2l_i}\left(\Delta\boldsymbol{\chi}_i^{\mathrm{T}}\left(\frac{\partial\delta\mathbf{X}}{\partial\boldsymbol{\chi}}^{\mathrm{T}}\mathbf{H} + \mathbf{H}^{\mathrm{T}}\frac{\partial\delta\mathbf{X}}{\partial\boldsymbol{\chi}}\right)\Delta\boldsymbol{\chi}_i\right)\\
&= \frac{1}{2l_i}\left(\Delta\boldsymbol{\chi}_i^{\mathrm{T}}\frac{\partial\delta\mathbf{X}}{\partial\boldsymbol{\chi}}^{\mathrm{T}}\mathbf{H}\Delta\boldsymbol{\chi}_i + \Delta\boldsymbol{\chi}_i^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\frac{\partial\delta\mathbf{X}}{\partial\boldsymbol{\chi}}\Delta\boldsymbol{\chi}_i\right)\\
&= \frac{1}{2l_i}\left(\Delta\boldsymbol{\chi}_i^{\mathrm{T}}\frac{\partial\delta\mathbf{X}}{\partial\boldsymbol{\chi}}^{\mathrm{T}}\Delta\mathbf{X}_i + \Delta\mathbf{X}_i^{\mathrm{T}}\frac{\partial\delta\mathbf{X}}{\partial\boldsymbol{\chi}}\Delta\boldsymbol{\chi}_i\right)\\
&= \frac{1}{2l_i}\left(\left(\frac{\partial\delta\mathbf{X}}{\partial\boldsymbol{\chi}}\right)^{\mathrm{T}} : \Delta\boldsymbol{\chi}_i\Delta\mathbf{X}_i^{\mathrm{T}} + \Delta\mathbf{X}_i\Delta\boldsymbol{\chi}_i^{\mathrm{T}} : \left(\frac{\partial\delta\mathbf{X}}{\partial\boldsymbol{\chi}}\right)\right)\\
&= \frac{1}{l_i}\Delta\mathbf{X}_i\Delta\boldsymbol{\chi}_i^{\mathrm{T}} : \left(\frac{\partial\delta\mathbf{X}}{\partial\boldsymbol{\chi}}\right)
\end{aligned} \tag{3.38}$$

Therefore, the derivative of l_i^2 may be expressed as:

$$\mathcal{D}(l_i^2)[\delta X] = 2\Delta\mathbf{X}_i\Delta\boldsymbol{\chi}_i^{\mathrm{T}} : \frac{\partial\delta\mathbf{X}}{\partial\boldsymbol{\chi}}^{\mathrm{T}} \tag{3.39}$$

Using equation 3.39, an expression for the derivative of l_{rms}^2 may be developed.

$$l_{\mathrm{rms}} = \sqrt{\frac{1}{3}\sum_{i=1}^{3} l_i^2} \tag{3.40}$$

$$\mathcal{D}(l_{rms})[\delta X] = \frac{1}{3l_{rms}} \sum_{i=1}^{3} \Delta \mathbf{X}_i \Delta \boldsymbol{\chi}_i^{\mathrm{T}} : \frac{\partial \delta \mathbf{X}}{\partial \boldsymbol{\chi}} \tag{3.41}$$

$$\mathcal{D}(l_{rms}^2)[\delta X] = \frac{2}{3} \sum_{i=1}^{3} \Delta \mathbf{X}_i \Delta \boldsymbol{\chi}_i^{\mathrm{T}} : \frac{\partial \delta \mathbf{X}}{\partial \boldsymbol{\chi}} \tag{3.42}$$

Using equations 3.32 and 3.42 and the standard quotient rule of differentiation, an expression for the first directional derivative of the AL quality measure can now be developed:

$$\mathcal{D}(q_{AL})[\delta X] = \frac{4}{\sqrt{3}} \frac{A_0 \det(\mathbf{H})}{l_{rms}^2} \left(\mathbf{H}^{\text{-T}} - \frac{2}{3l_{rms}^2} \sum_{i=1}^{3} \Delta \mathbf{X}_i \Delta \boldsymbol{\chi}_i^{\mathrm{T}} \right) : \frac{\partial \delta \mathbf{X}}{\partial \boldsymbol{\chi}} \tag{3.43}$$

and the first derivative of the VL quality measure may be expressed as:

$$\mathcal{D}(q_{VL})[\delta X] = 6\sqrt{2} \frac{V_0 \det(\mathbf{H})}{l_{rms}^3} \left(\mathbf{H}^{\text{-T}} - \frac{1}{2l_{rms}^2} \sum_{i=1}^{6} \Delta \mathbf{X}_i \Delta \boldsymbol{\chi}_i^{\mathrm{T}} \right) : \frac{\partial \delta \mathbf{X}}{\partial \boldsymbol{\chi}} \tag{3.44}$$

In order to derive the second directional derivative of the AL quality measure, a derivative of $\mathbf{H}^{\text{-T}}$ must be found. This is achieved using the product rule and noting that the directional derivative of the identity tensor is the null tensor and starting from the expression below,

$$\begin{aligned} \mathcal{D}(\mathbf{H}^{\text{-T}}\mathbf{H}^{\mathrm{T}})[\delta \mathbf{X}] &= \mathcal{D}(\mathbf{I})\,[\delta \mathbf{X}] = 0 \\ \mathcal{D}(\mathbf{H}^{\text{-T}})[\delta \mathbf{X}]\mathbf{H}^{\mathrm{T}} + \mathbf{H}^{\text{-T}}\mathcal{D}(\mathbf{H}^{\mathrm{T}})[\delta \mathbf{X}] &= 0 \\ \mathcal{D}(\mathbf{H}^{\text{-T}})[\delta \mathbf{X}]\mathbf{H}^{\mathrm{T}}\mathbf{H}^{\text{-T}} &= -\mathbf{H}^{\text{-T}}\mathcal{D}(\mathbf{H}^{\mathrm{T}})[\delta \mathbf{X}]\mathbf{H}^{\text{-T}} \\ \mathcal{D}(\mathbf{H}^{\text{-T}})[\delta \mathbf{X}] &= -\,\mathbf{H}^{\text{-T}} \left(\frac{\partial \delta \mathbf{X}}{\partial \boldsymbol{\chi}} \right)^{\mathrm{T}} \mathbf{H}^{\text{-T}} \end{aligned} \tag{3.45}$$

The following expressions, which are developed in Appendix A, are required for the derivation of the second derivative of the AL and VL quality measures:

$$\mathbf{H}^{-T} \otimes \mathbf{H}^{-T} : \frac{\partial \delta \mathbf{X}}{\partial \boldsymbol{\chi}} - \mathbf{H}^{-T} \frac{\partial \delta \mathbf{X}}{\partial \boldsymbol{\chi}}^{T} \mathbf{H}^{-T} = \mathcal{F}(\mathbf{H}^{-T}) : \frac{\partial \delta \mathbf{X}}{\partial \boldsymbol{\chi}} \tag{3.46}$$

$$\frac{\partial \delta \mathbf{X}}{\partial \boldsymbol{\chi}} \mathbf{A} = \mathcal{T}(\mathbf{A}) : \frac{\partial \delta \mathbf{X}}{\partial \boldsymbol{\chi}} \tag{3.47}$$

where both $\mathcal{F}$ and $\mathcal{T}$ are fourth order tensors, where $\otimes$ refers to the tensor outer product defined as $\mathbf{A} \otimes \mathbf{B} = \mathbf{A}_{ij}\mathbf{B}_{kl}$. Using the following tensor invariant from Bonet and Wood [31],

$$\left(\mathbf{H} : \frac{\partial \delta \mathbf{X}}{\partial \boldsymbol{\chi}}\right) \mathbf{H} = \mathbf{H} \otimes \mathbf{H} : \frac{\partial \delta \mathbf{X}}{\partial \boldsymbol{\chi}} \tag{3.48}$$

the product and quotient rules of differentiation and Equations 3.45, 3.46 and 3.47, the second derivative of the AL quality measure may be expressed as:

$$\begin{aligned}\mathcal{D}^2(q_{AL})[\delta X] = \frac{\partial \delta \mathbf{X}}{\partial \boldsymbol{\chi}}^T : \frac{4}{\sqrt{3}} \frac{A_0 \det(\mathbf{H})}{l_{rms}^2} & \left[\mathcal{F}(\mathbf{H}^{-T}) - \frac{2}{3l_{rms}^2} \mathbf{H}^{-T} \otimes \left(\sum_{i=1}^{3} \Delta \mathbf{X}_i \Delta \boldsymbol{\chi}_i^{\text{-T}} \right) \right. \\ & - \frac{2}{3l_{rms}^2} \left(\sum_{i=1}^{3} \Delta \mathbf{X}_i \Delta \boldsymbol{\chi}_i^{\text{-T}} \right) \otimes \mathbf{H}^{-T} \\ & - \frac{2}{3l_{rms}^2} \mathcal{T} \left(\sum_{i=1}^{3} \Delta \boldsymbol{\chi}_i \Delta \boldsymbol{\chi}_i^{\text{-T}} \right) \\ & \left. + \frac{8}{9l_{rms}^4} \left(\sum_{i=1}^{3} \Delta \mathbf{X}_i \Delta \boldsymbol{\chi}_i^{\text{-T}} \right) \otimes \left(\sum_{i=1}^{3} \Delta \mathbf{X}_i \Delta \boldsymbol{\chi}_i^{\text{-T}} \right) \right] \\ & : \frac{\partial \delta \mathbf{X}}{\partial \boldsymbol{\chi}} \end{aligned} \tag{3.49}$$

and the second derivative of the VL quality measure may be expressed as:

$$\begin{aligned}\mathcal{D}^2(q_{VL})[\delta X] = \frac{\partial \delta \mathbf{X}}{\partial \boldsymbol{\chi}}^T : 6\sqrt{2} \frac{V_0 \det(\mathbf{H})}{l_{rms}^3} & \left[\mathcal{F}(\mathbf{H}^{-T}) - \frac{1}{2l_{rms}^2} \mathbf{H}^{-T} \otimes \left(\sum_{i=1}^{6} \Delta \mathbf{X}_i \Delta \boldsymbol{\chi}_i^{\text{-T}} \right) \right. \\ & - \frac{1}{2l_{rms}^2} \left(\sum_{i=1}^{6} \Delta \mathbf{X}_i \Delta \boldsymbol{\chi}_i^{\text{-T}} \right) \otimes \mathbf{H}^{-T} \\ & - \frac{1}{2l_{rms}^2} \mathcal{T} \left(\sum_{i=1}^{6} \Delta \boldsymbol{\chi}_i \Delta \boldsymbol{\chi}_i^{\text{-T}} \right) \\ & \left. + \frac{5}{12l_{rms}^4} \left(\sum_{i=1}^{6} \Delta \mathbf{X}_i \Delta \boldsymbol{\chi}_i^{\text{-T}} \right) \otimes \left(\sum_{i=1}^{6} \Delta \mathbf{X}_i \Delta \boldsymbol{\chi}_i^{\text{-T}} \right) \right] \\ & : \frac{\partial \delta \mathbf{X}}{\partial \boldsymbol{\chi}} \end{aligned} \tag{3.50}$$

These expressions for the first and second derivatives of the AL and VL quality measures with respect to the gradient of deformation tensor are used with the objective functions introduced in the next section to construct the system of equations required for mesh optimisation.

3.2.6 Anisotropic quality measures

The quality measures described in the previous sections are not suitable for use with anisotropic meshes. This is because the ideal elements are equilateral triangles or tetrahedra in the case of all of the quality measures except for the ideal weight inverse mean ratio, in which the ideal element is defined by the user. Although the user may define the shape and size of the ideal element which may even be a function of the position in the mesh, this measure is rotation invariant. This means that while it is possible to optimise meshes using an anisotropic variant of this measure, the optimised elements may not be correctly oriented.

The gradient of deformation tensor, introduced in Section 3.2.5.2, is not rotation invariant meaning that an anisotropic quality measure could be developed using this tensor. For example, the ideal weight inverse mean ratio could be modified to incorporate this tensor, where $\mathbf{AW}^{-1}$ is replaced by $\mathbf{H}$:

$$q = \frac{||\mathbf{H}||_F^2}{2|det(\mathbf{H})|} \tag{3.51}$$

where $\mathbf{H}$ is the gradient of deformation calculated between the current element and the ideal element at a particular position.

Barlow [32] develops measures of the deformation of elements based on the local coordinate system. These are expressed in terms of rigid displacement, aspect ratio, skew and in the case of quadratic and cubic elements, edge and face curvature. He concludes that the suggested measures are suitable for evaluating element performance based on convergence requirements but not for predicting the behaviour of an element in a given stress field. However, a method of enforcing orientation dependence of distortion measures, and thus quality measures, is suggested. This involves measuring the geometry in a non-orthogonal coordinate system and scaling those coordinates.

3.3 Mesh quality objective functions

Mesh quality optimisation requires an objective function which combines the qualities of a group of elements into a scalar value. For example, one could express the quality,

Q, of a mesh as the sum of the qualities of every element as:

$$Q = \sum_{i=1}^{k} q_i \tag{3.52}$$

where k is the number of elements in the mesh. Such an objective function is referred to as a L^1 norm. An L^2 norm is given as:

$$Q = \sum_{i=1}^{k} q_i^2 \tag{3.53}$$

As stated in Section 2.2.1, simple objective functions such as those described above are very good at improving average element quality, however the worst elements may suffer as a consequence as the improvement of many average quality elements has a greater impact on the value of the objective function than the improvement of several extreme elements. Such an objective functions may even invert some elements, as one negative number may not sufficiently influence the objective function. However as stated in Section 2.1.5, one poor element may render a Finite Element analysis intractable. Therefore, it is desirable to use an objective function that targets the quality of the worst element, rather than the average.

Given this discussion, at first glance, an infinity norm seems like an ideal objective function. This is where the quality of a group of elements is expressed as the quality of the worst element. In this case, any attempt to optimise the mesh will improve the worst element. However, nodes are shared between elements. So if a node is moved to increase the quality of one element, the quality of adjoining elements may be adversely affected. As the infinity norm contains no information about the adjoining elements' quality, there is no way of knowing when the element being improved is no longer the worst element in the mesh. Therefore, such an objective function is described as being non-smooth. A non-smooth optimisation algorithm was developed by [22], which enabled the improvement of the worst element in a mesh and is described in Section 2.2.1.

A genuinely smooth objective function which both contains information on the quality of every element in the mesh and penalises the worst elements to such an extent that the improvement process focuses on these elements should, in theory, yield better results in a shorter analysis time since there is no requirement to approximate when the quality of the worst element changes. Also, a much wider range of numerical optimisation techniques may be used, since such an objective function is smooth. In the following section an objective function that is both smooth and focuses on extreme elements is

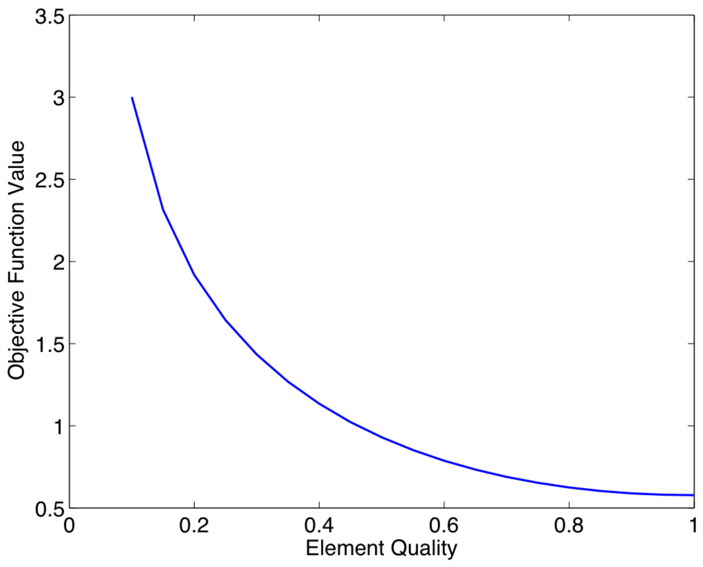


Figure 3.8: Plot of the Log-Barrier function

proposed.

3.3.1 Penalising the worst element

In order to address the deficiencies of existing objective functions, a new objective function is proposed which involves using a log-barrier to penalise the worst elements in a mesh. This is adapted from Scherer et al [33] where a barrier function is used to ensure that elements do not become inverted. A log-barrier function expresses the quality of every element in the solution space as a function of the worst element. This function also has an invertibility guarantee as the quality of an element cannot be less than the barrier as its quality would be undefined. The log-barrier objective function is calculated as follows:

$$Q = \sum_{i=1}^{k} \left(\frac{q_i^2}{2(1-\gamma)} - log(q_i - \gamma) \right) \tag{3.54}$$

where q_i is the quality of element of i and γ is a constant in the range $0-0.99$ times the quality of the worst element, q_{min}, in the mesh. This is shown graphically in Figure 3.8, where the objective function contribution of an element is plotted against its quality.

3.3.2 Optimising the objective function

The next step in the optimisation process is to assemble the global matrix of second derivatives, (**S**), and the global vector of first derivatives (**f**), of the quality measure. These are analogous to the FE stiffness matrix and force vector. These are assembled from the element gradients and Hessians, calculated in the previous sections. These are vector and matrix quantities with **f** having n entries and **S** having n^2 entries where n is the number of degrees of freedom in the system. In two dimensions, n is equal to twice the number of nodes in the mesh as each node is free to move in the x and y direction. In three dimensions, n is three times the number of mesh nodes.

In general, **S** and **f** are assembled from the first and second derivatives of the operation performed to calculate the objective function. For example, **S** and **f** for the $|L_1|$ objective function are assembled as follows:

$$\mathbf{f} = \operatorname*{\mathsf{A}}_{i=1}^{k} \nabla q_i \tag{3.55}$$

$$\mathbf{S} = \operatorname*{\mathsf{A}}_{i=1}^{k} \nabla^2 q_i \tag{3.56}$$

where A is the standard assembly operator and k is the number of elements in the mesh.

For the log-barrier objective function, these are assembled as follows:

$$\mathbf{f} = \operatorname*{\mathsf{A}}_{k=1}^{n} \left(\left[\frac{q_k}{1-\gamma} - \frac{1}{q_k - \gamma} \right] \nabla q_k \right) \tag{3.57}$$

$$\mathbf{S} = \operatorname*{\mathsf{A}}_{k=1}^{n} \left(\nabla q_k \left[\frac{1}{1-\gamma} - \frac{1}{(q_k - \gamma)^2} \right] \nabla q_k^T + \left[\frac{q_k}{1-\gamma} - \frac{1}{q_k - \gamma} \right] \nabla^2 q_k \right) \tag{3.58}$$

The assembled system of equations takes the following form:

$$\mathbf{S}\delta\mathbf{X} = -\mathbf{f} \tag{3.59}$$

where $\delta\mathbf{X}$ is the vector of unknown nodal displacements. This vector is then solved for using a Newton-Raphson iterative solver.

3.3.3 Termination of the optimisation process

As previously stated, the goal of the optimisation approach is to increase the quality of the worst elements in a mesh to an acceptable quality. There is very little to be gained by greatly increasing the quality of the average element as the problems associated with poor meshes are specifically related to the worst elements. Numerical optimisation is generally terminated when the solution is deemed to have converged, convergence is commonly deemed to have occurred when the residual is below a user defined value, generally in the range $10^{-3} - 10^{-12}$. Careful examination of Figure 3.9 provides some revealing insight into how the mesh optimisation process works. This graph was created by examining the optimisation process of the two-dimensional square mesh with 20,258 elements, "Square" shown in Figure 3.10. The mesh is examined at each iteration of the optimisation process, with the range of angles shown on the left y axis and the magnitude of the residual shown on the right logarithmic y-axis. The time taken to reach each iteration is shown on a non-linear bar on top. After one iteration the residual has been reduced by 90% and the range of angles has drastically reduced. After the second iteration, the range of angles has reached its final value and the residual is now less than 1% of its initial value. This is achieved after $5.08s$. Using standard convergence criteria, the optimisation process does not terminate for a further $6.91s$. The final mesh is negligibly better than the mesh after two iterations. Therefore, it can be concluded from this that there is little benefit in continuing the optimisation process past this stage, thus the optimisation process performed in the following chapters is deemed to have converged when the increase in the quality of the worst element between two successive iterations is below 0.1%.

This effect is increased when the log-barrier function is used. As previously stated, with the log-barrier function the quality of an element is expressed as a function of the quality of the worst element in the mesh. After each iteration of the optimisation process, the quality of the worst elements changes, thus the function being minimised changes. Therefore, the large improvement in the quality of the worst elements observed in the first iterations in Figure 3.9, is repeated for several iterations.

3.4 Meshes

To demonstrate the effectiveness of the algorithms developed so far in this thesis and those presented in subsequent chapters, nine meshes are considered, Figure 3.10, including both 2D triangular meshes and 3D tetrahedral meshes. These mesh are the

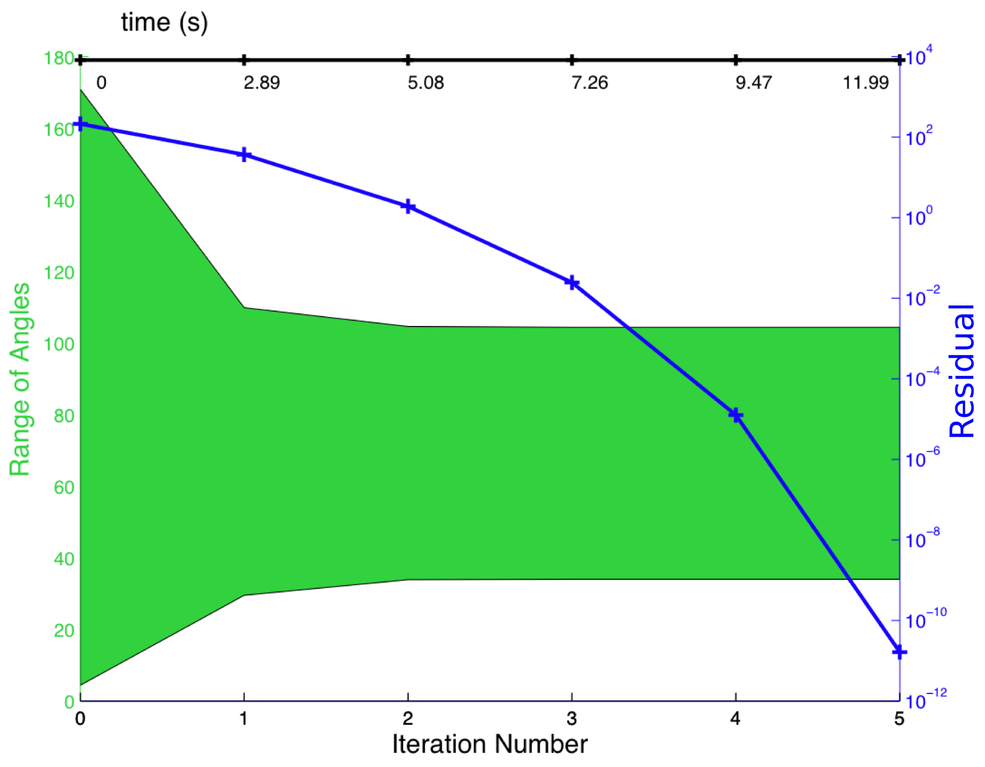


Figure 3.9: The quality of a mesh at each iteration of the optimisation process. The range of angles in the mesh is shown in green and the magnitude of the residual is shown in blue. The cumulative time taken for each iteration is shown on a non-linear bar on top.

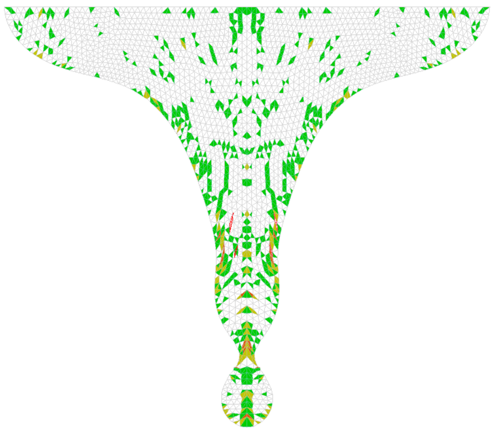
product of different mesh generation algorithms, have a large variance in their initial qualities and have many different boundary conditions and should therefore be a good means of evaluating the effectiveness of the mesh optimisation algorithms. There is also a wide variance in the number of elements in each mesh. Meshes with applications in both FEA and computer graphics are used.

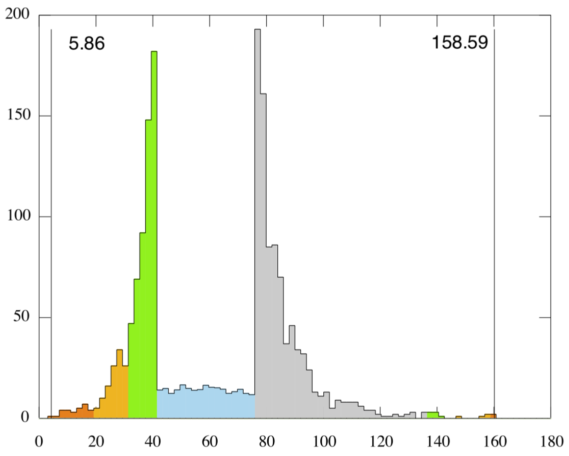
- "Square" comes included with Mesquite. The quality of this mesh was reduced from its original state as a further test of each quality measure.
- "Hanging droplet" and "oscillating droplet" are meshes obtained from intermediate stages of various axi-symmetric FE simulations of surface tension on micro-fluid droplets. These meshes are symmetric, thus optimisation is only performed on one half of each. However, the reflected image is shown for clarity.
- "Rand2" is a modified version of a mesh generated by [23]. This mesh was designed to demonstrate the effectiveness of flips and other topological changes. Thus, in its original form it is impossible to optimise without these topological changes. Several flips were performed to reach a configuration where optimisation would be effective.
- "Dragon" and "Cow" were generated by [34].
- The crack surface in "Graphite Brick" was formed by simulating an external force being placed on a graphite brick. As this mesh was obtained from an intermediate stage of an analysis from which many problems were found to be caused by poor quality elements, it is an ideal example of the class of problem from which the motivation for this thesis was drawn. This mesh is characterised by poor quality elements and its very complex crack surface. The crack surface is calculated by solving the system of physical equations associated with the analysis. Therefore, the only source of information relating to the shape of the domain is the mesh. The algorithms developed in Chapter 5 were developed to deal with such cases. As this mesh is of very poor quality, its optimisation would clearly be beneficial to the results of the FE analysis performed on it. The challenges associated with optimising this mesh are accentuated by the fact that all four nodes of the poorest elements are on the mesh surface.
- "Bone" was used in an FE simulation of a femur subject to mechanical loading [35]. The simulation of biological entities is very demanding on meshes. For example, the simulation of bone growth requires the mesh to adapt in a very complex manner.

- The crack surface in "Concrete Cylinder" was formed by simulating the pull-out of a steel anchor encased in a concrete cylinder [36]. This is another perfect example of the class of problem which could greatly benefit from mesh optimisation.

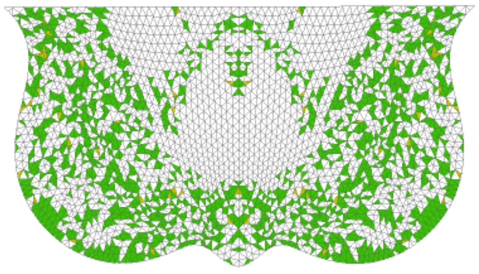
3.5 Summary

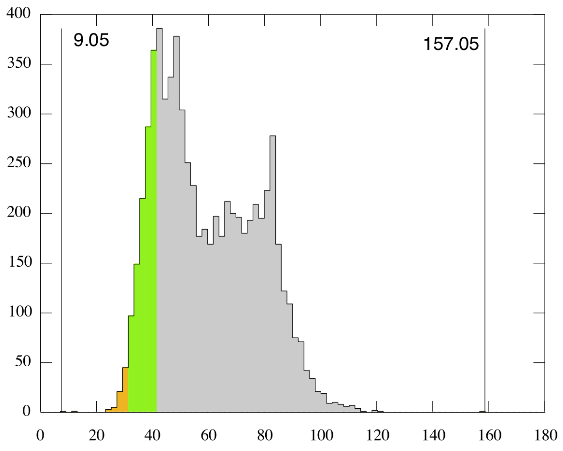
In this chapter, the quality measures and mesh optimisation algorithms which will be applied in later chapters were developed and explained. In the cases where existing algorithms were not suited to the problems at hand, new algorithms were developed. In the following chapter the meshes presented in the previous section are optimised using the algorithms presented in this chapter.



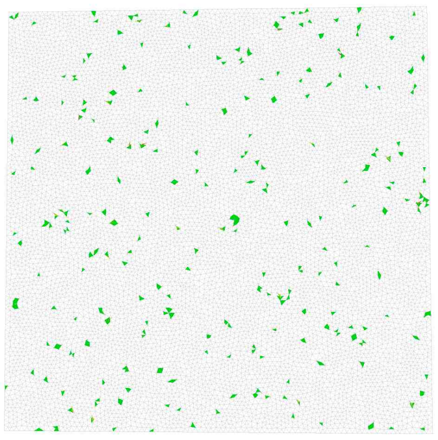


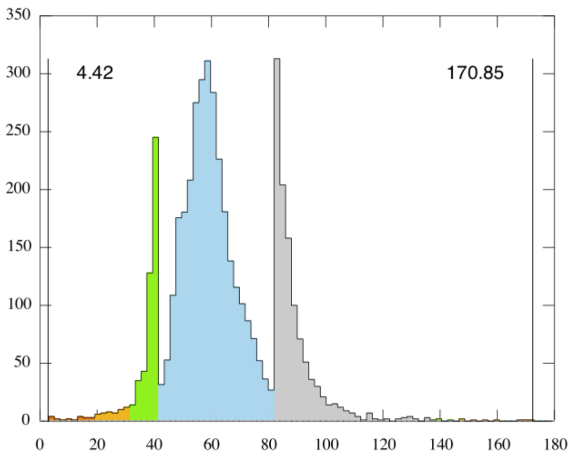
Hanging Droplet, 2126 Triangles





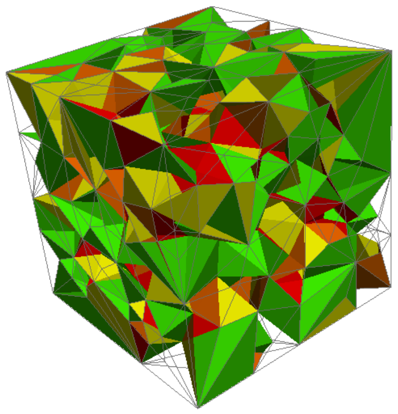
Oscillating Droplet, 2296 Triangles



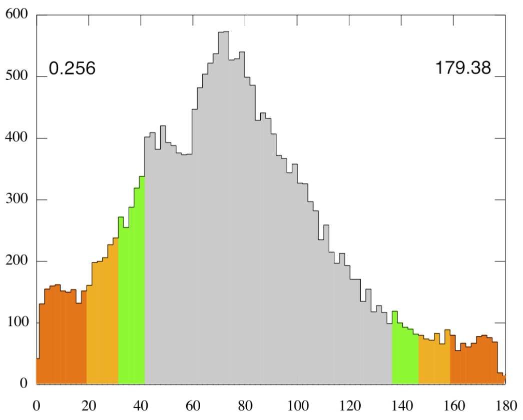


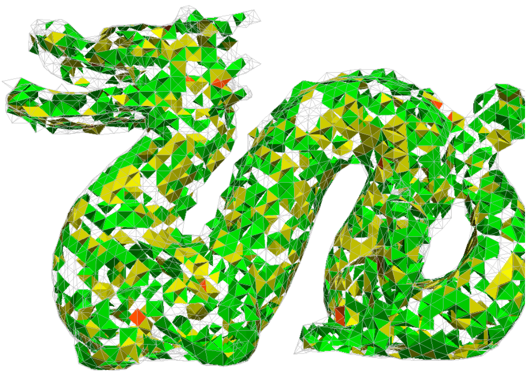
Square, 20258 Triangles

Figure 3.10: Meshes before improvement with Histograms of the range of dihedral angles (the height of blue columns have been divided by 20 due to the many occurrences of these angles). Red tetrahedra have angles under 10° or greater than 170°, orange tetrahedra have angles between 10° and 20° or 160° and 170°, yellow tetrahedra have angles between 20° and 30° or 150° and 160° and green tetrahedra have angles between 30° and 40° or 140° and 150°.

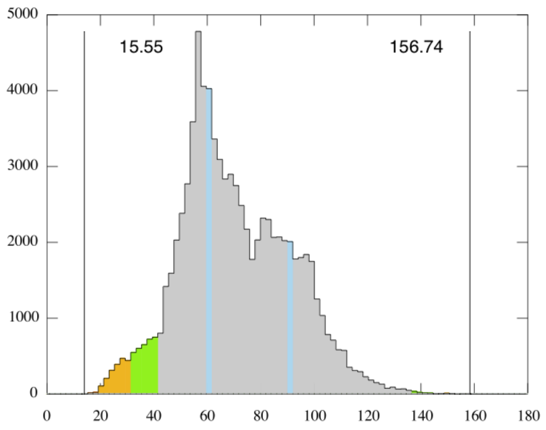


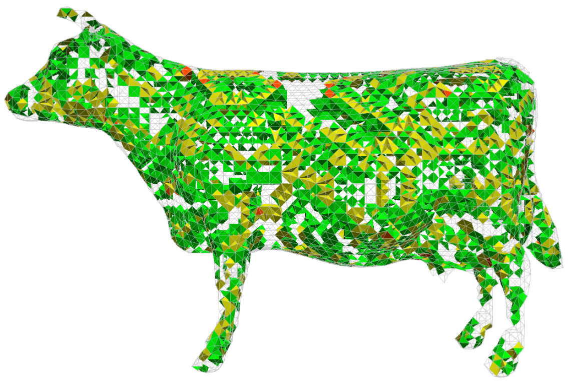
RAND2, 4372 Tetrahedra [37]





DRAGON, 32959 Tetrahedra [38]





COW, 42053 Tetrahedra

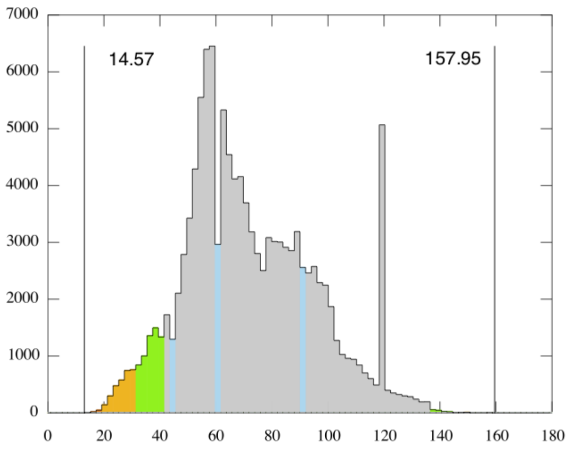
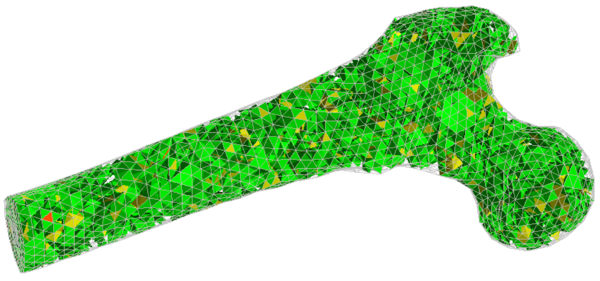
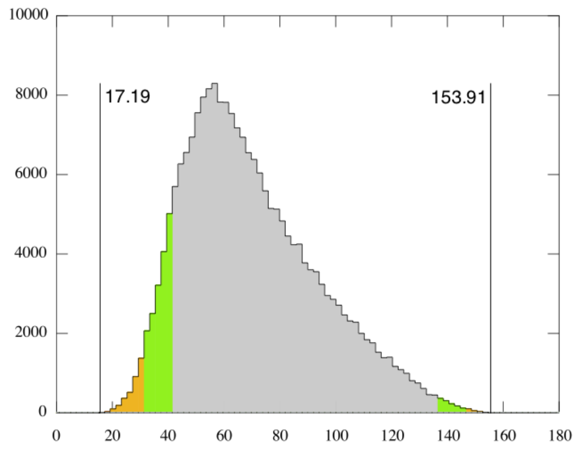
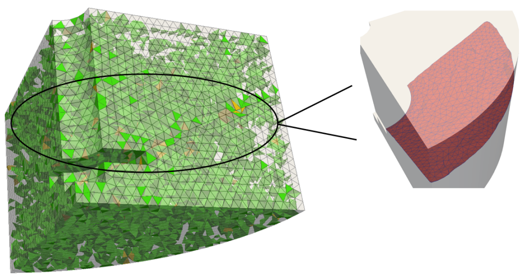


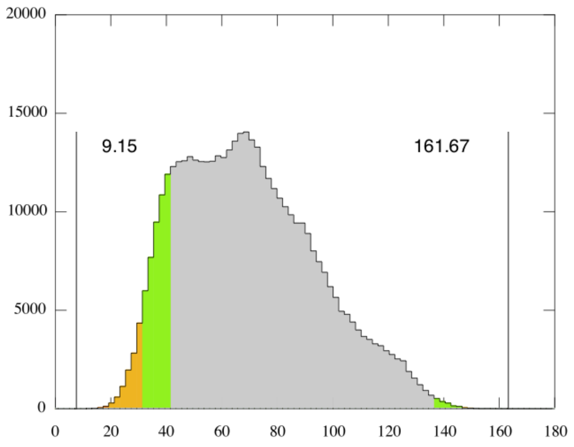
Figure 3.10: (contd.)



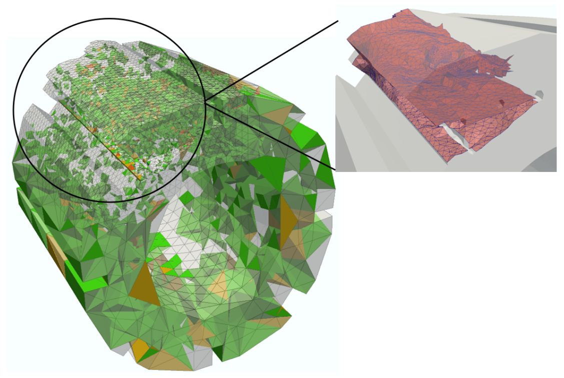


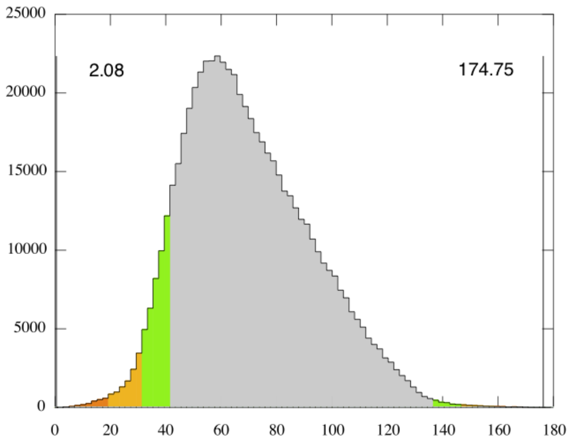
(a) Bone, 35832 Tetrahedra [35]





(b) Concrete Cylinder, 73684 Tetrahedra





(c) Graphite Brick , 100781 Tetrahedra

Figure 3.10: (contd.) Crack surfaces are shown in red.

Chapter 4

Unconstrained Mesh Optimisation Results and Discussion

4.1 Introduction

In this chapter the algorithms and techniques developed in the previous chapter are applied to the nine meshes introduced in Section 3.4. Each mesh was optimised using the area-length quality measure (AL) in 2D or the volume-length quality measure (VL) in 3D, the ideal weight inverse mean ratio (IMR) and the sine quality measure combined with both an $|L_1|$ objective function and the log-barrier objective function. The goal of this thesis is to develop algorithms and techniques to aggressively eliminate the worst elements in meshes, as these were found to be source of the problems associated with poor quality meshes. It was found in Section 3.3.3 that there is little benefit in allowing the optimisation process to run until it converges to a tight tolerance, as the worst elements are eliminated in the first iterations with only negligible improvement occurring after this point. Therefore, the optimisation process is terminated when the improvement in mesh quality between two successive iterations is less than a certain percentage, in this case a 0.1% increase in the quality of the worst element.

4.2 Results

The optimised mesh produced by each quality measure and objective function combination are shown in Figure 4.1. Histograms show the distribution of internal (2D)/dihedral

(3D) angles with the minimum and maximum angles and the range of angles shown. It should be noted that:

- red triangles/tetrahedra have angles under 10° or over 170°
- orange triangles/tetrahedra have angles under 20° or over 160°
- yellow triangles/tetrahedra have angles under 30° or over 150°
- green triangles/tetrahedra have angles under 40° or over 140°
- the heights of blue columns in the histograms have been divided by 20 because of the many occurrences of these angles
- Timings are given for a server running Linux with 16 dual core Intel Xeon processors and 512 GB of ram
- Five timings were taken and the average time is shown
- In each case the boundaries were unrestrained

It is reiterated here that this investigation is focused on unconstrained mesh optimisation. Constrained mesh optimisation, whereby the shape of the domain and area/volume is preserved throughout the optimisation process is investigated in the following two chapters. The layout of the results presented in the following pages is shown in Table 4.1.

	Initial Mesh	Initial range of angles
Obj Func / QM	$\lvert L_1 \rvert$	LB
AL/VL	Optimised Mesh	Optimised Mesh
	Optimised range of angles	Optimised range of angles
IMR	Optimised Mesh	Optimised Mesh
	Optimised range of angles	Optimised range of angles
Sine	Optimised Mesh	Optimised Mesh
	Optimised range of angles	Optimised range of angles

Table 4.1: Layout of results presented in the following pages. QM refers to quality measure, Obj Func refers objective function and LB refers to the log-barrier objective function.

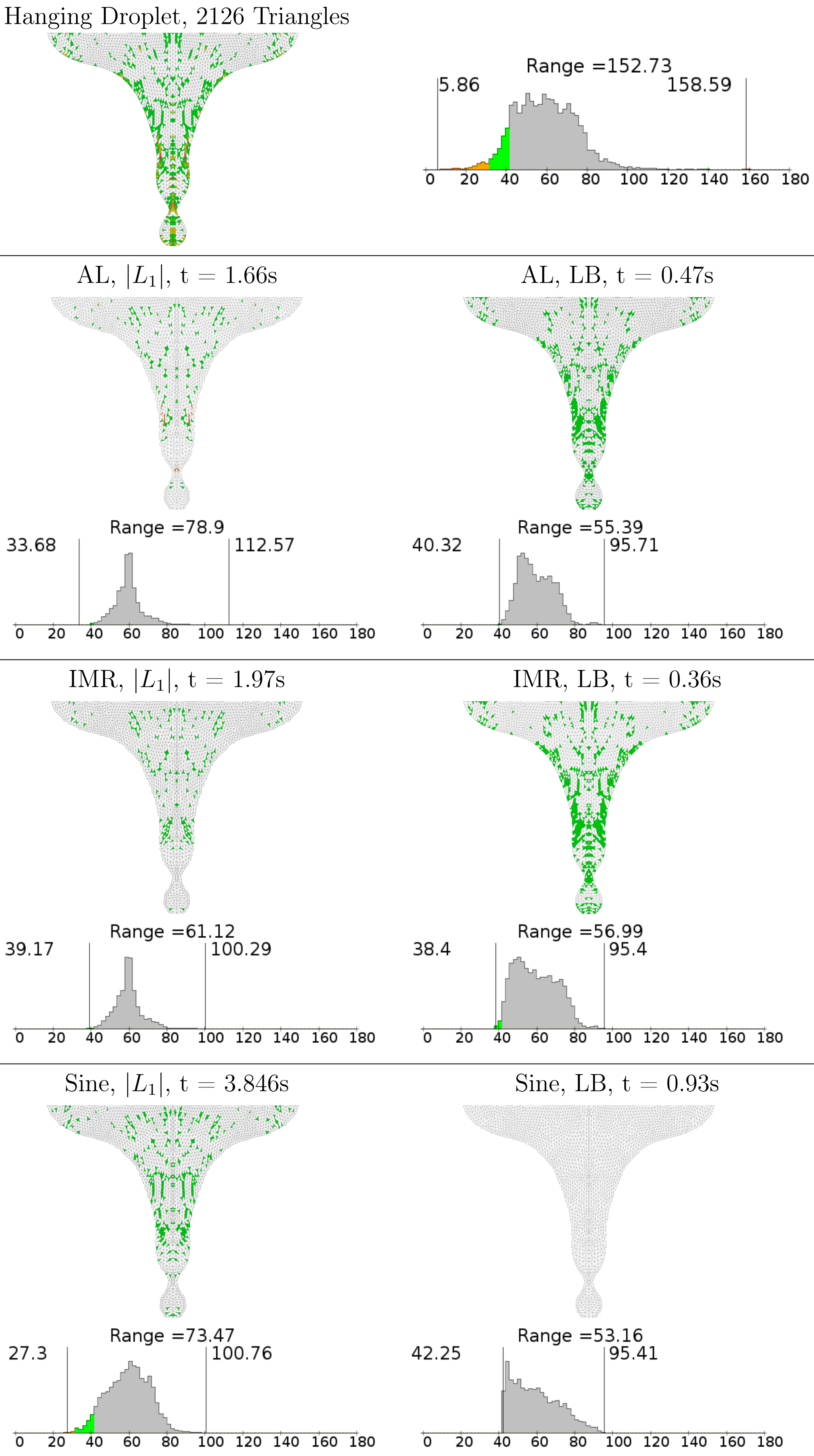


Figure 4.1: Unconstrained mesh optimisation results for "Hanging Droplet"

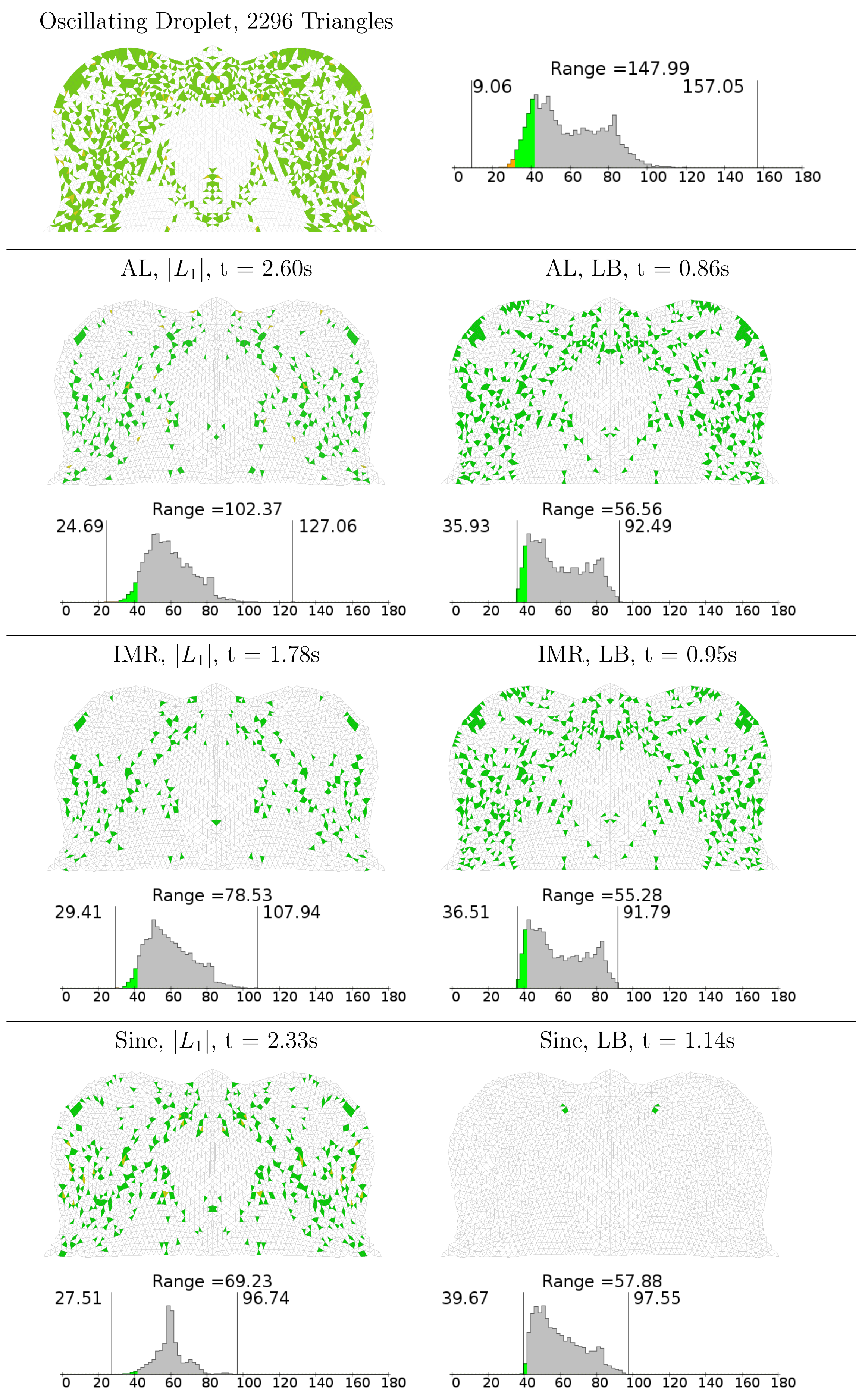


Figure 4.2: Unconstrained mesh optimisation results for "Oscillating Droplet"

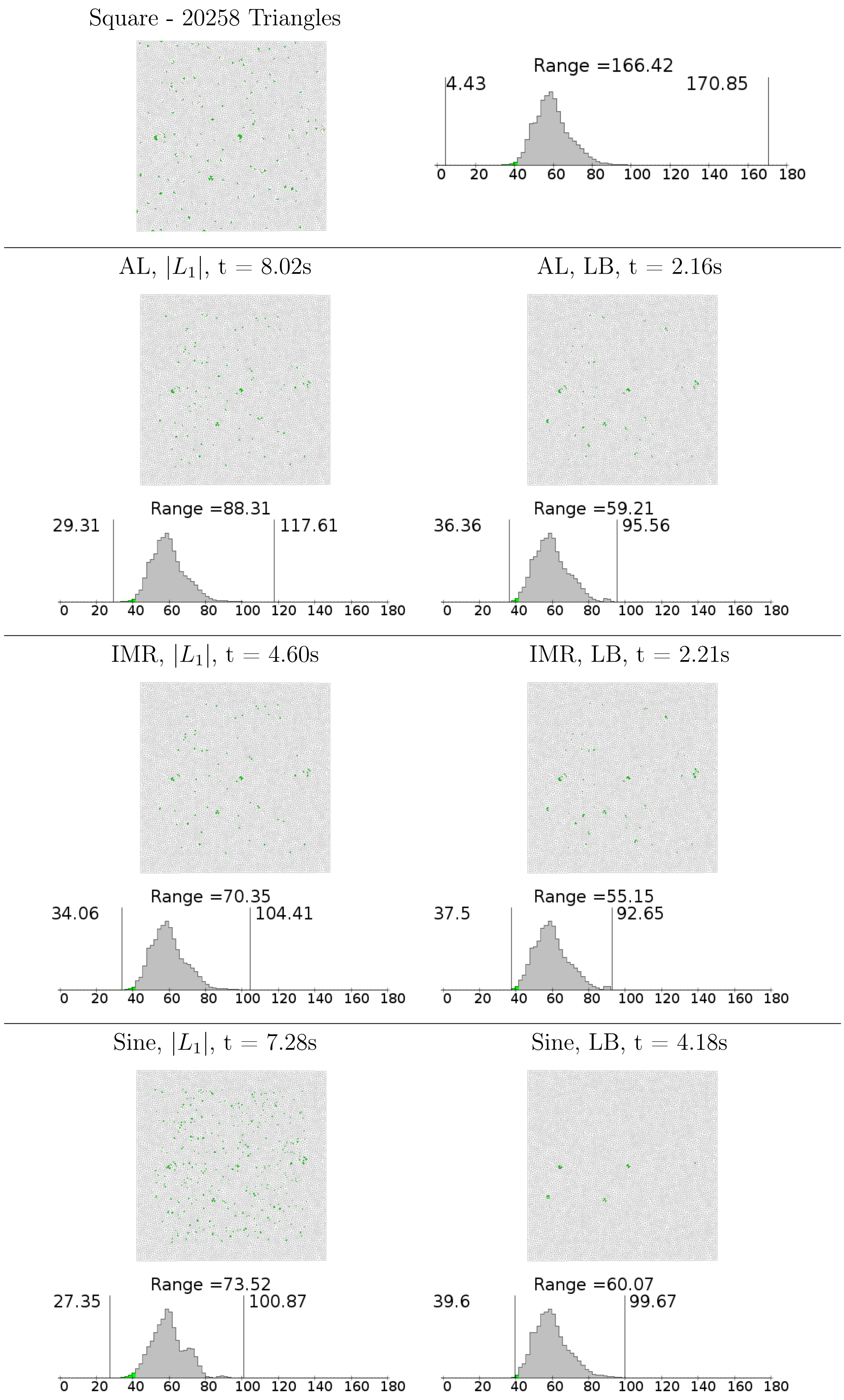


Figure 4.3: Unconstrained mesh optimisation results for "Square"

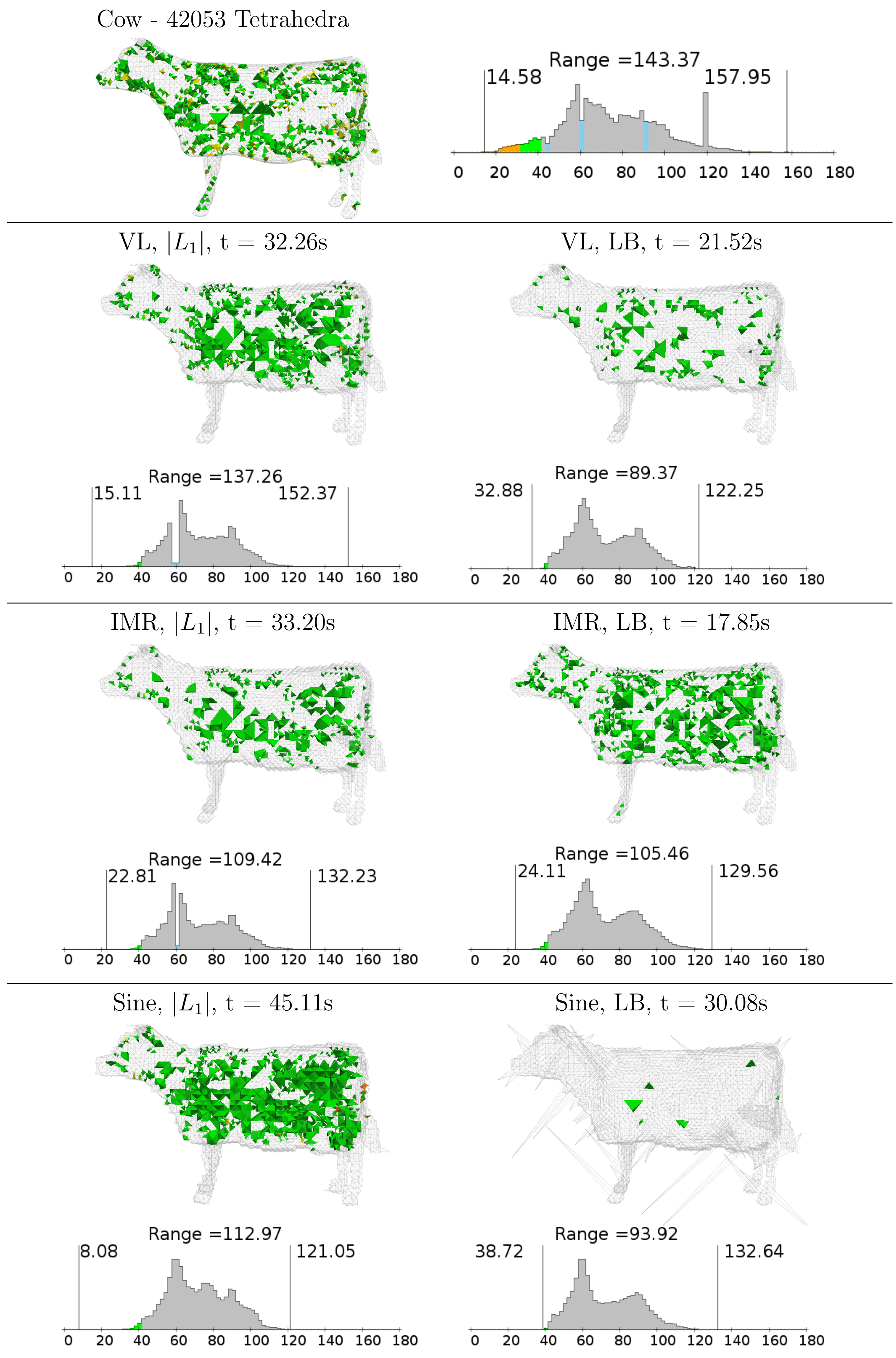


Figure 4.4: Unconstrained mesh optimisation results for "Cow"

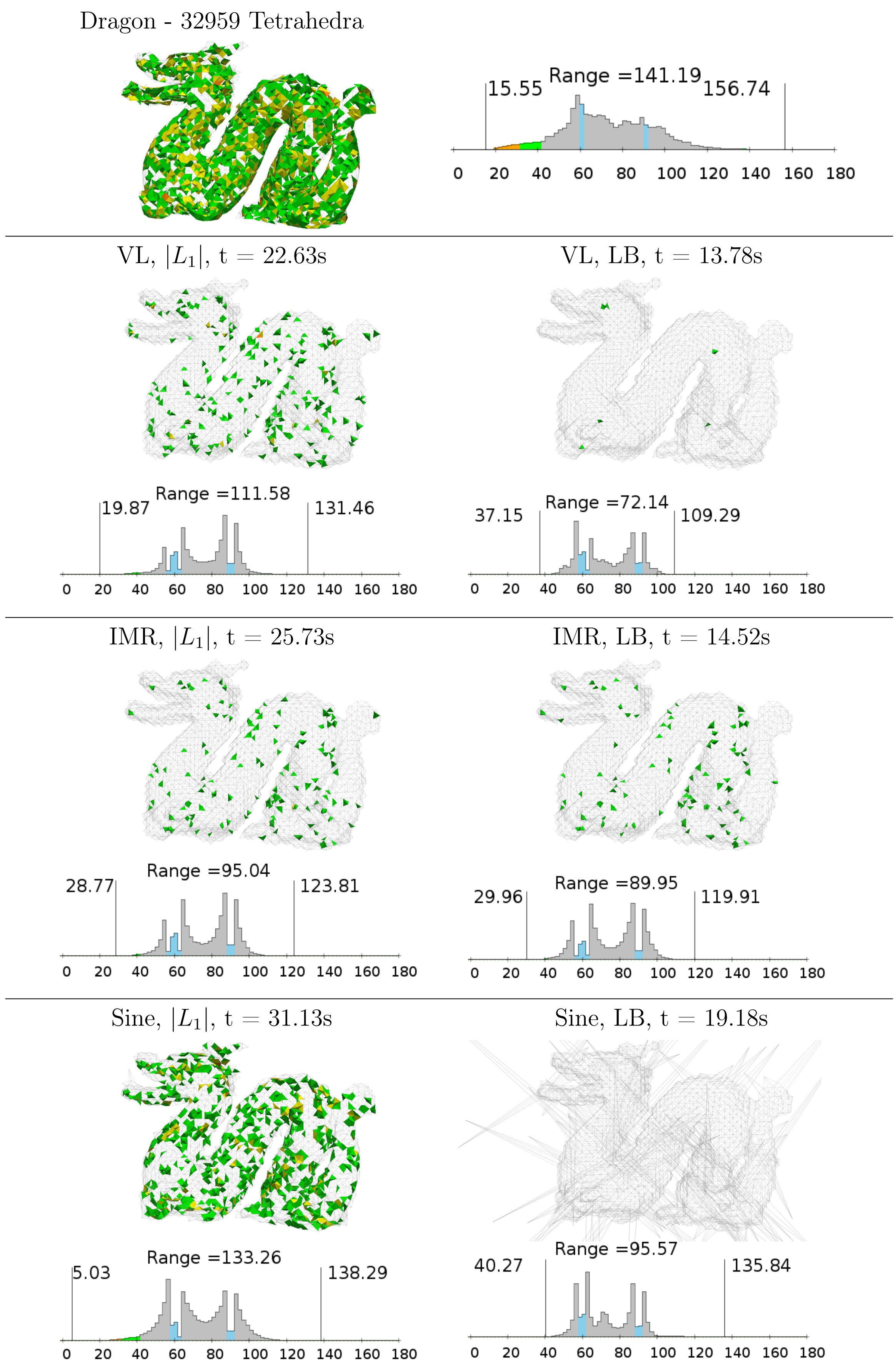


Figure 4.5: Unconstrained mesh optimisation results for "Dragon"

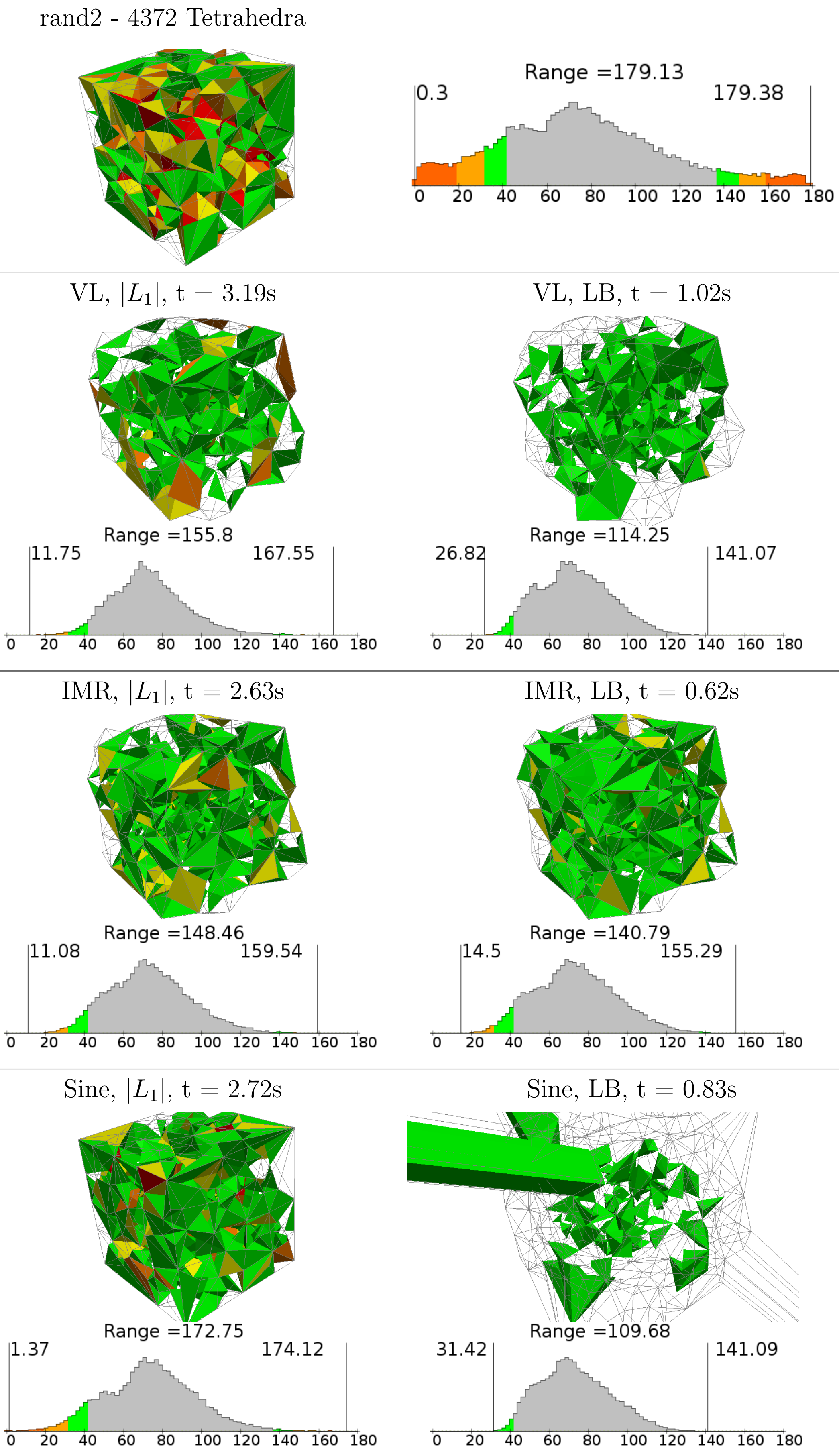


Figure 4.6: Unconstrained mesh optimisation results for "rand2"

Graphite Brick - 100781 Tetrahedra

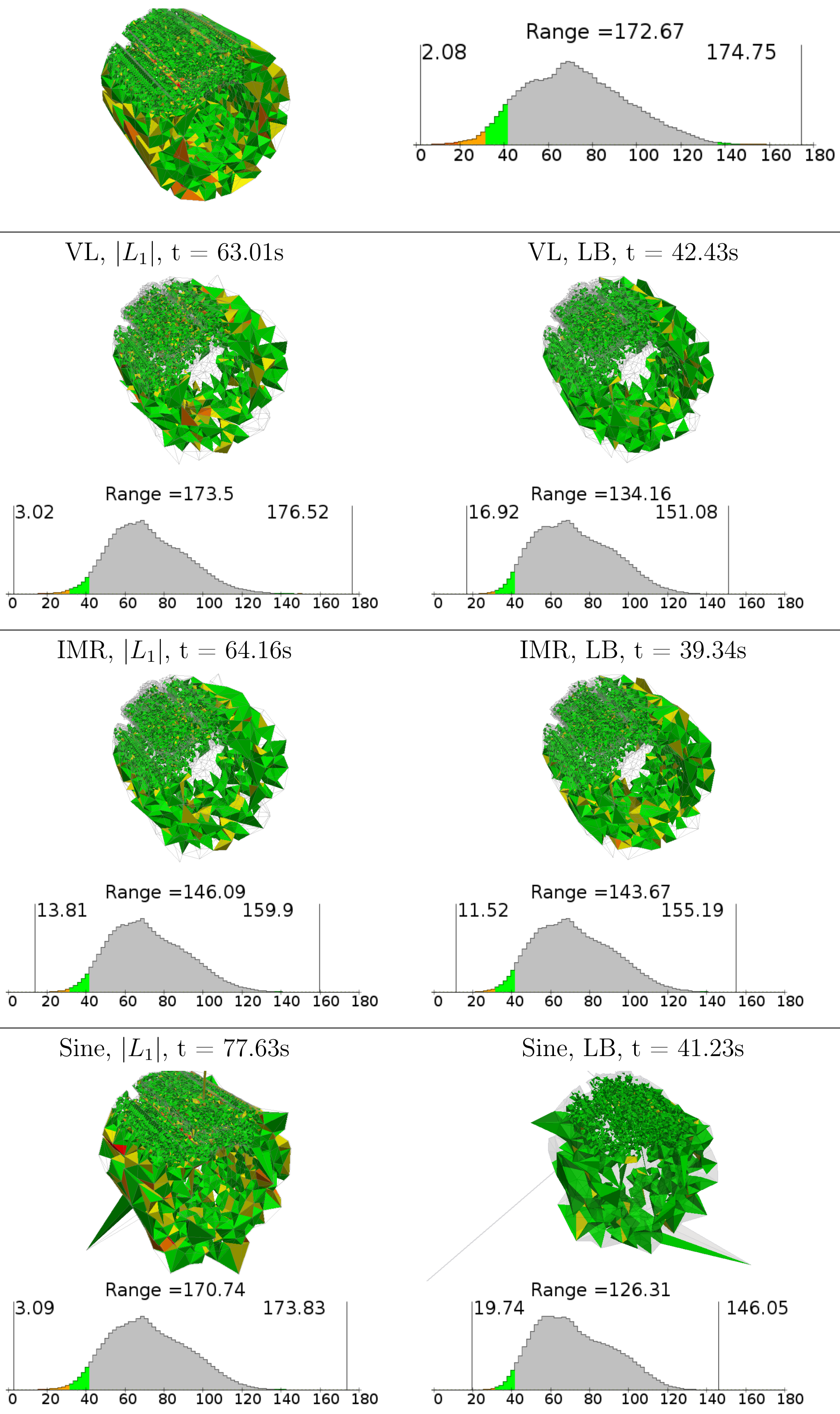


Figure 4.7: Unconstrained mesh optimisation results for "Graphite Brick"

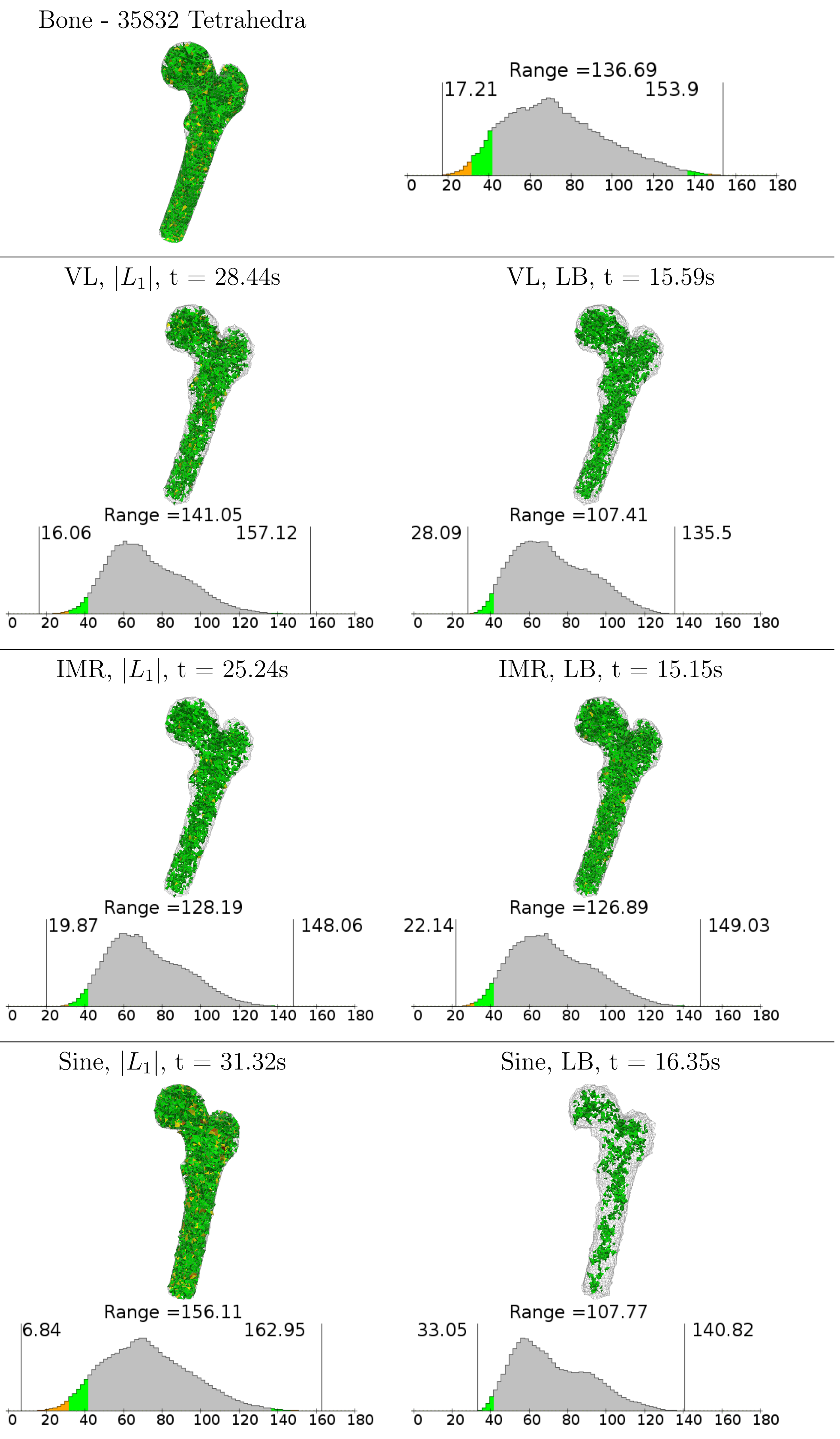


Figure 4.8: Unconstrained mesh optimisation results for "Bone"

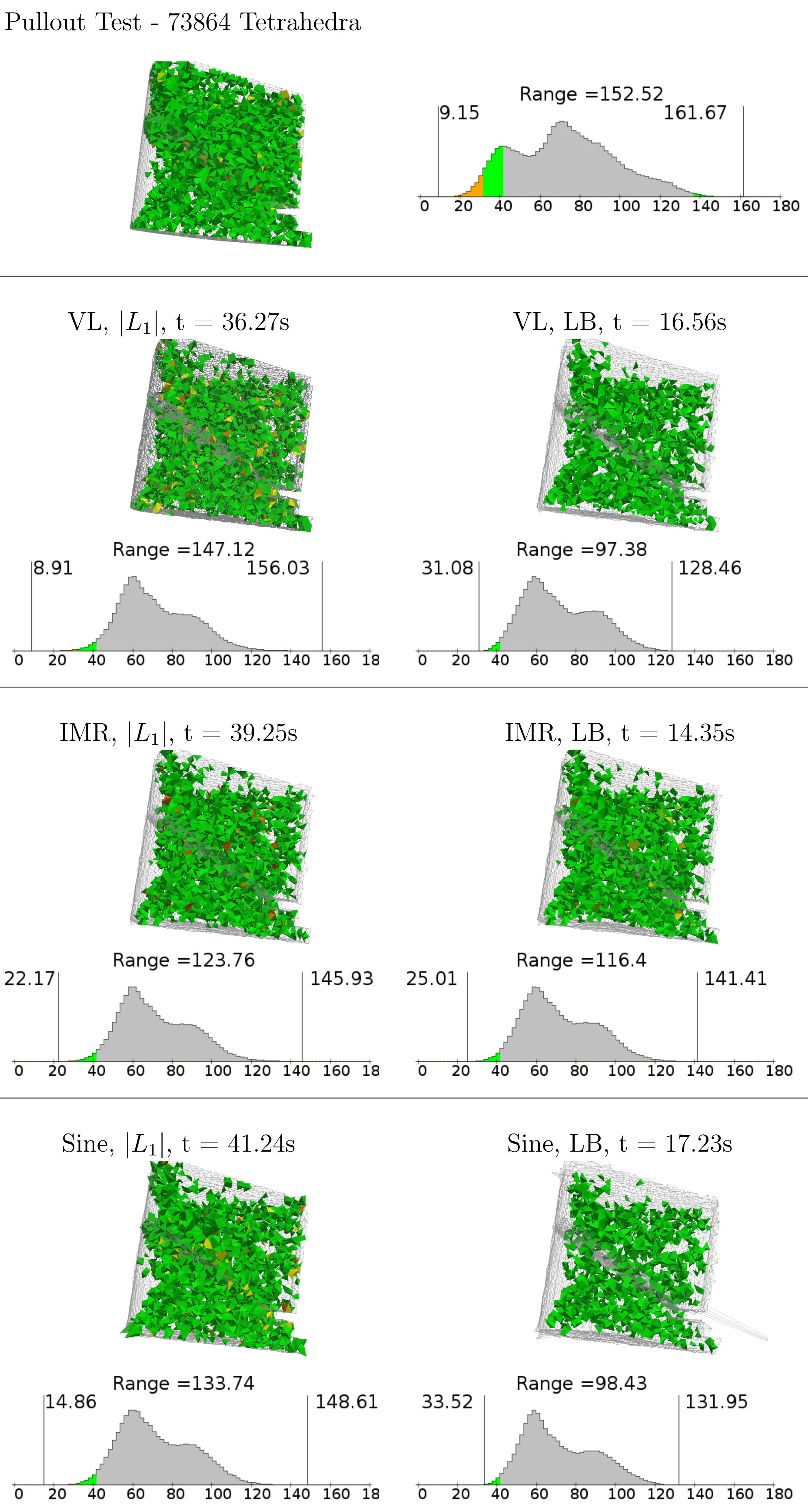


Figure 4.9: Unconstrained mesh optimisation results for "Pullout test"

4.3 Discussion

4.3.1 2D Results

It may be seen that each quality measure and objective function combination greatly improved each of the three 2D meshes. However there remain clear differences between the results obtained and the time taken to obtain these. Firstly the results obtained using each quality measure combined with an $|L_1|$ objective function will be discussed.

In terms of eliminating small angles, the IMR quality measure was the most effective for all three meshes. For each mesh, the use of the IMR quality measure resulted in a significantly greater smallest angle. In terms of large angles, the sine quality measure was overall the most effective. For both "Oscillating Droplet" and "Square", the sine quality measure has a much smaller largest angle than the other two quality measures and for "Hanging Droplet", the difference between the largest angle achieved by the IMR and the sine quality measure is negligible. When the time taken is accounted for, it must be concluded that when using an $|L_1|$ objective function, the IMR is the clear choice of quality measure to use.

For the log-barrier objective function, the results obtained differ greatly from those obtained using the $|L_1|$ objective function. In terms of effectiveness in eliminating small angles, the sine quality measure is the most effective, contrary to what was found for the $|L_1|$ objective function. Overall, both the AL and IMR perform equally well at eliminating small angles. For "Hanging Droplet", all three quality measures achieve very similar largest angles, however, for the other two meshes, the IMR is best, followed closely by the AL. The sine quality measure is the poorest concerning elimination of large angles.

In terms of time taken to optimise each mesh using the log-barrier objective function, one observation is resoundingly clear: the log-barrier objective function is significantly more efficient than the $|L_1|$ objective function. The decision to terminate the optimisation process when the quality of the worst element increases by less than 0.1% along with the log-barrier objective function harshly penalising the worst elements are the reasons why this is so. During the first iterations of the optimisation process, the focus is placed on the worst elements, thus meaning that these are eliminated much sooner in the optimisation process.

The conclusions as to which quality measure is best to use in 2D is not as clear as that concerning objective function. The optimisation of meshes using the sine quality measure was found to be more expensive compared with the other two quality measures. In terms of efficiency and quality of results achieved, there is very little distinction between either the IMR or the AL, although overall, the IMR performed slightly better.

4.3.2 3D Results

The examination of the optimised meshes produced by applying each of the three quality measures (VL, IMR and sine) and the $|L_1|$ objective function yields some interesting conclusions. Firstly, in terms of eliminating both large and small angles, the IMR quality measure is clearly the best quality measure to use in 3D when combined with an $|L_1|$ objective function. For every mesh except "Cow" and "rand2", it produced much better quality meshes than the other quality measures. The mesh produced by applying the sine quality measure to "Cow" has a much smaller largest dihedral angle than that produced by either the IMR quality measure or VL quality measure . The application of the VL quality measure to "rand2" produced a mesh with a slightly larger smallest angle, although this is not considerable. Similar to the timings observed in two dimensions, the sine quality measure in almost every case is slower than the other quality measures. The VL quality measure is marginally quicker than the IMR quality measure, although the IMR quality measure produces better meshes. Therefore, it may be concluded that when combined with an $|L_1|$ objective function, the IMR quality measure is the most effective of the three quality measures considered. It is interesting to note that when the initial mesh was of intermediate quality, as it the case for "Cow", "Dragon" and "Bone", optimisation using the sine quality measure actually reduced the smallest angle in the mesh. This is in agreement with the prediction made in Section 3.3, which stated that when mesh quality is measured using an $|L_1|$ objective function, minor improvement of many average quality elements, even if it occurs at the expense of harming the poorest quality elements, will result in a higher mesh quality.

The optimisation of the six 3D meshes using the log-barrier objective function is now examined. The sine quality measure is consistently the most effective at eradicating the smallest angles in meshes. Interestingly, the effectiveness of the IMR quality measure observed in two dimensions is it not observed here. The VL quality measure is much more effective than the IMR quality measure at eliminating small angles in all six of the meshes. For large angles, the VL quality measure is also very effective. For every mesh excluding "Graphite Block", where the sine quality measure is most effective, it performs the best. Similar to small angles, the IMR quality measure is not as effective at eliminating large angles in 3D as the other quality measures.

As was observed for the two-dimensional results, the sine quality measure is significantly more expensive than the other two quality measures. The IMR quality measure is slightly more efficient than the VL quality measure . However, the VL quality measure produced considerably better quality meshes than the IMR quality measure, thus, making this irrelevant.

One clear conclusion which may be drawn from the three-dimensional results studied here, mirroring that from the two-dimensional case, is that the log-barrier objective function is much more effective than the $|L_1|$ objective function, both in terms of mesh quality and efficiency. A recommendation for the best quality measure for optimising meshes in 3D is not straightforward and depends strongly on the application requirements, i.e. do small or large angles cause the greatest problems. The following recommendations are made based on a generalist's needs. Although the IMR quality measure is the most efficient, it is not the best in terms of quality. If large angles and efficiency are the greatest concerns, the use of the VL is recommended, if small angles pose the greatest problems and efficiency is not a high priority, the sine measure is recommended.

It is very interesting to note that although it the focus of this research is the improvement of the worst internal/dihedral angles in meshes, quality measures which indirectly measure these angles perform better in many cases than the sine measure which directly targets poor angles.

As was predicted in the previous chapter, the use of the sine measure can cause spire tetrahedra to form where a node is free to move a large distance without adversely affecting the quality of its connected elements. "Bone" is the only mesh where this phenomena does not occur. In constrained optimisation, which is discussed in the following chapters, this phenomena does not occur as the boundary nodes are not free to move in such a manner. It is advisable to check all meshes optimised using the sine measure for spire tetrahedra if such elements are unsuited to the intended application. If a mesh has all good dihedral angles, but is of poor quality when measured using either VL or IMR quality measures, then spire tetrahedra are present.

Figure 4.10 shows the effect of not constraining the mesh surface. The boundary of the domain has completely changed and spire tetrahedra can clearly be seen, meaning the optimised mesh does not accurately discretise the original domain. In the following two

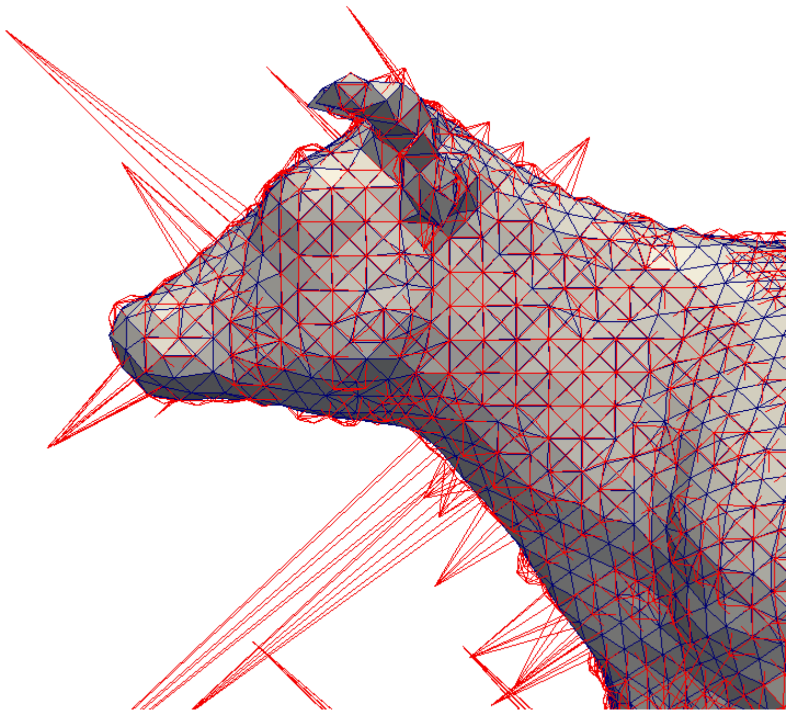


Figure 4.10: The blue mesh is the original mesh and the red mesh is optimised.

chapters, the development of a method which preserves both the shape of the domain and area/volume is discussed.

Chapter 5

Optimising Boundary Nodes

The goal of mesh optimisation is to produce higher quality meshes while ensuring that the optimised mesh is suitable for its intended purpose. Therefore, nodes which lie on the boundary of the domain require special treatment in order to conserve the shape of the domain and its area/volume. This is illustrated in Figure 5.1. This requires moving the free node, highlighted in red (Figure 5.1a), to its optimal position. However, both the domain's area and shape must be conserved. If this node is allowed to move freely then the domain shape and area/volume are liable to change, similar to that shown in Figure 5.1b. In order to conserve the shape of the domain and its area, the free node can only be moved either to the left or right, Figure 5.1c. To achieve this, a method must be developed that ensures that the calculated nodal search direction for boundary nodes preserves the shape of the domain and its area/volume. This is referred to as a constrained optimisation problem.

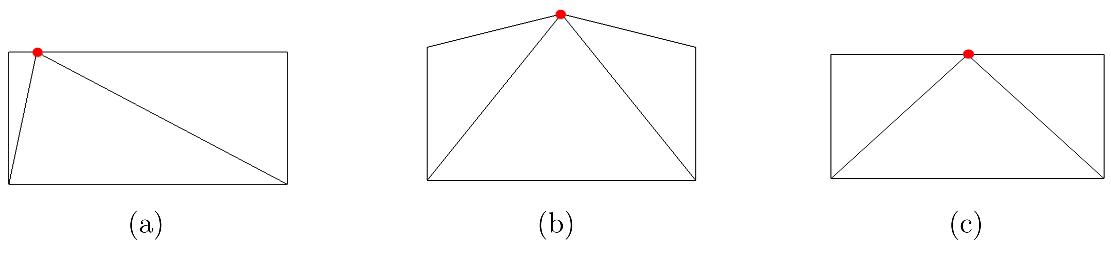


Figure 5.1: Example of constrained mesh optimisation, where the original mesh is shown in (a), the unconstrained optimised mesh is shown in (b) and the constrained optimised mesh is shown in (c).

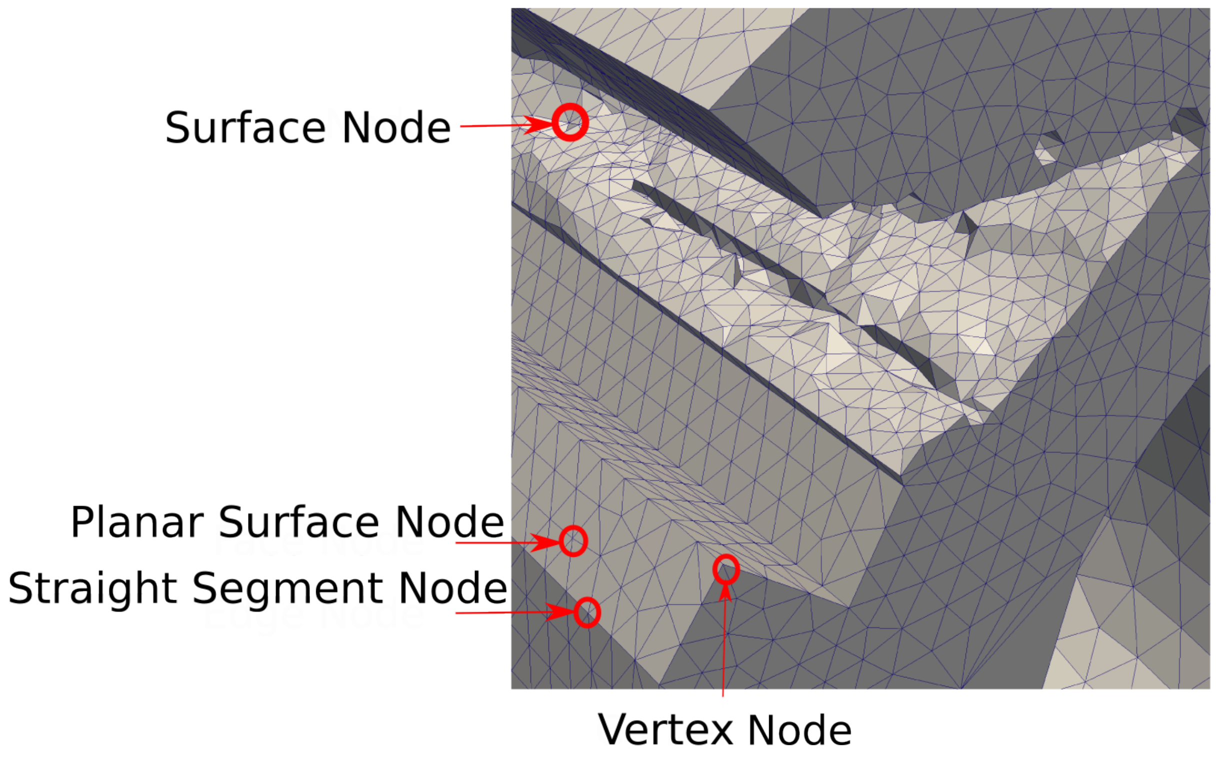


Figure 5.2: Boundary node classification. Each boundary node is classified according to the type of surface it lies on.

5.1 Classification of boundary nodes

The challenge here is to develop a method of calculating nodal search directions which both improve the quality of the mesh and preserves the domain boundary. It is important to distinguish between boundary nodes whose movement would alter the domain geometry and volume, for example the corner node of a cube, and other boundary nodes that can move in certain directions without changing the domain geometry/volume. Different types of domain boundaries present different challenges; therefore, a classification system for boundary nodes is required. The different boundary types are highlighted on the mesh in Figure 5.2.

- Vertex node: A vertex node is a node without which the domain geometry cannot be defined. Therefore, a vertex node cannot be moved without changing the domain shape.
- Planar Surface node: lies on a planar surface. A planar surface node may be moved in-plane without changing the domain shape.
- Straight Segment node: lies on a straight segment of a domain which is the

intersection of two planes. An straight segment node may be moved along the line separating these two planes.

- Surface node: A surface node lies on a non-planar surface. A surface node is difficult to move without changing the domain shape. Two methods for moving such nodes are presented in Section 5.3.

In the following sections, a straight segment node and a planar surface node will be treated as special cases. These are implemented separately as they represent a significant computational saving compared to the general, but extremely sophisticated process which is described after.

5.2 Movement of straight segment node

There are two main methods for moving straight segment nodes. The first involves moving the node freely to its optimal position that does not conserve domain geometry and then "snapping" it back to the closest point on the edge. This method has several disadvantages, primarily because the mesh optimisation process is unlikely to converge. Furthermore, if the mesh optimisation process is coupled with the solution of a physical problem, the "snap" procedure could introduce additional and significant residuals. Therefore, it is desirable to modify the system of mesh optimisation equations so that the calculated search direction is compatible with the constraints. In order to find the optimal nodal position, the optimisation procedure described in Chapter 3 is implemented, whereby the first and second derivatives of the objective function are used with a Newton solver. In 2D, the general system of equations for mesh optimisation, $\mathbf{S}\delta\mathbf{X} = -\mathbf{f}$, is given as:

$$\begin{bmatrix} f_{xx} & f_{xy} \\ f_{yx} & f_{yy} \end{bmatrix} \begin{bmatrix} \delta x \\ \delta y \end{bmatrix} = - \begin{bmatrix} f_x \\ f_y \end{bmatrix} \tag{5.1}$$

where, for example, f_x is the first derivative of the objective function with respect to x, f_{xy} is the second derivative of the objective function with respect to x then y and δx is the unknown nodal search direction in the x direction. Consider the problem shown in Figure 5.3, where the node cannot move in the y direction. The entries of the Hessian matrix and the rows of the gradient vector corresponding to movement of the constrained node in the y direction are replaced with zeros and the diagonal entries of the Hessian matrix with 1, Equation 5.2. The system of equations is then solved in the normal manner.

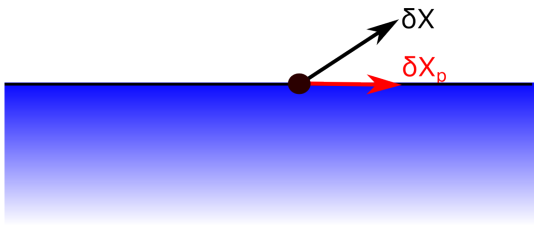


Figure 5.3: Constrained gradient.

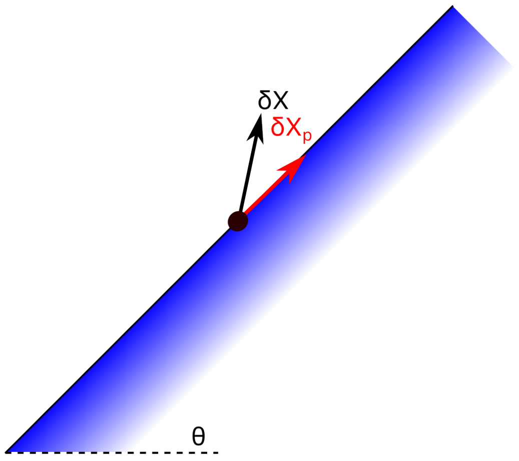


Figure 5.4: Constrained gradient when the axes do not align with the edge. The axes must be rotated so that it is aligned with the edge.

$$\begin{bmatrix} f_{xx} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \delta x \\ 0 \end{bmatrix} = - \begin{bmatrix} f_x \\ 0 \end{bmatrix} \tag{5.2}$$

For a more general case, such as that shown in Figure 5.4, it is necessary to rotate the system of equations as follows:

$$\mathbf{T}^{\mathrm{T}}\mathbf{S}\mathbf{T}\delta\mathbf{X} = -\mathbf{T}\mathbf{f} \tag{5.3}$$

where in 2D,

$$\mathbf{T} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \tag{5.4}$$

and θ is the angle of the segment from the x-axis. The constraint is then applied as before and the matrix and vector are rotated back to the original coordinate system.

Planar surface nodes are dealt with in a similar manner to straight segment nodes, thus this process is not described in detail here.

5.3 Movement of surface nodes

The challenge here is to increase the quality of meshes whilst preserving the domain geometry and area/volume. A common way of dealing with this is to not move surface nodes at all. However, this severely hinders the improvement process. As an example, all four nodes of the poorest elements in the mesh "Graphite Brick" presented in Section 3.4 are surface nodes. Therefore, it is not possible to optimise this mesh without moving surface nodes. The complex crack surface of this mesh was created by modelling physical equations, thus no continuous representation of the geometry exist. Only information which may be determined from the mesh surface may be used when optimising surface nodes. Therefore, an effective technique for moving surface nodes in a manner which respects both the domain geometry and volume is required. Without the ability to move surface nodes, meshes arising from the very class of problem from which the motivation for this thesis is drawn cannot be optimised.

The techniques described in the following sections are very general techniques which are effective when applied to both segment nodes and planar surface nodes. However, since the techniques described in the previous section for specific cases are both straight forward to implement and computationally efficient, it is much more sensible to utilise them where possible.

5.3.1 Surface quadrics

Stellar [18] uses a technique called quadric smoothing to move nodes lying on curved surfaces. This method assigns an error to a node which has been moved based on how far it has moved from the planes created by the original triangular faces that adjoined it [18]. This approach is summarised here. Let P be the set of planes created by the surface triangular faces adjoining a node, v. The quadric error for a point $\mathbf{x}$ relative to v is defined as:

$$Q_v(\mathbf{x}) = \Sigma \tau_i(\mathbf{x})^2 \tag{5.5}$$

where $\tau_i(x)$ is the perpendicular distance of x from the i^{th} plane. This means that if a node moves along a surface, there is no quadric error. However, if a node moves perpendicular to a surface, the quadric error increases rapidly. By limiting the quadric error, the amount by which a node may move from a surface is limited [18]. A penalty function is used to trade the quality of an element off against its quadric error. Klingner [18] has shown that it is possible to greatly improve meshes by making small changes to the surface.

Although using surface quadrics has been shown to be effective, this method has the disadvantage that the geometry of the domain is being changed and there is no guarantee that the volume will remain constant. Therefore, a method which only uses information which may be derived from the discretised domain and results in surface node movements which conserve the geometry and area/volume of the domain is required.

5.3.2 Generating surface constraints from the discretised domain

The following sections described the development and implementation of an algorithm capable of optimising surface nodes based only on information derived from the mesh surface.

5.3.2.1 Derivation of the constraint equation

This section discusses the development and implementation of an algorithm which allows for the movement of nodes on a non-planar surface. This algorithm does not change the underlying mesh geometry as it is based on the hypothesis that for a given shape, the volume to surface area ratio is a constant. Therefore:

$$\frac{V}{A} = C_0 \tag{5.6}$$

where V is the domain volume, A is the surface area of the domain and C_0 is a constant. Although in many FE simulations, the domain volume and surface area may change throughout the simulation, these changes are governed by the physical processes being simulated. It is of utmost importance that the mesh optimisation process does not affect these, as this will affect the results of the simulation. A simple two-dimensional example is shown in Figure 5.5. In two-dimensions, the same notation is used for convenience even though the ratio conserved is actually $\delta\frac{A}{L}$, where L is the length of the domain boundary. If $l = 10cm$, then $V = 60.35cm^2$ and $\frac{V}{A} = 0.6035$. If l increases to $15cm$, V becomes $135.8cm^2$ and $\frac{V}{A}$ becomes 0.905.

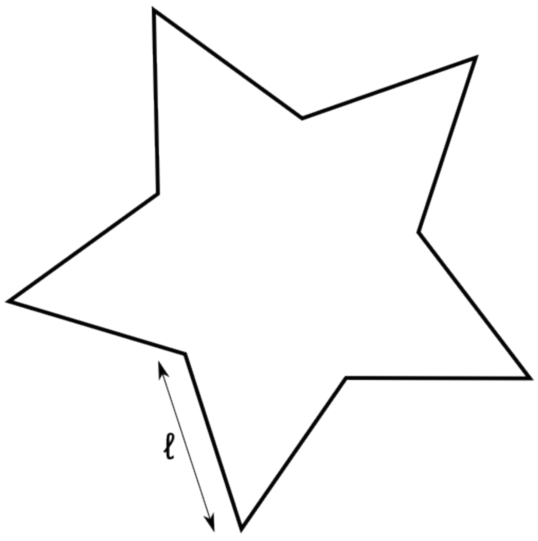


Figure 5.5: Star

From Equation 5.6

$$\int_V \mathrm{d}V = C_0 \int_A \mathrm{d}A \tag{5.7}$$

The following relation for the divergence of a vector will be used to modify the left hand side of this equation:

$$\nabla \cdot (\mathbf{X}) = \frac{\partial x}{\partial x} + \frac{\partial y}{\partial y} + \frac{\partial z}{\partial z} = 3 \tag{5.8}$$

where $\mathbf{X} = (x, y, z)$ is a Cartesian coordinate of a point on the surface. Therefore:

$$\int_V 1dV = \frac{1}{3}\int_V 3dV = \frac{1}{3}\int_V \nabla \cdot (\mathbf{X})dV \tag{5.9}$$

The divergence theorem states that [39]:

$$\int_V \nabla \cdot \mathbf{X}dV = \int_A \mathbf{X} \cdot \frac{1}{\|\mathbf{n}\|}\mathbf{n}dA \tag{5.10}$$

where $\mathbf{n}$ is the outward pointing normal at this point. Combining equations 5.7, 5.9 and 5.10, the volume integral becomes a surface integral:

$$\int_V \mathrm{d}V = \frac{1}{3}\int_V \nabla \cdot (\mathbf{X})\mathrm{d}V = \frac{1}{3}\int_A \mathbf{X} \cdot \frac{1}{\|\mathbf{n}\|}\mathbf{n}\mathrm{d}A \tag{5.11}$$

Combining equations 5.7 and 5.11:

$$\frac{1}{3}\int_A \mathbf{X} \cdot \frac{1}{\|\mathbf{n}\|}\mathbf{n}\mathrm{d}A = C_0 \int_A \mathrm{d}A \tag{5.12}$$

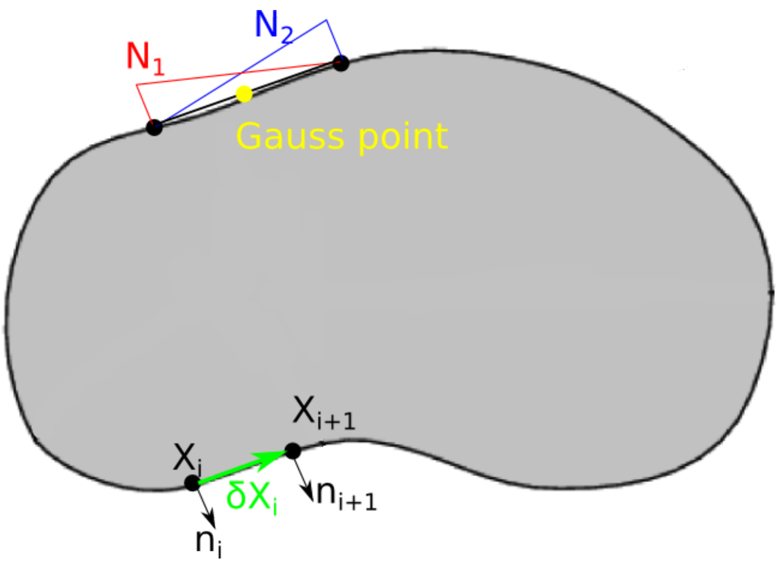


Figure 5.6: Discretised and continuous domain

Rewriting the above gives:

$$\frac{1}{3}\int_A (\mathbf{X}\cdot\frac{1}{\|\mathbf{n}\|}\mathbf{n} - C_1)\mathrm{d}A = 0 \quad \text{where} \quad C_1 = 3\,C_0.$$

which yields a local variant as follows:

$$\mathbf{X}\cdot\frac{\mathbf{n}}{\|\mathbf{n}\|} = C_1 \tag{5.13}$$

A first order Taylor Series expansion yields:

$$\mathbf{X}_i\cdot\frac{\mathbf{n}_i}{\|\mathbf{n}_i\|} + \frac{\mathbf{n_i}}{\|\mathbf{n}_i\|}\cdot\frac{\partial\mathbf{X}_i}{\partial\mathbf{X}_i}\delta\mathbf{X}_{i+1} + \mathbf{X_i}\cdot\frac{1}{\|\mathbf{n}_i\|}\frac{\partial\mathbf{n}_i}{\partial\mathbf{X}_i}\delta\mathbf{X}_{i+1} -$$

$$(\mathbf{X}_i\cdot\mathbf{n}_i)\frac{\mathbf{n}_i}{\|\mathbf{n}_i\|^3}\frac{\partial\mathbf{n}_i}{\partial\mathbf{X}_i}\delta\mathbf{X}_{i+1} = C_1 \tag{5.14}$$

Where $\delta\mathbf{X}_i$ represents the change in position of point $\mathbf{X}$ between two successive iterations, $\mathbf{X}_i$ is the surface of domain at iteration i and $\mathbf{n}_i$ is the outward pointing normal at iteration i. These quantities are illustrated in in Figure 5.6. The shape functions (N) are shown in red and blue and these are used to calculate the coordinates of the gauss point, shown in yellow.

Rearranging,

$$\frac{\mathbf{N_i}}{\|\mathbf{n}_i\|}\cdot\delta\mathbf{X}_{i+1} + \mathbf{X_i}\cdot\frac{1}{\|\mathbf{n}_i\|}\frac{\partial\mathbf{n}_i}{\partial\mathbf{X}_i}\delta\mathbf{X}_{i+1} -$$

$$(\mathbf{X}_i \cdot \mathbf{n}_i)\frac{\mathbf{n}_i}{\|\mathbf{n}_i\|^3}\frac{\partial \mathbf{n}_i}{\partial \mathbf{X}_i}\delta\mathbf{X}_{i+1} = C_1 - \mathbf{X}_i \cdot \frac{\mathbf{n}_i}{\|\mathbf{n}_i\|} \tag{5.15}$$

The second and third terms of the left hand side cancel out, leading to the following surface constraint equation:

$$\frac{\mathbf{n_i}}{\|\mathbf{n}_i\|} \cdot \delta\mathbf{X}_{i+1} = C_1 - \mathbf{X}_i \cdot \frac{\mathbf{n}_i}{\|\mathbf{n}_i\|} \tag{5.16}$$

Equation 5.16 is enforced in a weighted residual sense:

$$\mathop{\mathsf{A}}_{k=1}^{n}\int_{A^e}\left(\mathbf{N}^T\frac{\mathbf{n}_i}{\|\mathbf{n}_i\|}\mathbf{N}\delta\mathbf{X}\right)dA = \mathop{\mathsf{A}}_{k=1}^{n}\int_{A^e}\left(\mathbf{N}^T C_1 - \mathbf{N}^T\mathbf{X}_i \cdot \frac{\mathbf{n}_i}{\|\mathbf{n}_i\|}\right)dA \tag{5.17}$$

Equation 5.17 may be rewritten as:

$$\mathbf{C}\delta\mathbf{X} = \mathbf{g} \tag{5.18}$$

where $\mathbf{C}$ is a constraint matrix and $\mathbf{g}$ is a residual vector. This constraint equation ensures that the volume to surface area ratio is conserved.

5.3.2.2 Enforcing the constraints

The mesh optimisation process involves solving the following non-linear system of equations:

$$\mathbf{S}\delta X = -\mathbf{f} \tag{5.19}$$

where $\mathbf{S}$ is the global Hessian matrix of the objective function, $\delta\mathbf{X}$ is the unknown nodal search directions and $\mathbf{f}$ is the global gradient vector of the objective function. The assembly of $\mathbf{S}$ and $\mathbf{f}$ is described in Section 3.3. Ainsworth [40] presents a method for modifying such a system of equations to account for the constraints of the form in Equation 5.18 and the following summary of this method is adapted from this paper and from Kaczmarczyk et al [41].

The mesh optimisation problem can be expressed as a constrained quadratic programming problem:

$$\begin{aligned} &\underset{\delta\mathbf{X}}{min} \qquad \mathcal{Q} = \frac{1}{2}\delta\mathbf{X}^{\mathrm{T}}\mathbf{S}\delta\mathbf{X} - \delta\mathbf{X}^{\mathrm{T}}\mathbf{f} \\ &\text{subject to } \mathbf{C}\delta\mathbf{X} - \mathbf{g} = 0 \end{aligned} \tag{5.20}$$

The Euler conditions for the stationary point of the Lagrangian are found to be

$$\begin{aligned} \mathbf{S}\delta\mathbf{X} + \mathbf{C}^{\mathrm{T}}\lambda &= -\mathbf{f} \\ \mathbf{C}\delta\mathbf{X} &= \mathbf{g} \end{aligned} \tag{5.21}$$

While it is possible to solve the constrained problem using this approach, the number of unknowns is increased and the character of the system matrix is altered to an indefinite saddle point problem. The numerical solution of such a system is very inefficient and thus, not suitable for use with large problems with a constraint applied to each node. The method developed by Ainsworth [40] does not modify the character of the system matrix and it produces a matrix with approximately the same condition number as the unconstrained stiffness matrix.

Assuming that the problem is well posed (that there exists a unique solution to the constrained problem and that the constraint matrix is of full rank), then the following matrices are well defined:

$$\mathbf{Q} = \mathbf{I} - \mathbf{C}^T(\mathbf{C}\mathbf{C}^T)^{-1}\mathbf{C} \qquad \mathbf{R} = \mathbf{C}^T(\mathbf{C}\mathbf{C}^T)^{-1} \tag{5.22}$$

where $\mathbf{Q}$ is a projection matrix and $\mathbf{R}$ is an auxiliary matrix. If the constrained matrix $\mathbf{S}'$ and the force vector $\mathbf{f}'$ are defined as:

$$\mathbf{S}' = \mathbf{C}^T\mathbf{C} + \mathbf{Q}^T\mathbf{S}\mathbf{Q} \tag{5.23}$$

and

$$\mathbf{f}' = \mathbf{C}^T\mathbf{g} + \mathbf{Q}^T(\mathbf{f} - \mathbf{S}\mathbf{R}\mathbf{g}) \tag{5.24}$$

there exists a unique solution $\delta\mathbf{X}$ to the problem

$$\delta\mathbf{X} = \mathbf{S}^{-1}\mathbf{f} \tag{5.25}$$

and the corresponding Lagrange multipliers can be recovered from

$$\lambda = \mathbf{R}^T(\mathbf{f} - \mathbf{S}\delta\mathbf{X}) \tag{5.26}$$

The system is modified as follows to explicitly account for the constraint equation:

$$\mathbf{S}'\delta X = -\mathbf{f}' \tag{5.27}$$

where

$$\mathbf{S}' = \mathbf{C}^T\mathbf{C} + \mathbf{Q}^T\mathbf{S}\mathbf{Q} \tag{5.28}$$

$$\mathbf{f}' = \mathbf{C}^T\mathbf{g} + \mathbf{Q}^T(\mathbf{f} - \mathbf{S}\mathbf{R}\mathbf{g}) \tag{5.29}$$

The solution obtained from solving this modified system of equations will respect any constraints applied to it. This means that nodes which lie on complex surfaces may now be moved whilst preserving the underlying mesh geometry and domain volume. The ability to move such nodes permits the optimisation of meshes which previously could not be. It is worth noting that this technique is equally effective when applied to a curved segment node.

5.4 Summary

In this chapter, algorithms which allow the optimisation of boundary nodes whilst preserving both shape of the domain and its area/volume were developed. In the following chapter, these algorithms are applied to the nine meshes introduced in Section 3.4.

Chapter 6

Constrained Mesh Optimisation Results and Discussion

6.1 Introduction

In this chapter, the constrained mesh optimisation algorithms developed in the previous chapter are applied to the nine meshes introduced in Chapter 3. In a similar manner to Chapter 4, each of the nine meshes was optimised using the area-length quality measure (AL) in 2D or the volume-length quality measure (VL) in 3D, the ideal weight inverse mean ratio (IMR) and the sine measure combined with both an L_1 objective function and the log-barrier objective function. As described in the previous chapter, the most suitable set of constraints was applied to each surface node in order to ensure that the shape and area/volume of the domain was conserved in the most efficient manner possible. The same convergence criteria used in Chapter 4 were also used here, that is that the optimisation process was terminated when the improvement in quality of the worst element between two successive iterations is less 0.1%.

6.2 Results

The results presented on the following pages in Figures 6.1-6.9 are presented in the same manner as in the previous results chapter. One piece of additional information is included, that is the change in the volume to surface area ratio, $\delta\frac{V}{A}$. For the two-dimensional meshes, the same notation is used for convenience even though the ratio conserved is actually $\delta\frac{A}{L}$, where L is the length of the domain boundary. The layout of the results is described in Table 4.1.

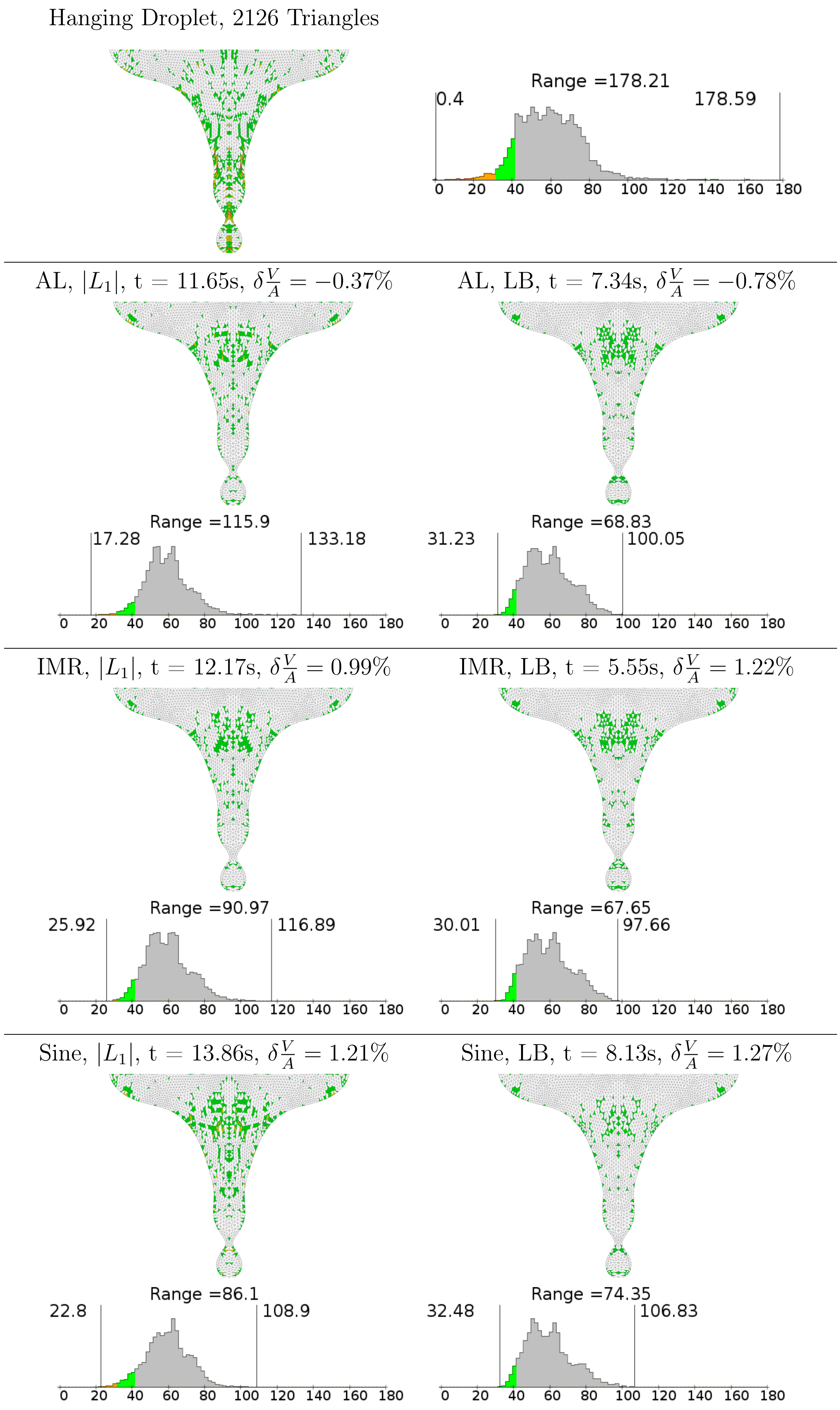
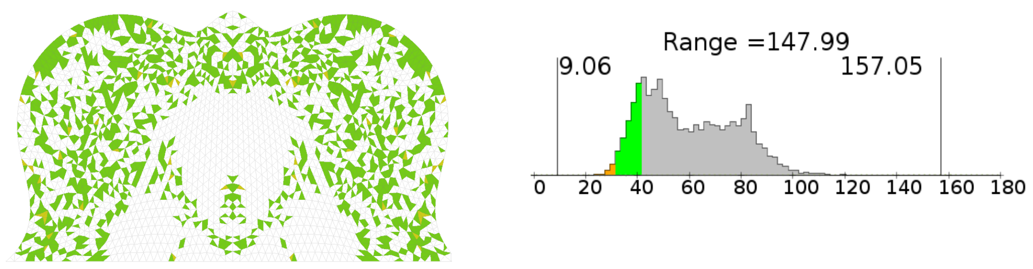
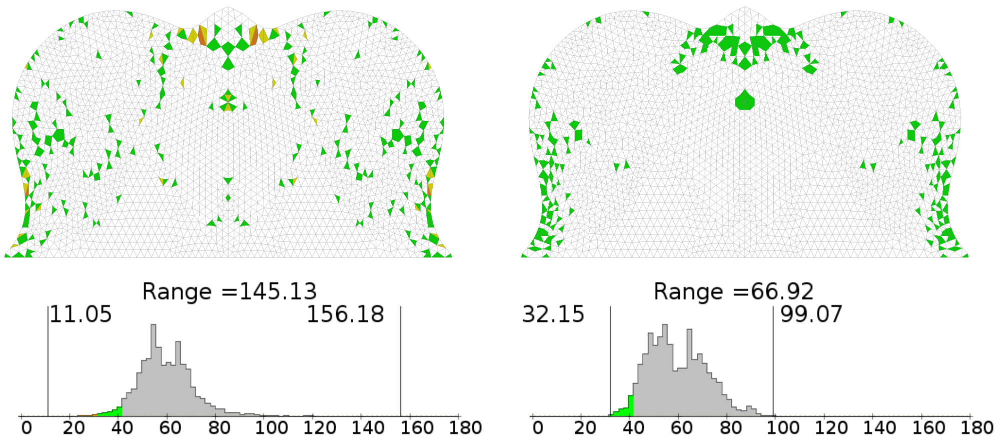


Figure 6.1: Constrained mesh optimisation results for "Hanging Droplet"

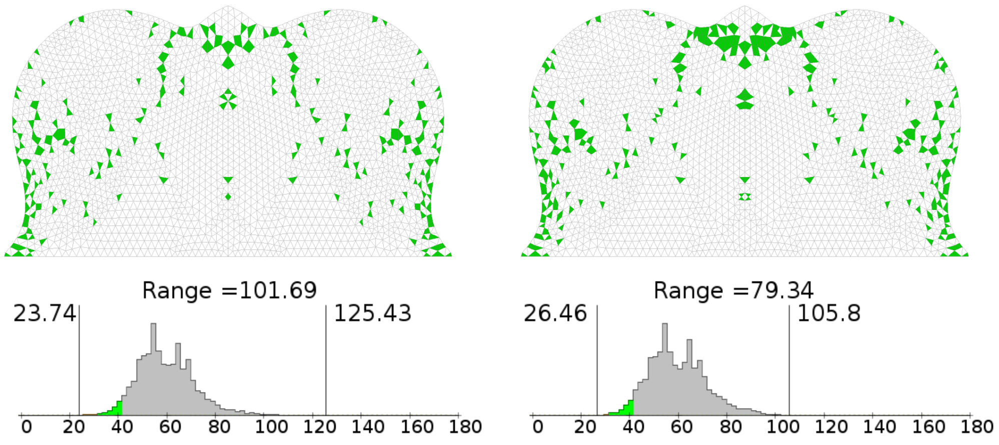
Oscillating Droplet, 2296 Triangles



AL, $|L_1|$, t $=$ 10.65s, $\delta\frac{V}{A} = 1.16\%$ AL, LB, t $=$ 7.08s, $\delta\frac{V}{A} = 1.32\%$



IMR, $|L_1|$, t $=$ 11.06s, $\delta\frac{V}{A} = -0.71\%$ IMR, LB, t $=$ 6.58s, $\delta\frac{V}{A} = 0.65\%$



Sine, $|L_1|$, t $=$ 11.87s, $\delta\frac{V}{A} = 1.03\%$ Sine, LB, t $=$ 8.13s, $\delta\frac{V}{A} = -0.17\%$

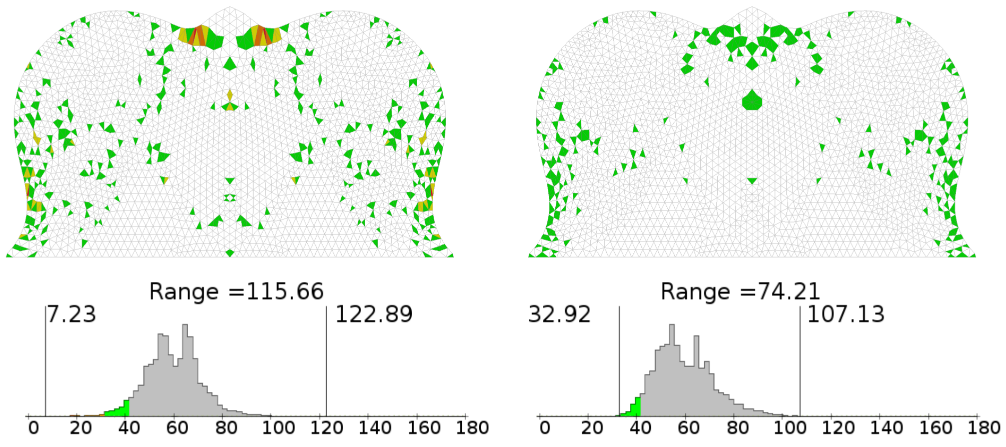


Figure 6.2: Constrained mesh optimisation results for "Oscillating Droplet"

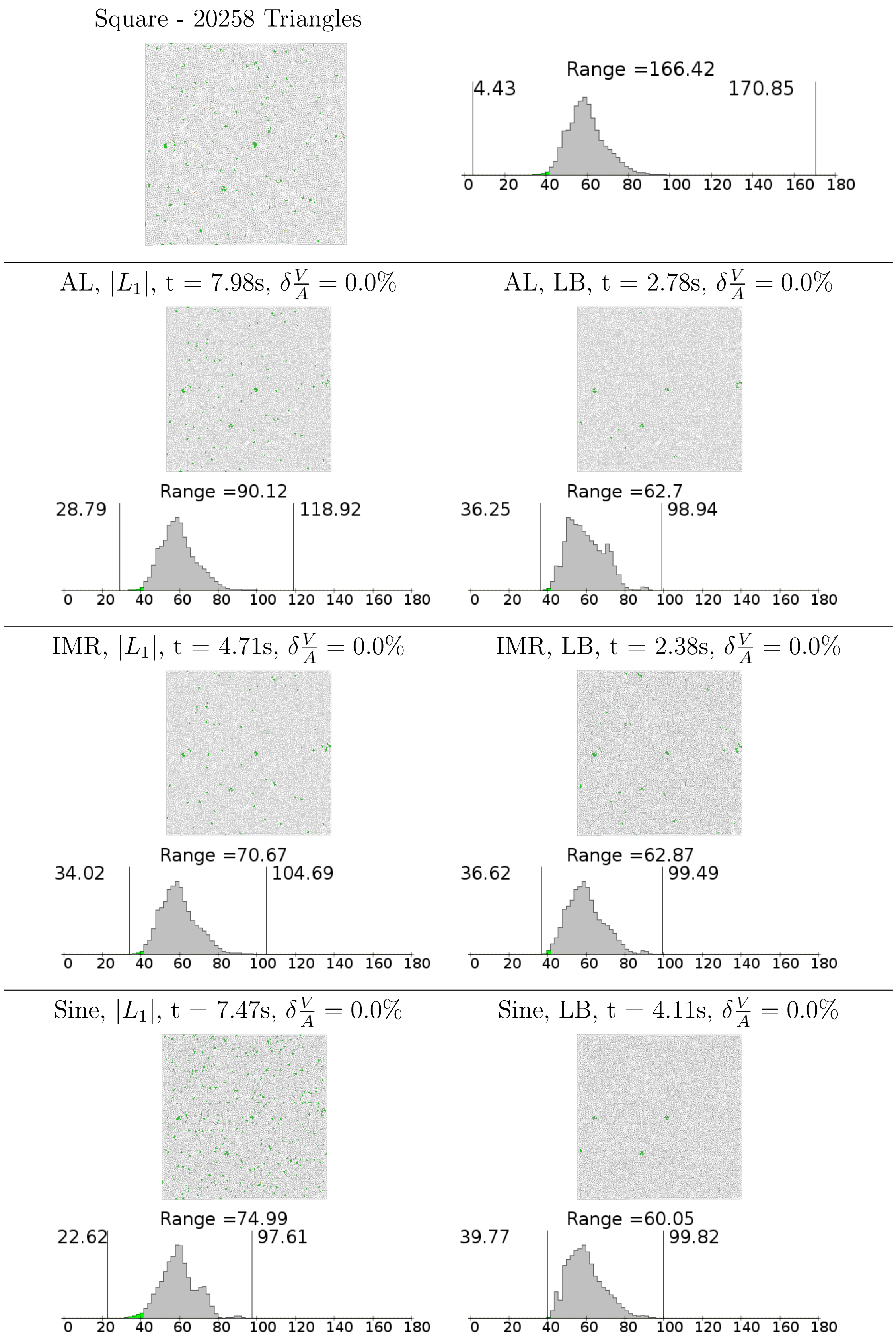


Figure 6.3: Constrained mesh optimisation results for Square"

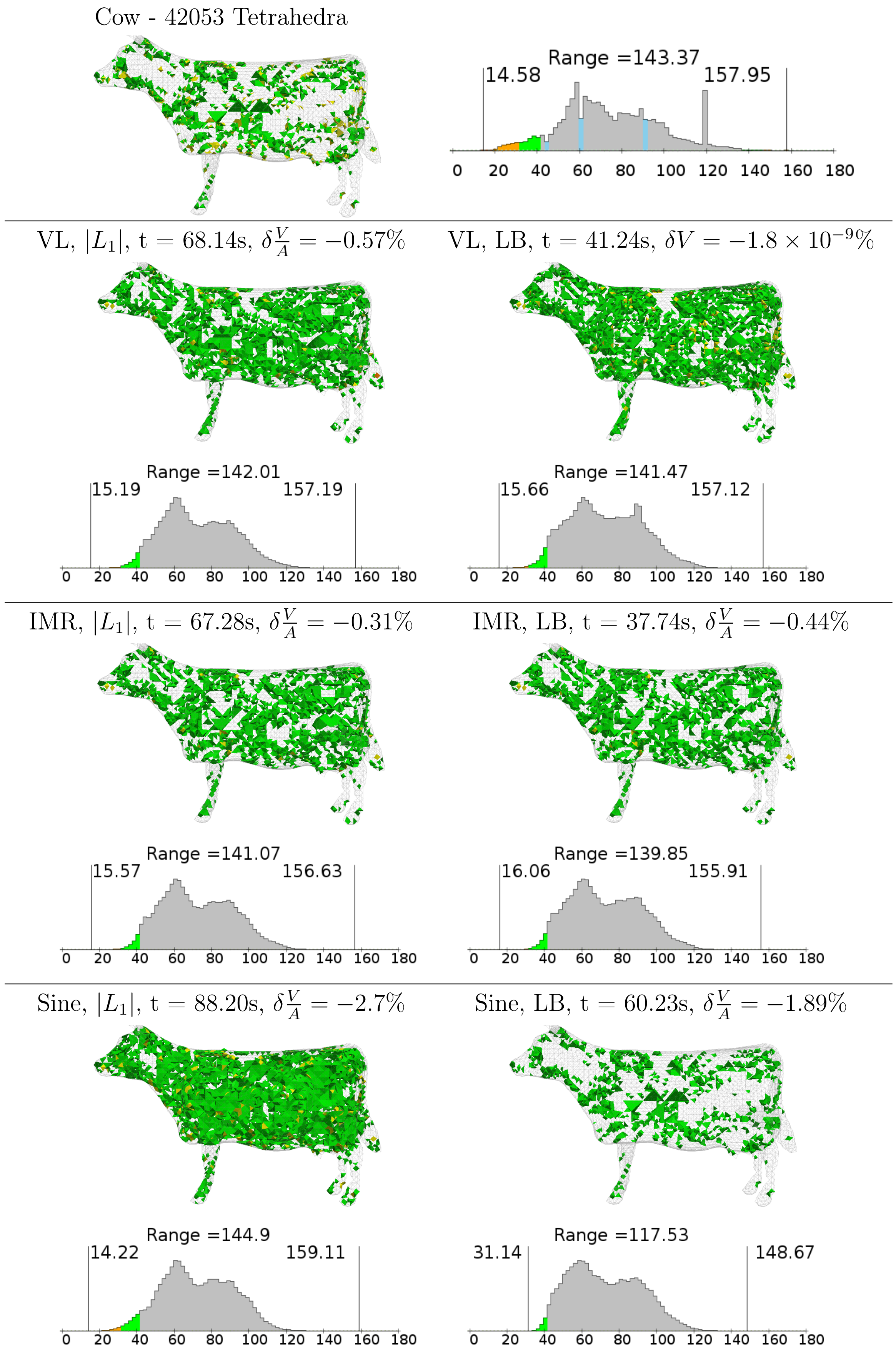


Figure 6.4: Constrained mesh optimisation results for "Cow"

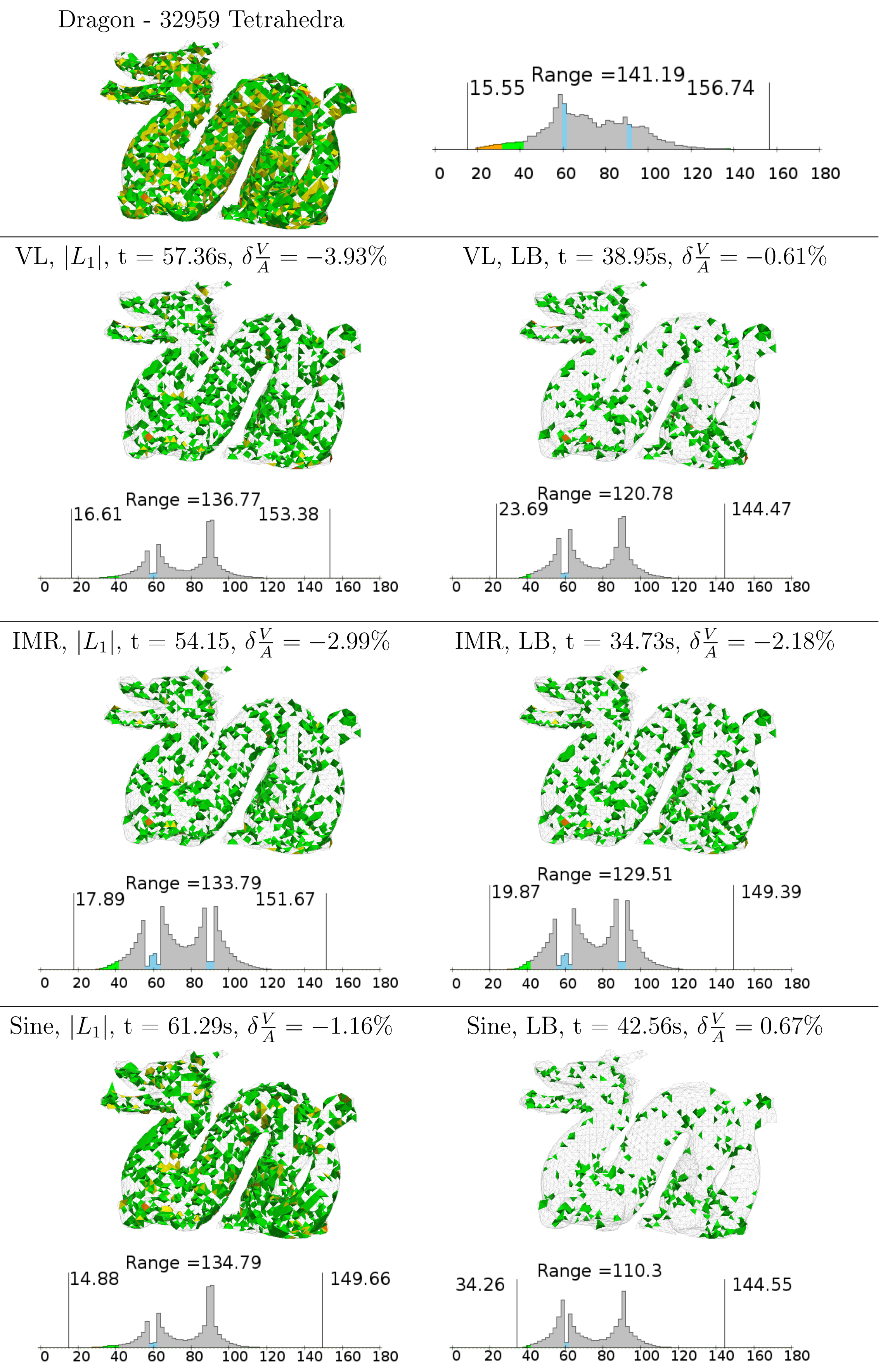


Figure 6.5: Constrained mesh optimisation results for "Dragon"

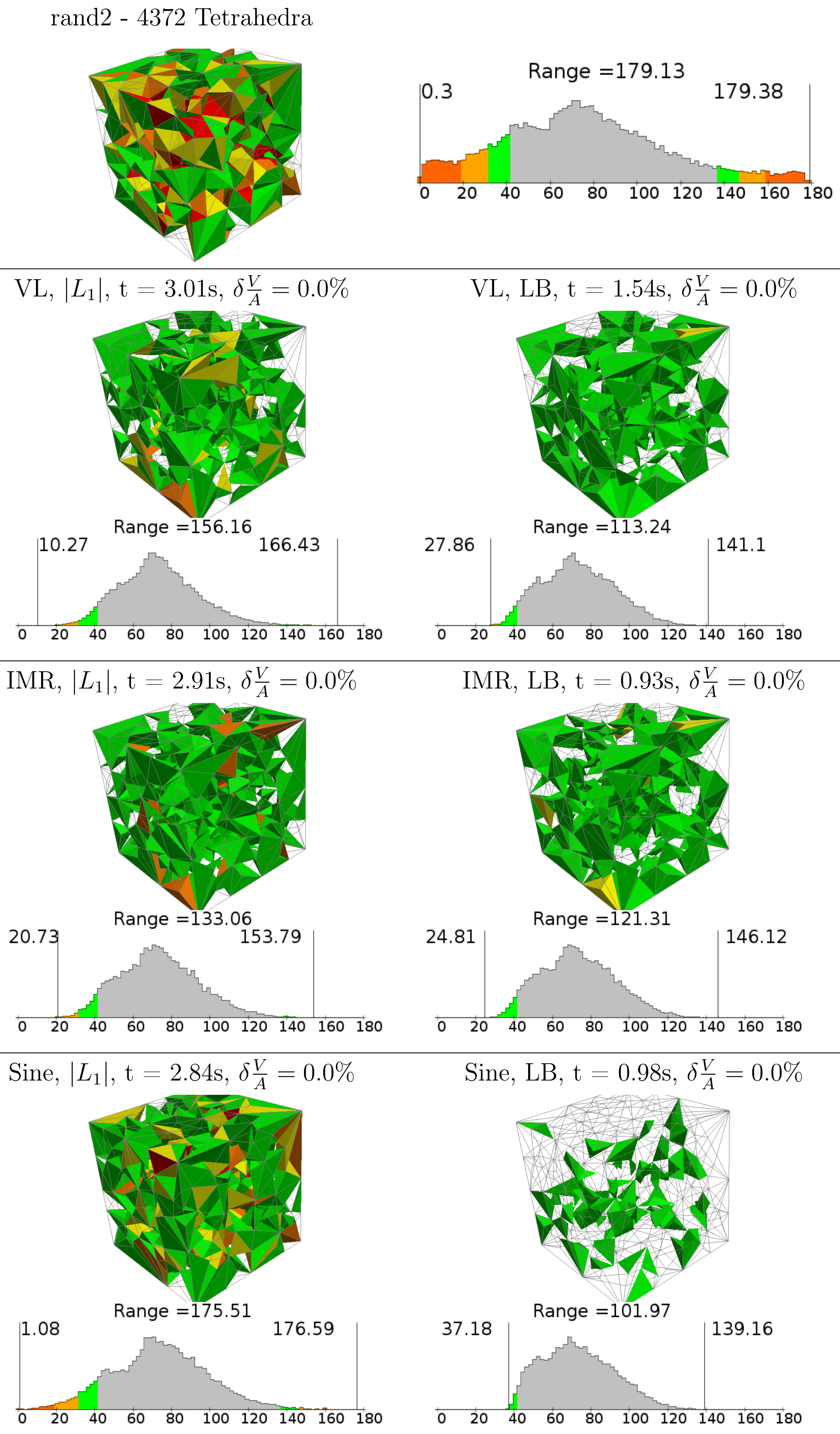
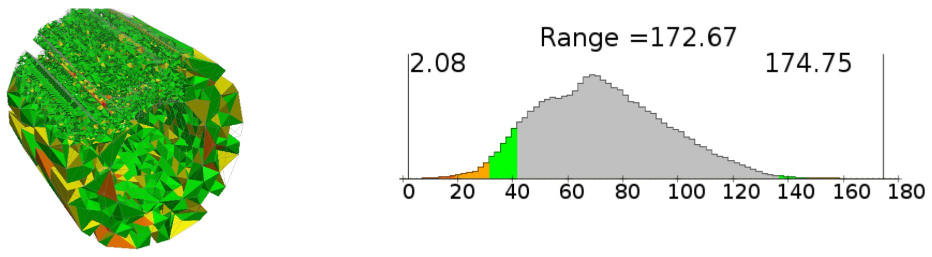


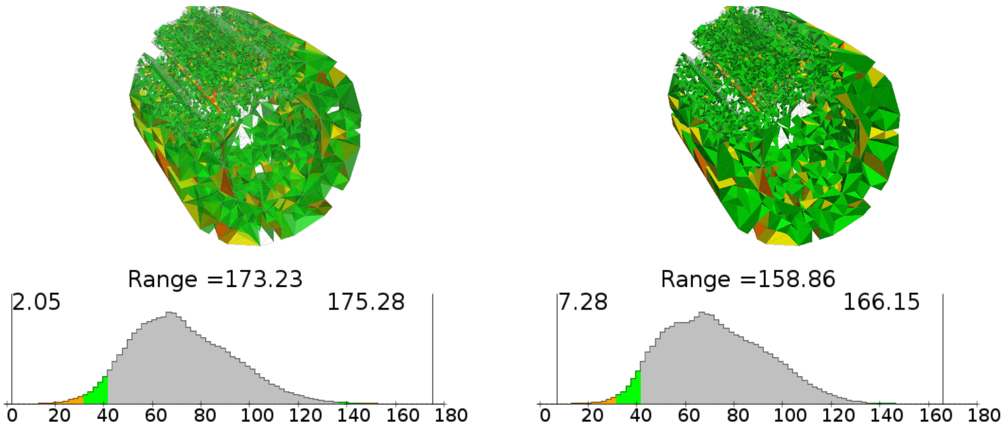
Figure 6.6: Constrained mesh optimisation results for "rand2"

Graphite Brick - 100781 Tetrahedra



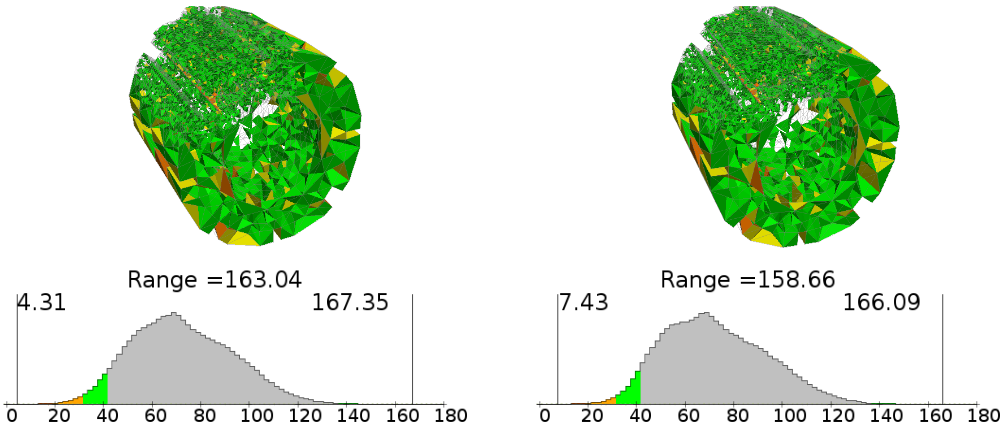
VL, $|L_1|$, t $=$ 180.57s, $\delta\frac{V}{A} = 1.31\%$

VL, LB, t $=$ 67.13s, $\delta\frac{V}{A} = 0.87\%$



IMR, $|L_1|$, t $=$ 194.34s, $\delta\frac{V}{A} = -1.99\%$

IMR, LB, t $=$ 71.56s, $\delta\frac{V}{A} = 0.54\%$



Sine, $|L_1|$, t $=$ 210.24s, $\delta\frac{V}{A} = 1.67\%$

Sine, LB, t $=$ 81.94s, $\delta\frac{V}{A} = 1.34\%$

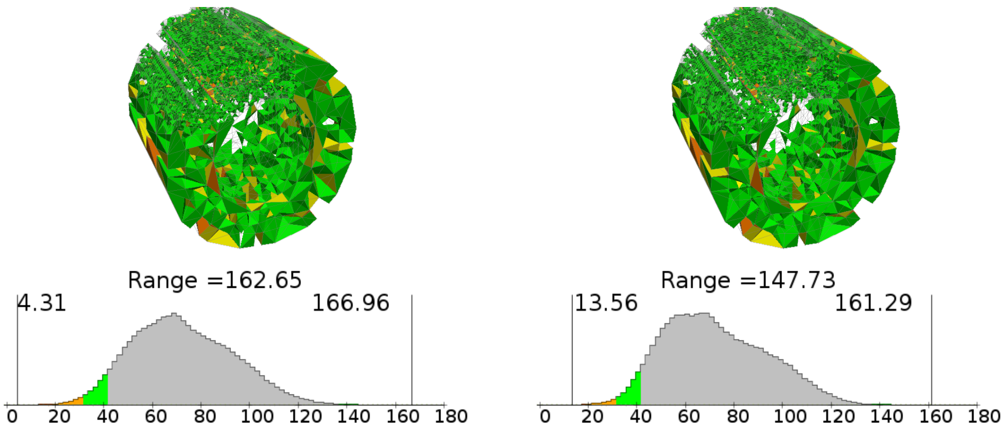


Figure 6.7: Constrained mesh optimisation results for "Graphite Brick"

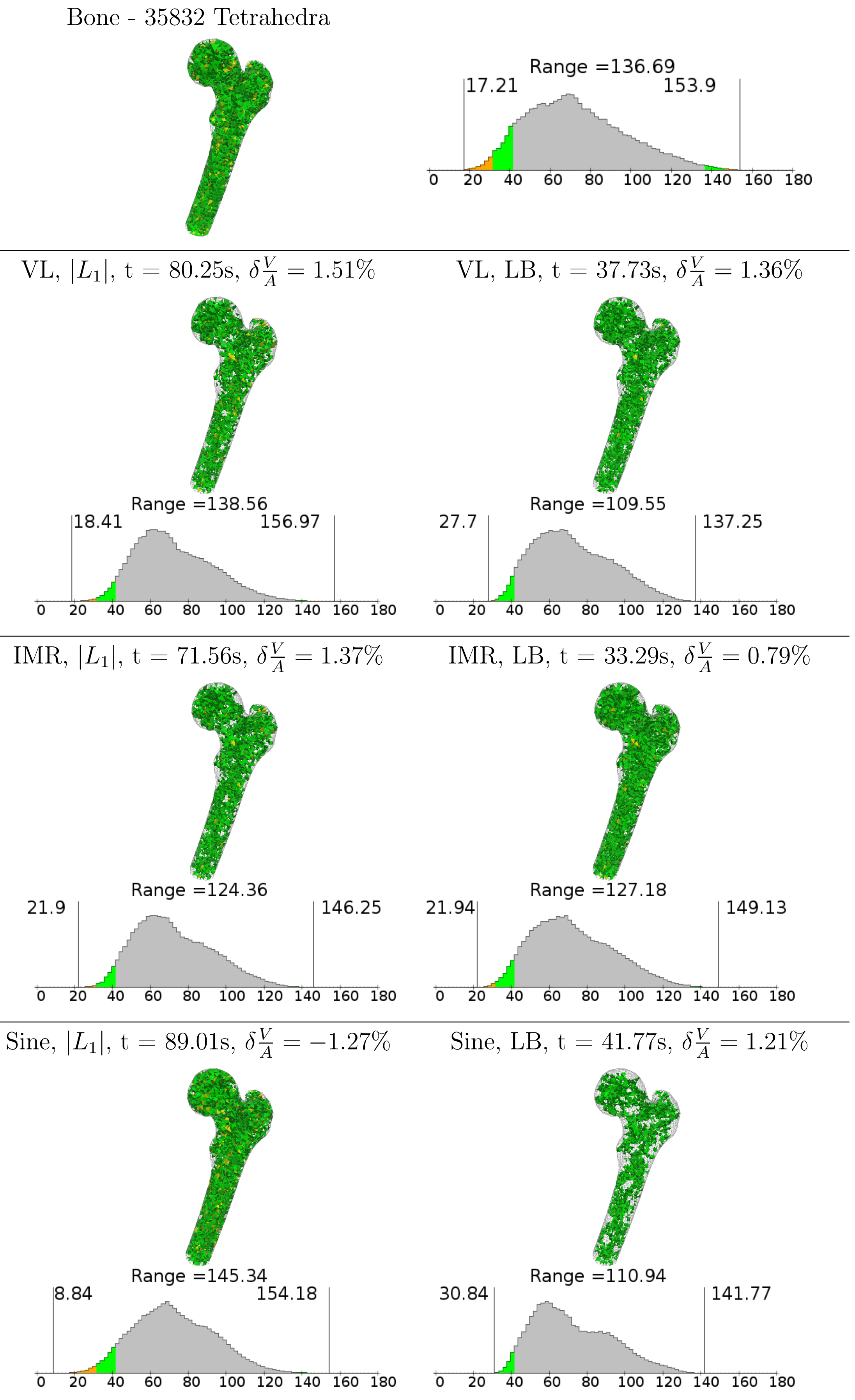
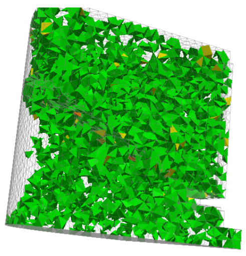
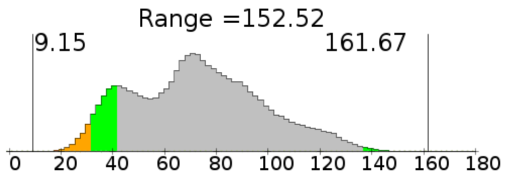
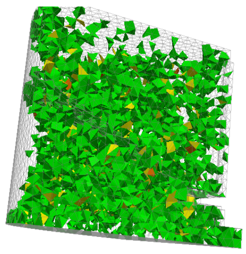
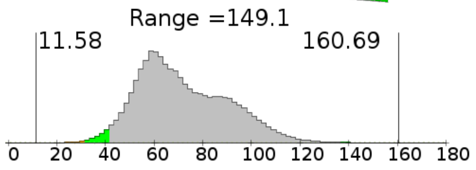


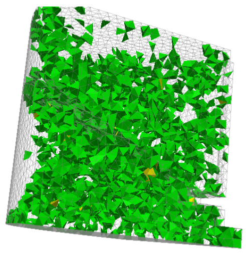
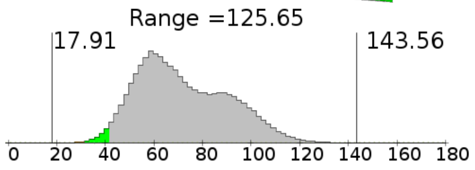
Figure 6.8: Constrained mesh optimisation results for "Bone"

Pullout Test - 73864 Tetrahedra

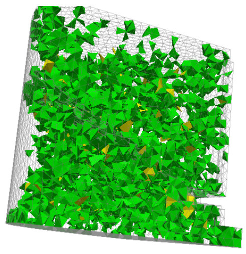
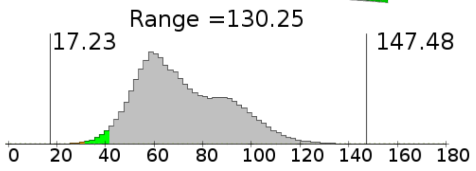



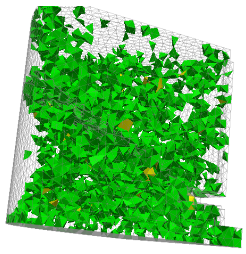
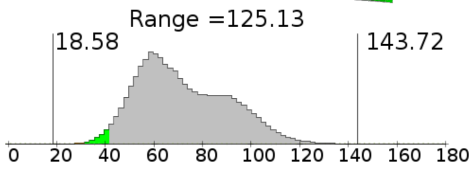
VL, $|L_1|$, t $=$ 81.25s, $\delta\frac{V}{A} = -0.85\%$ VL, LB, t $=$ 56.38s, $\delta\frac{V}{A} = -1.12\%$

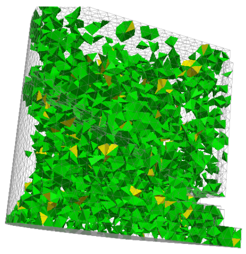
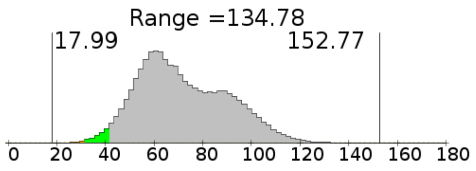



IMR, $|L_1|$, t $=$ 68.08s, $\delta\frac{V}{A} = 1.89\%$ IMR, LB, t $=$ 49.74s, $\delta\frac{V}{A} = 1.52\%$

Sine, $|L_1|$, t $=$ 96.83s, $\delta\frac{V}{A} = -0.57\%$ Sine, LB, t $=$ 60.30s, $\delta\frac{V}{A} = -1.11\%$

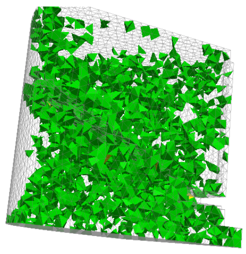
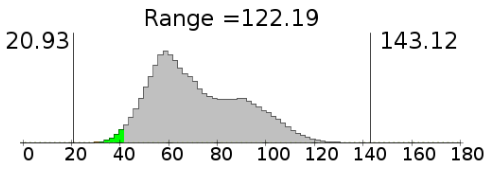



Figure 6.9: Constrained mesh optimisation results for "Pullout Test"

6.3 Discussion

Examination of the optimised meshes shows that the algorithm described in the previous chapter for constraining surface nodes, based only on information obtained from the initial mesh, is very effective. Despite the many complex surfaces present in these meshes, including both surfaces used in computer graphics and those formed by the simulation of physical phenomena, the volume to surface area ratio, $\delta\frac{V}{A}$, was successfully conserved throughout the optimisation process. The single greatest change observed in the volume to surface area ratio is 3.93% in the case of "Dragon" optimised using the $|L_1|$ objective function and the VL quality measure. In most cases, the change is much less than this. The second significant observation is that the process of constraining surface nodes is very expensive. All of surfaces of "Square" and "rand2" are planar and this is reflected in the time taken to optimise them. The time taken to optimise each of the other meshes is significantly greater than that taken in the unconstrained case.

6.3.1 2D Results

Examination of the three 2D meshes optimised using the constrained optimisation algorithm shows that, despite the constraints applied to the surface nodes, all meshes were successfully optimised. The greatest change in the volume to surface area ratio for all 2D meshes is 1.32%. This indicates that this algorithm is very effective in two-dimensions.

Optimisation using the $|L_1|$ objective function combined with each quality measure shows that the IMR quality measure was the most effective at eliminating small angles, producing meshes with considerably greater smallest angles than either the sine or the AL quality measures. Similar to what was observed with the unconstrained optimisation results in Chapter 4, the sine quality measure actually reduced the smallest angle in "Oscillating Droplet". In terms of elimination of large angles, the sine quality measure achieved the best results for all three meshes. For both "Square" and "Hanging Droplet" the sine quality measure was significantly more effective than the other two quality measures at eliminating large angles. Interestingly, there is very little difference in the time taken to optimise both "Hanging Droplet" and "Oscillating Droplet" by all three quality measures. However, the use of the IMR quality measure was significantly quicker when applied to "Square".

The effectiveness and efficiency of the log-barrier objective function observed with the unconstrained optimisation results is again realised with constrained mesh optimisation. In every case, the meshes optimised using the log-barrier objective function are of higher quality than those optimised using the $|L_1|$ objective function. The optimisation process was also completed in much less time, compared with the time taken to optimise the meshes using the $|L_1|$ objective function. The sine quality measure was the least efficient of the three quality measures. The optimisation process using the sine quality measure required considerably more time than that using the other two quality measures. However, in terms of small angle elimination, the sine quality measure was the most effective for all three 2D meshes, albeit only slightly so for "Hanging Droplet" and "Oscillating Droplet". The effectiveness of the sine quality measures at eliminating small angles is not repeated with large angles, with the largest angle present on the optimised meshes being significantly larger than the other two quality measures, save for "Square", where it is comparable. For "Hanging Droplet" the IMR quality measure performs best and for "Oscillating Droplet" the AL quality measure performs best.

6.3.2 3D Results

All six of the 3D meshes were successfully optimised, despite the challenges posed by the complicated shapes of the domains and the poor quality of the initial meshes. Careful scrutiny of the meshes optimised via a combination of the $|L_1|$ objective function and each of the three 3D quality measures, shows that the IMR quality measure was the most effective at eliminating small angles. The IMR quality measure was also very effective at eliminating large angles, except for "Dragon" and "Graphite Brick", where the sine quality measure was best. Note that the time taken to optimise all meshes, except "rand2", is significantly greater than that taken in the unconstrained case. This clearly demonstrates the expense associated with this algorithm. The time required by the optimisation process using all three quality measures to optimise "rand2" is approximately equal to that required by the unconstrained case.

As was the case with all of the 2D results and the unconstrained results, the use of the log-barrier objective function produced meshes of much higher quality than those produced using the $|L_1|$ objective function and in significantly less time. Interestingly the effectiveness of the sine quality measure combined with the log-barrier objective function in 2D is mirrored in 3D. By a clear margin, this quality measure was the most effective at eliminating small angles. The sine quality measure was also the most effective at eliminating large angles, although not as remarkable as its performance in eliminating small angles. The spire tetrahedra which were formed by using the sine quality measure with unconstrained optimisation were not observed in any of the 3D meshes optimised using the sine quality measure. The results obtained indicate that the IMR quality measure in 3D is not nearly as effective as it is in 2D. For those whom efficiency is a great concern, the use of VL quality measure combined with the log-barrier objective function is recommended. If the time taken by the optimisation process is not as important as the quality of the optimised mesh, the sine quality measure is the obvious choice.

6.4 Conclusions

The results presented in the previous section demonstrate the effectiveness of the shape and volume preservation techniques developed in Chapter 5. The largest change in the volume-surface area ratio observed was for "Dragon" optimised using the VL quality measure combined with the $|L_1|$ objective function with a change of -3.93%. Interestingly, the volume was completely preserved in every case, the largest change in volume observed was less than one-thousandth of a percent. This is a very important for problems involving fluids as preservation of volume is crucial to obtaining accurate results. The ability to preserve the domain shape and volume has enabled the optimisation of meshes which previously could not be. In many of these complex meshes, the worst elements are located on the surface of the domain, often with all nodes on the surface. This means that these nodes are defining the surface and any movement of them will results in unacceptable changes to the mesh shape and volume.

It is worth noting that the mesh optimisation routines included in the release version of Mesquite could not improve these meshes due to all the nodes of the worst elements being on the mesh surface. Every mesh apart from "Square" and "rand2", whose surfaces are entirely planar, could not have been optimised as effectively by any other software. For example, Stellar using surface quadrics to preserve the domain shape, can alter the volume to surface ratio by up to 15% in some cases. Although the resulting mesh in this case may visually be very similar, mathematically they are very different. Whilst such a technique is very suited to computer graphics and animation, it is clear that it is not applicable to scientific applications.

To demonstrate the effectiveness of the surface optimisation algorithm, the original mesh and optimised mesh of the crack surface of "Graphite Brick" are overlain on

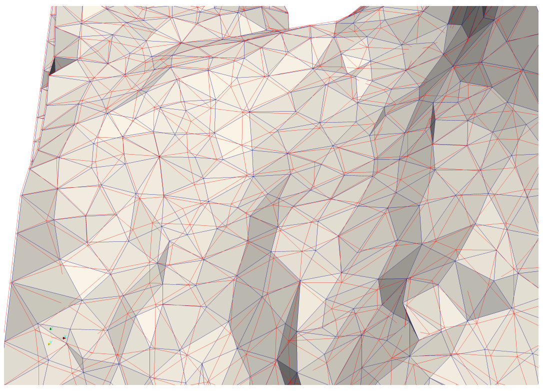


Figure 6.10: Effectiveness of surface mesh improvement: the red mesh is optimised and the blue mesh is the original mesh

each other so that it is possible to see the movement of surface nodes, Figure 6.10. Although large nodal displacements are observed in some places, the overall crack shape is preserved. The results obtained from using this complex mesh demonstrate how effective this the surface constraint algorithm is. Along with the preservation of the shape of the domain, the volume is also completely preserved, meaning this technique may be applied to many complex simulations. For the first time, complex mesh surfaces may now be optimised whilst preserving both shape and volume using only the mesh to define the surface.

Chapter 7

Mesh Optimisation as Part of the Finite Element Solution Process

7.1 Introduction

Many physical processes involve domains which change with time. The simulation of crack propagation in mechanics of materials and surface tension in fluid dynamics are two examples where large deformations and evolving domains can occur. In order to accurately model the physical processes at play, the mesh must adapt to the domain. The problems associated with poor quality meshes are explained in Chapter 2 as are the consequences of interpolating data from a poor quality mesh to an updated mesh of the same domain. In this chapter a technique is developed to adapt a mesh to a deforming domain, thus minimising the numerical errors induced by transferring data between meshes. This is achieved by considering mesh quality optimisation as an integral part of the overall Finite Element (FE) algorithm.

In Chapter 3, the meshes used to verify the mesh optimisation algorithms developed as part of this research were introduced. Some of these were obtained from FE simulations which suffered from many issues due to poor quality meshes. It is these problems from which the motivation for this research was drawn. For example, "Hanging Droplet" and "Oscillating Droplet" were obtained from a FE simulation of surface tension in a micro-fluid droplet. "Graphite Brick" and "Pullout Test" were obtained from the intermediate stages of FE simulations of crack propagation in graphite and concrete. The meshes in these simulations must adapt to the complex and constantly evolving domains, thus accommodating both large deformations in the case of both droplet meshes and crack

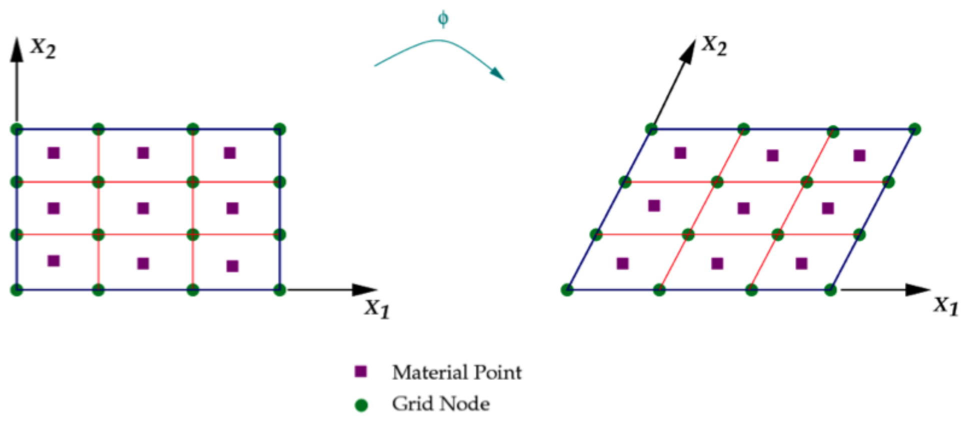


Figure 7.1: Lagrangian description of motion. Adapted from [42]

surface evolution in the case of "Graphite Brick" and "Pullout Test".

Although mesh optimisation and FE analysis are very closely related, both are often treated as separate processes independent of one other. A novel aspect of this thesis is the development of an integrated approach where mesh optimisation is undertaken together with the FE analysis of the non-linear physical problem.

7.2 Mesh adaption techniques for large deformations

There are two standard formulations for FE simulations, these are Lagrangian and Eulerian formulations. In a Lagrangian formulation each mesh vertex tracks a particular material point, thereby the mesh deforms with the domain, Figure 7.1. This implies that the motion of a particular material particle is tracked through time and its behaviour is always known. This method enables accurate surface tracking, such as crack fronts and fluid free surfaces, throughout the deformation process. This method also facilitates the modelling of materials with history-dependent constitutive relations [3], for example a plasticity multiplier. However, elements can become very distorted when a deforming domain is modelled using this configuration, thus requiring frequent re-meshing, either after each time-step or when the quality of the domain deteriorates past a certain point. Re-meshing involves the transfer of data from a poor quality, distorted mesh to the new mesh. The numerical errors, discussed in Chapter 2, associated with this transfer of data may be so punitive as to render the results of any simulation of a large deformation which uses this method meaningless.

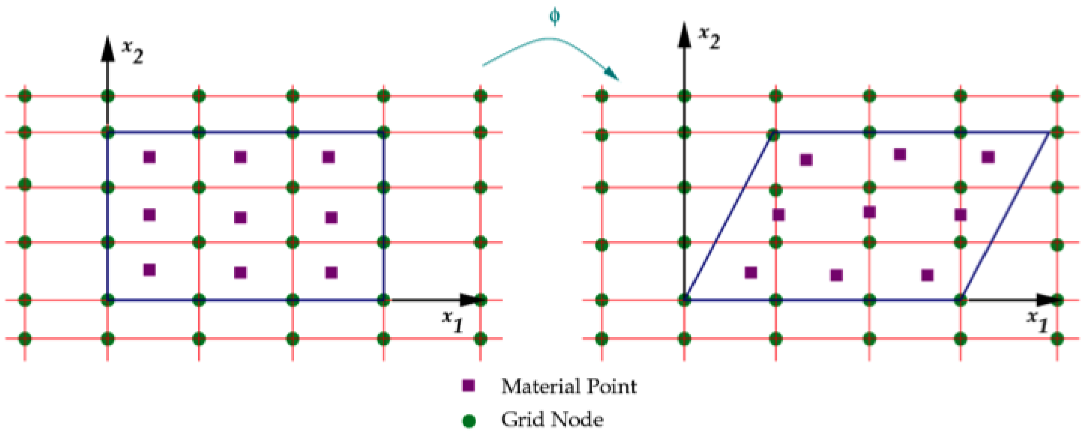


Figure 7.2: Eulerian description of motion. Adapted from [42]

Another method for dealing with large deformations and which is often used in Computational Fluid Dynamics (CFD) is an Eulerian formulation. In this formulation, the mesh vertices remain stationary as the continuum evolves and material properties are calculated at each mesh node. Mesh quality is maintained during large deformations but accurate surface tracking is not possible. This is illustrated in Figure 7.2. In many simulations, the interesting and important physical processes occur at or near the boundary and such a configuration would not accurately capture these physical phenomena.

Dheeravongkit [43] proposes an alternative form of a Lagrangian configuration whereby the initial mesh is pre-deformed so that it has approximately the opposite shape of the final shape that it will deform to. Therefore the analysis starts with a non-optimal mesh, which improves during the deformation process and then finally degrades again to a lesser extent than it would have without the initial deformation. However, this method is not compatible with the class of problem being considered here as knowledge of the deformed shape must be known *a priori*. In the case of the fluid droplet studied later in this chapter, the domain assumes many different shapes throughout the deformation process and the use of this method would require finding a mesh which can adequately represent each of these.

In order to address the short comings of the above methods, a new method for defining the relationship between the domain and the mesh called the Arbitrary Lagrangian Eulerian (ALE) formulation [3] was developed. This method aims to combine the best features of both Lagrangian and Eulerian formulations, whilst avoiding their respective drawbacks. This means that mesh vertices may move with the continuum in a Lagrangian manner or remain stationary in an Eulerian manner or move in some arbitrarily specified way [3]. The use of an ALE formulation allows the efficient and accurate simulation of large deformations. This method is discussed in this chapter, with particular reference to problems involving fluids. This freedom to arbitrarily move mesh nodes allows for much more accurate modelling of complex surfaces with fewer meshing issues than the other two formulations. However, the use of this method requires solving for additional degrees of freedom, the mesh velocity. The mesh velocity is used to determine the updated nodal positions.

In an ALE formulation, three configurations are defined: a reference configuration (χ), a material configuration (X) and a spatial configuration (x), Figure 7.3. Each mesh point relates to a material point and each material point refers to a point in space. The user is free to choose the reference configuration independent of the material configuration [44]. Typically one of two approaches is used, either taking the initial configuration (Total ALE - T-ALE) or taking the configuration at the end of the previous time-step as the reference configuration (Updated ALE - U-ALE). In a T-ALE formulation, all derivatives are calculated with respect to the Lagrangian coordinates in contrast with an U-ALE formulation where all derivatives are calculated with respect to Eulerian coordinates [45] which are updated as the analysis progresses. The updated approach is of more interest here as it is more suited to problems involving large strains and flows [46]. In the updated approach, a means of calculating the spatial configuration from the reference configuration is required. Mesh velocities are solved for during the previous step and the reference configuration is updated to reflect these. Therefore at the start of each time-step the reference configuration, the spatial configuration and the material configuration coincide, $x \equiv X \equiv \chi$.

7.2.1 ALE mesh update procedures

The use of an ALE formulation gives great flexibility in determining the location of mesh nodes as a domain deforms. In order to calculate the positions of the nodes, mesh velocities must be calculated at each time-step or load step of a calculation. The method chosen to determine the updated positions of nodes greatly impacts the success of the simulation [3]. In order to update the mesh at every iteration, a method of calculating the mesh velocities and therefore the new nodal positions is required. Several methods have been proposed to achieve this with ALE methods. The approach taken by [47] involves breaking the process up into three steps. Firstly, a Lagrangian step is taken in which the mesh nodes move in a Lagrangian manner: they track the material point. Next, there is a rezone step in which the nodes move in order to

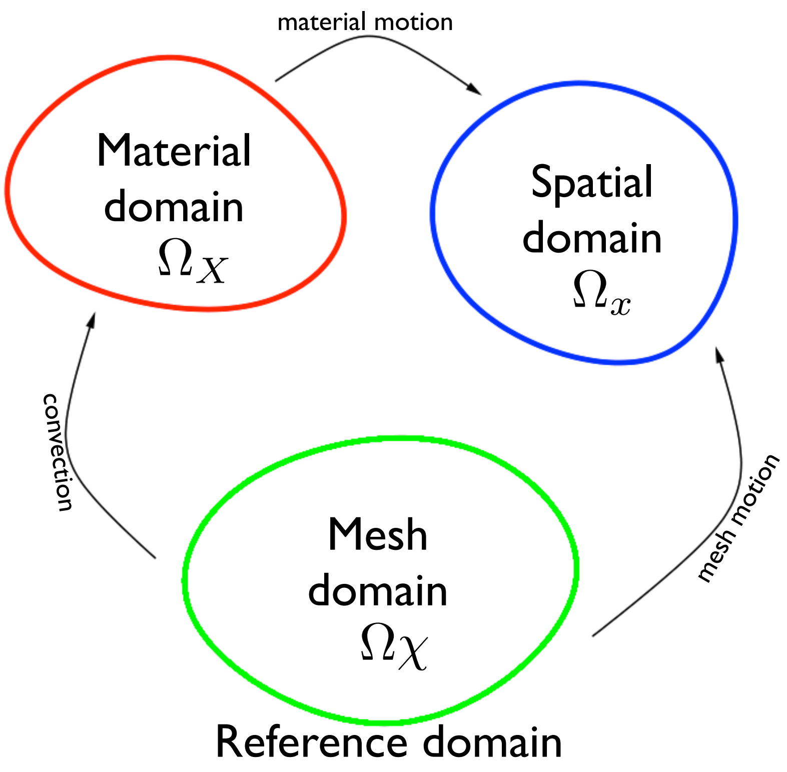

material motion
Material domain Ω_X
Spatial domain Ω_x
convection
Mesh domain Ω_χ
mesh motion
Reference domain

Figure 7.3: ALE Domain Mapping. Adapted from [42]

improve mesh quality and then a remapping step where the solution is transferred from the old mesh to the new one. Several iterations of this may be performed to ensure physical equilibrium is reached [44]. The disadvantage of this method is that errors are generated by the interpolation of the solution from the old mesh to the new mesh as well the introduction of non-linearities due to the change in the mesh as described in Chapter 2. This is referred to as staggered ALE or S-ALE [44].

An alternative form of ALE is the monolithic form (M-ALE) whereby the physical equations and the mesh motion equations are coupled and therefore solved simultaneously. This results in a much larger matrix as the size of the system of equations is proportional to the square of the number of degrees of freedom. However, if the equations are properly linearised, quadratic convergence can be maintained [44]. This method overcomes the disadvantages of the staggered approach as there is no need to interpolate the solution from one mesh to another.

Several examples have been found in the literature of mesh optimisation being used to calculate the updated mesh nodal positions when using S-ALE, [48] and [47], however, no examples have been found of mesh optimisation being used in a M-ALE simulation. Laplacian smoothing has been widely used to calculate the updated nodal positions in M-ALE simulations. Laplacian smoothing involves moving a node to the average of its connected neighbour's positions. The idea behind Laplacian smoothing is that the nodes of high quality meshes are separated by equal distances. However, the inverse is not true - good quality meshes need not consist of equidistant nodes. As a result, Laplacian smoothing can induce excessive and unnecessary nodal displacements in initially good meshes. Every nodal displacement generates non-linearities which negatively effects the rate of convergence and the efficiency of a calculation. Laplacian smoothing has been shown to be somewhat effective in two-dimensions when the domain is convex, however, if the domain is non-convex, elements may easily become inverted [3]. Techniques which aim to prevent the inversion of elements exist but these are either computationally expensive or require the derivation of terms particular to each geometry [3]. Three dimensional Laplacian smoothing is only effective in the most simple of cases. Another disadvantage of Laplacian smoothing is that the user has no control over nodal density meaning that local mesh refinement is not possible as the nodes will disperse equally through the mesh.

More sophisticated mesh smoothing algorithms can be used once the mesh topology is preserved [3] and such methods have been demonstrated in the S-ALE approach. Giulani [48] developed a method which aims to minimise both the squeeze and distortion of elements in the mesh and Sarrate [49] improved on this. Knupp [47] uses Jacobian based mesh quality measures to calculate the mesh velocities for a S-ALE model. In this next section, a new method is proposed which uses mesh quality optimisation to calculate the mesh velocities as part of an M-ALE simulation. This method is then applied to a simulation of a micro-fluid droplet.

7.3 Calculating ALE mesh velocities using mesh quality optimisation

In this section, the development of a method of determining the mesh velocities in an M-ALE simulation using mesh quality optimisation is discussed. This involves coupling the physical equations governing the evolution of a micro-fluid droplet towards its equilibrium position and the mesh quality optimisation equations so that the updated mesh positions will be compatible with physical equilibrium. In order to successfully couple both sets of equations, an understanding of the governing equations is required, as well as a detailed knowledge of the general implementation of the simulation.

7.3.1 Overview

The medical industry conducts vast amounts of research into the diagnosis of diseases. Currently, the diagnosis of diseases via analysis of blood samples involves manually taking a sample using a syringe, then loading it into a centrifuge and spinning it in order to separate the sample into its constituent components so that these may be tested. This process is both expensive and time-consuming. In many parts of the developing world, demand for such tests is high whereas access to the required equipment is limited. Therefore, there is a great interest in the development of more efficient methods which are both cheap and suitable for use in the field. One proposed solution to this problem is to use a lab on a chip. This consists of a microchip onto which the blood sample is placed. The sample is then subject to external excitation in the form of surface acoustic waves. Such devices are based on the premise that the external excitation of a blood sample will cause the blood droplet to separate into its constituent parts, thus enabling the testing of samples in field conditions and eliminating the need for laboratory analysis. The development of this technology requires a thorough understanding of the physical processes which cause the droplet to separate as well as the actual separation process. The application of computational modelling to the study of these phenomena is ideal for gaining insight into the behaviour of the blood droplet at this scale. As is the case in many areas of engineering, the use of virtual prototypes could greatly accelerate the development process as well significantly reducing the cost.

A computational model of a micro-fluid droplet was developed by Mackenzie [42] which aims to model the evolution and separation of a micro-fluid droplet subject to external excitation. The first step in this process is to develop a computational model of the evolution of a micro-fluid droplet from a non-equilibrium initial position to its final equilibrium position. When this is successful, the surface acoustic waves which cause the droplet to separate into its constituent parts will be added to the model. This computational model is based on a M-ALE formulation and in its current form uses Laplacian smoothing to calculate the change in mesh velocity. In the following sections, this computational model and the problems associated with the use of Laplacian smoothing are described in detail. The process of replacing Laplacian smoothing with mesh optimisation and the results of this is also described.

7.3.2 Problems associated with Laplacian smoothing

In the current implementation of this computational framework, the change in mesh velocity is calculated using Laplacian smoothing. The motivation for substituting Laplacian smoothing for mesh optimisation fall into two categories. Firstly many issues have been tracked to poor quality mesh. Analyses have been terminating prematurely due to issues with meshes and the domain was being re-meshed too frequently. A certain degree of re-meshing will always be required due to the large deformations observed, however, this should be limited as much as possible. Volume loss was often observed and it is desirable to minimise this as much as possible.

Secondly, the success of the simulation was found to be heavily dependent on a user defined, artificial mesh viscosity parameter to limit the calculated mesh velocities. This parameter is an analogue to the physical fluid viscosity. The greater the value of this parameter, the more difficult it is for a mesh node to move. This parameter is problem dependent and the only way to find a suitable value is by fine tuning. It is also very difficult to choose this parameter correctly in situations where one part of the fluid is moving quickly and the other slowly, thus large amounts of user intervention are required. Experience has shown that if an incorrect value of this parameter is chosen, the analysis may terminate immediately or it may continue for some time before terminating prematurely. This use of this parameter is therefore both tedious, time-consuming and an inefficient use of computing resources. The next stages of this development process involves the 3D implementation of this model. Therefore techniques which will translate to three dimensions are required and for this reason it is necessary to replace Laplacian smoothing with a more effective, reliable and user friendly algorithm.

7.3.3 Deformation of the fluid droplet

The initial testing stages of this project involved laboratory experiments to gain an understanding of the behaviour of a micro-fluid droplet. This involved placing a droplet of water on a speaker emitting waves of various frequencies. Water was used due to the strict regulations governing the use of blood in laboratory testing. A high speed camera captured the evolution of the shape of the fluid droplet, Figure 7.4. It is clear from these images that the droplet undergoes large deformations, and thus a mesh of the droplet must be able to adapt to accommodate these deformations. The first step in this process is to understand the physical processes governing the behaviour of the

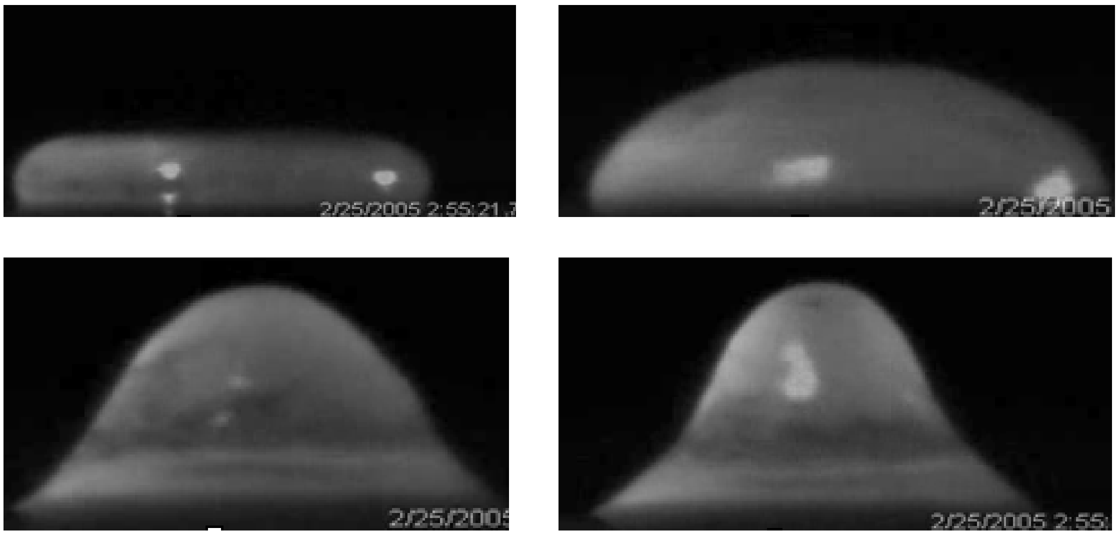


Figure 7.4: Evolution in the shape of a fluid droplet subject to external excitation, [50]

droplet.

7.3.4 The governing equations

In the following sections, the physics of this problem are described as an appreciation and understanding of this is essential to the successful implementation of mesh quality optimisation into the existing computational model.

7.3.4.1 The Navier-Stokes equations

The Navier-Stokes equations for incompressible fluids are a set partial differential equations which describe fluid motion. They are derived by applying Newton's Second law to fluid motion whilst accounting for conservation of momentum, mass and energy. Fluid stress is assumed to be the sum of a viscous term and a pressure term. There are many derivations of the Navier Stokes equations available, for example Acheson [51] and Farside [52].

The problem under consideration in this section is concerned with fluid motion at the micro-scale. For this derivation it is assumed that the fluid is a continuum and that there are no discrete particles in the fluid. Following the *continuum hypothesis* the properties of the bulk of the fluid may be applied to any point in the fluid. In the following sections, two important concepts for the understanding of the Navier-Stokes equations are introduced: the material derivative and Reynold's transport theorem. Conversation of mass and momentum will then be discussed.

The material derivative:

The material derivative is the derivative of a property of the material, e.g. temperature or velocity, under consideration with respect to a moving coordinate system [53]. The property under consideration generally depends on time, t and its position of the fluid element at that time.

$$f(t) = f(x, y, t) = f(\mathbf{s}, t) \tag{7.1}$$

where $\mathbf{s}$ is a position vector. At time t the fluid particle has coordinates $x(t), y(t)$. In the time interval δt, the element moves from (x, y) to $(x + \delta x, y + \delta y)$. As f is a function of x, y, t, there will thus be a corresponding change in f, denoted as δf. Therefore:

$$\delta f = \frac{\partial f}{\partial t}\delta t + \frac{\partial f}{\partial x}\delta x + \frac{\partial f}{\partial y}\delta y \tag{7.2}$$

The observed rate of change is therefore:

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt} \tag{7.3}$$

Velocity is defined as the rate of change of position s:

$$\frac{d\mathbf{s}}{dt} = (u, v) = \mathbf{u} = (\frac{dx}{dt}, \frac{dy}{dt}) \tag{7.4}$$

and

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + u\frac{\partial f}{\partial x} + v\frac{\partial f}{\partial y} = \frac{\partial f}{\partial t} + \mathbf{u}\nabla \cdot f \tag{7.5}$$

In fluid mechanics the rate of change of the property of a fluid element is normally denoted as $\frac{D}{Dt}$. Therefore the material derivative of the property f of a fluid element is defined as:

$$\frac{Df}{Dt} = \frac{\partial f}{\partial t} + \mathbf{u}\nabla \cdot f \tag{7.6}$$

Reynold's transport theorem:

Reynold's transport theorem is a conservation law concerning intensive properties of a control volume. An intensive property is a physical property of a system which is independent of the system size and the amount of material present [42]. It states that the rate of change of an intensive property **S** defined over a control volume Ω is equal to the sum of the loss or gain of the property through the boundaries of the control volume plus what is created or destroyed by sources or sinks within the control volume [42]. Mathematically it is stated as follows:

$$\frac{\partial}{\partial t}\int_{\Omega}\mathbf{s}dV = -\int_{\delta\Omega}\mathbf{s}\mathbf{v}\cdot\mathbf{n}dA - \int_{\Omega}QdV \tag{7.7}$$

where $\delta\Omega$ is the bounding surface of Ω, $\mathbf{n}$ is the unit outward pointing normal of $\delta\Omega$ and Q is any sources or sinks within Ω. Using the divergence theorem, the area integral may be changed into a volume integral. The divergence theorem states that the density within a region of space may only change by flow into or away from the region through its boundary [39]. Therefore:

$$\int_{\delta\Omega}\mathbf{s}\mathbf{v}\cdot\mathbf{n}dA = \int_{\Omega}\nabla\cdot(\mathbf{s}\mathbf{v})dV \tag{7.8}$$

Therefore:

$$\frac{\partial}{\partial t}\int_{\Omega}\mathbf{s}dV = -\int_{\Omega}\nabla\cdot(\mathbf{s}\mathbf{v})dV - \int_{\Omega}QdV \tag{7.9}$$

The left-hand side of equation 7.9 may be modified using Leibniz's rule which is a formula for differentiation of a definite integral whose limits are functions of the differential variable [54]. Formally it states that:

$$\frac{\partial}{\partial t}\int_{a(x)}^{b(x)} f(x,t)dx = \int_{a(x)}^{b(x)}\frac{\partial f}{\partial x}(x,t)dx \tag{7.10}$$

Applying this to equation 7.9 results in:

$$\int_{\Omega}\frac{\partial\mathbf{s}}{\partial t}dV = -\int_{\Omega}\nabla\cdot(\mathbf{s}\mathbf{v})dV - \int_{\Omega}QdV \tag{7.11}$$

Therefore:

$$\int_{\Omega}\left(\frac{\partial\mathbf{s}}{\partial t}dV + \nabla\cdot(\mathbf{s}\mathbf{v}) + Q\right)dV = 0 \tag{7.12}$$

However, this result is true irrespective of size, shape or location of volume V which is only possible if this relation holds at every point in the fluid [52]. Therefore

$$\frac{\partial\mathbf{s}}{\partial t} + \nabla\cdot(\mathbf{s}\mathbf{v}) + Q = 0 \tag{7.13}$$

Conservation of mass

Assuming that there are no sources or sinks of momentum within the control volume ($Q = 0$) and using the density of the fluid as the intensive property, equation 7.13 may be used to give an expression for conservation of mass:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = \frac{\partial \rho}{\partial t} + tr\left(\frac{\partial \mathbf{v}}{\partial \mathbf{x}}\right) = 0 \tag{7.14}$$

Conservation of momentum

Momentum per unit volume is the product of density and fluid velocity. Substituting this into equation 7.13 gives:

$$\frac{\partial}{\partial t}\rho \mathbf{v} + \nabla \cdot (\rho \mathbf{v}\mathbf{v}) + Q = 0 \tag{7.15}$$

A body force $\mathbf{b}$, which may be considered as a source or sink of momentum per unit volume is introduced. Equation 7.15 may be expanded using the chain rule of differentiation giving:

$$\left(\frac{\partial \rho}{\partial t}\mathbf{v} + \frac{\partial \mathbf{v}}{\partial t}\rho\right) + (\mathbf{v}\mathbf{v} \cdot \nabla \rho + \rho \mathbf{v} \cdot \nabla \mathbf{v} + \rho \mathbf{v} \nabla \cdot \mathbf{v}) = \mathbf{b} \tag{7.16}$$

$$\mathbf{v}\left(\frac{\partial \rho}{\partial t} + \mathbf{v} \cdot \nabla \rho + \rho \nabla \cdot \mathbf{v}\right) + \rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v}\right) = \mathbf{b} \tag{7.17}$$

$$\mathbf{v}\left(\frac{\partial \rho}{\partial t} + \nabla \cdot (\mathbf{v}\rho)\right) + \rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v}\right) = \mathbf{b} \tag{7.18}$$

The term in the left parenthesis is simply the expression developed in the previous section for conversation of mass. Therefore, using equation 7.15:

$$\rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v}\right) = \mathbf{b} \tag{7.19}$$

As the term in parenthesis is the material derivative of the velocity:

$$\rho \frac{D\mathbf{v}}{Dt} = \mathbf{b} \tag{7.20}$$

This is a generalised form of the Cauchy momentum equation.

Strong form of the Navier-Stokes equations

Assuming that the body force term in equation 7.20 consists of two terms, a term to describe forces resulting from stresses and a term to describe other forces present such as gravity, or formally as

$$\mathbf{b} = \nabla \cdot \boldsymbol{\sigma} + \rho \mathbf{g} \tag{7.21}$$

where $\boldsymbol{\sigma}$ is the Cauchy stress tensor and $\mathbf{g}$ is gravity which in the vertical direction is ρg. Substituting equation 7.21 into 7.20 gives:

$$\rho \frac{D\mathbf{v}}{Dt} = \nabla \cdot \boldsymbol{\sigma} + \rho g \tag{7.22}$$

The Cauchy stress tensor may be expressed as follows [31]:

$$\sigma = \begin{bmatrix} \sigma_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_{zz} \end{bmatrix} \tag{7.23}$$

This may be decomposed into its volumetric and deviatoric components as follows:

$$\sigma = - \begin{bmatrix} p & 0 & 0 \\ 0 & p & 0 \\ 0 & 0 & p \end{bmatrix} + \begin{bmatrix} \sigma_{xx} + p & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_{yy} + p & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_{zz} + p \end{bmatrix} \tag{7.24}$$

or simply as:

$$\sigma = -p\mathbf{I} + \mathbb{T} \tag{7.25}$$

where p is the pressure, $\mathbf{I}$ is the identity matrix and $\mathbb{T}$ is the deviatoric stress tensor. The pressure in a fluid is equal to the negative of the mean normal stress [52]. Therefore:

$$p = -\frac{1}{3} \left(\sigma_{xx} + \sigma_{yy} + \sigma_{zz} \right) \tag{7.26}$$

Several assumptions may be made concerning the deviatoric stress tensor. For example for Newtonian fluids, the viscous forces due to fluid flow are proportional to the strain rate at every point in the fluid [51]. For a Newtonian fluid the following assumptions can be made concerning the deviatoric stress tensor [42]:

- It is zero when the fluid is at rest. It also depends only on the spatial derivatives of the fluid velocity.
- It may be expressed as the product of the flow velocity gradient $\nabla \mathbf{v}$ and a viscosity tensor $\mathbf{A}$ or $\mathbb{T} = \mathbf{A}(\nabla \mathbf{v})$.
- The fluid is isotropic, thus, the viscosity tensor $\mathbf{A}$ is isotropic.

- It is symmetric and may be expressed in terms of two viscosity coefficients, μ and λ, where μ is the first viscosity coefficient commonly referred to simply as the viscosity and λ the second coefficient of viscosity often referred to as the bulk viscosity. The bulk viscosity is difficult to determine and a common approximation is $\frac{2}{3}$ [55].

Following these assumptions, the deviatoric stress tensor may be expressed as:

$$\mathbb{T} = 2\mu\mathbf{e} + \lambda\Delta\mathbf{I} \tag{7.27}$$

where Δ is the rate of expansion of the flow defined as [55]:

$$\Delta = \nabla \cdot \mathbf{v} \tag{7.28}$$

and $\mathbf{e}$ is the strain rate tensor defined as:

$$\mathbf{e} = \frac{1}{2}\left(\nabla\mathbf{v}\right) + \frac{1}{2}\left(\nabla\mathbf{v}^T\right) \tag{7.29}$$

This tensor is a measure of the rate of change of the velocity components in each direction [56]. From equations 7.24 and 7.26 it is clear that the trace of this tensor is zero. Consider an incompressible fluid, that is a fluid in which the density is constant within an infinitesimal volume which moves with the fluid velocity. The following may be assumed:

- viscosity μ is a constant
- the bulk viscosity λ is zero
- there are no sources or sinks of momentum within the control volume as the density is constant. From equation 7.15, this means that $\nabla \cdot \mathbf{v} = 0$.

The deviatoric stress tensor therefore simplifies to:

$$\mathbb{T} = 2\mu\mathbf{e} \tag{7.30}$$

where:

$$\mathbf{e} = \frac{1}{2}\nabla\mathbf{v} + \frac{1}{2}\nabla\mathbf{v}' \tag{7.31}$$

Combining equations 7.22, 7.25 and 7.30 gives:

$$\rho\left(\frac{\partial\mathbf{v}}{\partial t} + \mathbf{v}\cdot\nabla\mathbf{v}\right) = -\nabla p + \nabla\cdot\left(2\mu\mathbf{e}\right) + \rho g \tag{7.32}$$

which is the final form of the momentum equation for an incompressible Newtonian fluid.

7.3.4.2 The weak form of the Navier-Stokes equations

Equations 7.15 and 7.32 are what are referred to as the strong form of the Navier-Stokes equations. These relationships must be satisfied at every point in the domain. A weak form of these equations must be formulated in order to obtain approximate solutions to them using the FEM. The weak form of a partial differential equation is one which the solution is required to hold in an average sense over the entire domain. In order to transform a partial differential equation into its weak form, its dot product must be taken with a suitable weight function and then it must be integrated over the domain [57].

The weak form of the mass equation is expressed as:

$$\int_V tr\left(\frac{\partial \mathbf{v}}{\partial \mathbf{x}}\right)\mathbf{w}_p dV = 0 \tag{7.33}$$

where $\mathbf{w}_p$ is a weight function. The development of the weak form of the momentum equation involves taking the dot product of equation 7.32 with a suitable weight function:

$$\int_V \left(\rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}\cdot\nabla\mathbf{v}\right) + \nabla p - 2\mu\nabla\cdot\mathbf{e} - \rho g\right)\mathbf{w}_v dV = 0 \tag{7.34}$$

where $\mathbf{w}_v$ is a vector-valued weight function. There are two acceleration terms in this equation, the unsteady acceleration $\frac{\partial \mathbf{v}}{\partial t}$ and the convective acceleration $\mathbf{v}\cdot\nabla\mathbf{v}$. These will be referred to collectively as the acceleration, $\mathbf{a}$. The stress tensor term is integrated by parts and recalling equation 7.21:

$$\int_V \left(\mathbf{w}_v\rho\cdot\mathbf{a} - p\nabla\cdot\mathbf{w}_v + 2\mu\mathbf{w}_v\nabla\cdot\mathbf{e} - \mathbf{w}_v\cdot\mathbf{b}\right)dV = 0 \tag{7.35}$$

7.3.4.3 Surface tension and contact angle

At the micro-scale under consideration here, surface tension is the dominant force, thus it is essential that this is accounted for. In doing so, one of two approaches may be taken [58]. It may be taken as a boundary condition at the fluid interface or it may be included as forces that act at the fluid-gas interface. The first approach is appropriate if the evolution of the interface is described explicitly whereas the second approach is

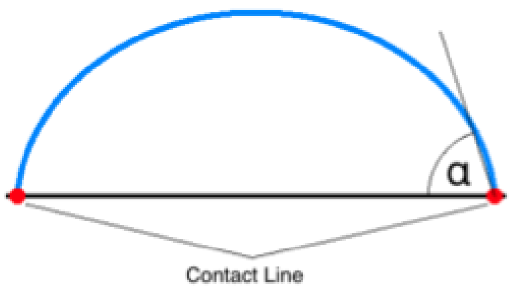


Figure 7.5: Illustration of the contact line. Adapted from [59]

suitable when the evolution of the interface is described implicitly [58], as is the case in the problem under consideration here.

The Young-Laplace equation is used to calculate the pressure difference, Δp, across an interface between two fluids:

$$\Delta p = -\gamma \nabla_s \cdot \mathbf{n} \tag{7.36}$$

where γ is the surface tension, $\mathbf{n}$ is the outward pointing unit normal to the interface and ∇_s is the surface gradient operator defined as $\nabla_s = (\mathbf{I} - \mathbf{n} \otimes \mathbf{n})\nabla \cdot$ [58]. The pressure difference is therefore a function of the geometry of the interface. The angle between the water and the solid surface is referred to as the contact angle, Figure 7.5.

7.3.4.4 The weak form of the surface tension and contact line forces

The following derivation is adapted from Saksono and Perić [58]. The standard Cauchy stress tensor $\boldsymbol{\sigma}(\mathbf{x}, t)$ is defined as:

$$\mathbf{t}(\mathbf{x}, t, \mathbf{n}) = \boldsymbol{\sigma}\mathbf{n} \tag{7.37}$$

where $\mathbf{t}$ is the traction vector and $\mathbf{n}$ is the unit normal. Recalling equation 7.36, the pressure difference may be expressed as the difference between the external pressure and the internal pressure or $\Delta p = p_{ext} - p_{int}$. Using this, the continuity of forces across the fluid-gas interface Γ may be expressed as:

$$\boldsymbol{\sigma}\mathbf{n} = -p_{int}\mathbf{n} = p_{ext}\mathbf{n} + 2\gamma\nabla_s\mathbf{n} \quad on \quad \Gamma \tag{7.38}$$

Expressing equation 7.38 in its weak form:

$$\boldsymbol{\sigma}\mathbf{n} = \int_{\Gamma} p_{ext}\mathbf{n} \cdot \mathbf{w} da - \int_{\Gamma} \gamma \nabla_s \mathbf{n} \cdot \mathbf{w} da \tag{7.39}$$

where $\mathbf{w}$ is a weighting function. The second integral may be further developed using the divergence theorem on surface Γ.

$$-\int_{\Gamma} \nabla_s \mathbf{n} \cdot \mathbf{w} da = \int_{\Gamma} \nabla_s \mathbf{w} da - \int_{C} \gamma \mathbf{w} \cdot \mathbf{m} ds \tag{7.40}$$

where C is the boundary curve of surface Γ and $\mathbf{m}$ is the bi-normal vector of C, which is a vector which is both tangent to the surface and orthogonal to the curve and is a function of the contact angle [58]. In this equation, the contribution of surface tension is accounted for by the first term and the contribution of the contact line force by the second.

7.3.5 Implementation of the computational framework

In the following sections, the computational framework is described in detail. In this problem, the evolution of the fluid droplet towards its equilibrium position is traced by calculating the change in the droplet over a very short period of time, or time-step, and repeating for subsequent time-steps until it reaches its equilibrium position.

7.3.5.1 Overview of the computational model

The problem under consideration is a two-dimensional, axisymmetric model of a micro-fluid droplet which simulates its evolution from an initial, non-equilibrium position to its final equilibrium position. The governing equations consist of the Navier-Stokes equations for incompressible fluids and the Young-Laplace equation for surface tension. The weak form of these equations, derived in the previous sections, are then discretised using a monolithic arbitrary Lagrangian Eulerian formulation. An implicit time integration scheme is used to calculate the droplet's evolution in time, therefore, the governing equations must be linearised. As a monolithic ALE scheme is used, the updated positions of the mesh nodes are calculated, using the mesh quality optimisation algorithms developed throughout this thesis, at the same time as the physical equations.

As this thesis is concerned with mesh quality optimisation, the areas of greatest interest are the application of the mesh quality optimisation equations introduced in Chapter 3 and the re-meshing algorithm which has a direct impact on the quality of the mesh throughout the entire analysis. The Newton-Raphson solver and the adaptive time-step algorithm also have a great impact on the success of the mesh optimisation process. Therefore, these are also discussed in the following sections.

7.3.5.2 Discretisation of the governing equations and the element stiffness matrix and force vector

The development of the time and space discretised form of the weak form of the governing equations for this problem in a cylindrical coordinate system is a significant process completed by the author's colleagues. As this thesis is concerned with mesh quality optimisation, it is therefore far beyond its scope. This process is covered in detail by Mackenzie [42]. The governing equations in their discretised form are given below. Mackenzie [42] should be consulted for a full explanation.

$$\begin{aligned}\mathbf{R}_v(\mathbf{v},\hat{\mathbf{v}}) &= 2\pi\int_r\int_z \mathbf{w}_p r_n\frac{\partial\mathbf{v}_i}{\partial\boldsymbol{\chi}}:\mathbf{I}drdz + \Delta t\pi\int_r\int_v \mathbf{w}_p\frac{\partial\mathbf{v}_i}{\partial\boldsymbol{\chi}}:\mathbf{I}\mathbf{N}_r\boldsymbol{\delta}\hat{\mathbf{v}}drdr\\ &+ 2\pi\int_r\int_z \mathbf{w}_p r_n\frac{\partial\mathbf{N}}{\partial\boldsymbol{\chi}}:\boldsymbol{\delta}\hat{\mathbf{v}}drdr - 2\pi 2\pi\int_r\int_z \mathbf{w}\mathbf{r}_n\frac{\Delta t}{2}\left[\frac{\partial\mathbf{v}_i}{\partial\boldsymbol{\chi}}\right]^T:\frac{\partial\mathbf{N}}{\partial\boldsymbol{\chi}}\boldsymbol{\delta}\hat{\mathbf{v}}drdz\\ &+ 2\pi\int_r\int_z \mathbf{w}_p\cdot(v_i)_r drdz + 2\pi\int_r\int_z(\mathbf{w}_p\cdot\mathbf{N}_r)\,\boldsymbol{\delta}\hat{\mathbf{v}}drdz\end{aligned} \tag{7.41}$$

which may be written as:

$$\mathbf{R}_v(\mathbf{v},\hat{\mathbf{v}}) = \mathbf{f}_{\mathbf{pv}} + \mathbf{f}^{\mathbf{axi}}_{\mathbf{pv}} + \left(\mathbf{K}_{\mathbf{pv}} + \mathbf{K}^{\mathbf{axi}}_{\mathbf{pv}}\right)\boldsymbol{\delta}\mathbf{v} + \left(\mathbf{K}^{\mathbf{axi}}_{\mathbf{p}\hat{\mathbf{v}}}\right)\boldsymbol{\delta}\hat{\mathbf{v}} \tag{7.42}$$

where $\mathbf{K}$ refers to a stiffness term, $\mathbf{M}$ a mass matrix and $\mathbf{f}$ a force vector.

$$\begin{aligned}
\mathbf{R}_v(\mathbf{v},\hat{\mathbf{v}},p) = 2\pi & \int_r\int_z r_n\rho\mathbf{w}_v\left(\frac{\Delta\mathbf{v}}{\Delta t}+\frac{\partial\mathbf{v}_i}{\partial\boldsymbol{\chi}}(\mathbf{v}_i-\hat{\mathbf{v}}_i)\right)-r_n(\mathbf{w}_v\cdot\mathbf{b})dVrdz \\
&+\Delta t\pi\int_r\int_z\left(\rho\mathbf{w}_v\left(\frac{\Delta\mathbf{v}}{\Delta t}+\frac{\partial\mathbf{v}_i}{\partial\boldsymbol{\chi}}(\mathbf{v}_i-\hat{\mathbf{v}}_i)\right)-(\mathbf{w}_v\cdot\mathbf{b}\right)\mathbf{N}_r\boldsymbol{\delta}\hat{\mathbf{v}}drdz \\
&+2\pi\int_r\int_z r_n\rho\mathbf{w}_v\left(\frac{\mathbf{N}}{\Delta t}+\frac{\partial\mathbf{v}_i}{\partial\boldsymbol{\chi}}\mathbf{N}+\frac{\partial\mathbf{N}}{\partial\boldsymbol{\chi}}(\mathbf{v}_i-\hat{\mathbf{v}}_i))\right)\boldsymbol{\delta}\hat{\mathbf{v}}drdz \\
&+2\pi\int_r\int_z r_n\rho\mathbf{w}_v\left(\frac{\Delta t}{2}\frac{\partial\mathbf{v}_i}{\partial\boldsymbol{\chi}}\frac{\partial\mathbf{N}}{\partial\boldsymbol{\chi}}(\mathbf{v}_i-\hat{\mathbf{v}}_i)-\frac{\partial\mathbf{v}_i}{\partial\boldsymbol{\chi}}\mathbf{N}\right)\boldsymbol{\delta}\hat{\mathbf{v}}drdz \\
&+2\pi\int_r\int_z r_n\frac{\partial\mathbf{w}_v}{\partial\boldsymbol{\chi}}:\mathbf{I}p_idrdz-\Delta t\pi\int_r\int_z\left(\frac{\partial\mathbf{w}_v}{\partial\boldsymbol{\chi}}:\mathbf{I}p_i\right)\mathbf{N}_r\boldsymbol{\delta}\hat{\mathbf{v}}drdz \\
&+2\pi\int_r\int_z r_n\frac{\partial\mathbf{w}_v}{\partial\boldsymbol{\chi}}:\mathbf{I}\mathbf{N}\delta pdrdz+2\pi\int_r\int_z r_n\frac{\delta t}{2}\left[\frac{\partial\mathbf{w}_v}{\partial\boldsymbol{\chi}}\right]^T:p_i\frac{\partial\mathbf{N}}{\partial\boldsymbol{\chi}}\boldsymbol{\delta}\hat{\mathbf{v}}drdz \\
&+2\pi\int_r\int_z 2\mu r_n\frac{\partial\mathbf{w}_v}{\partial\boldsymbol{\chi}}:(\mathbf{e}+\boldsymbol{\delta}\mathbf{e}+\boldsymbol{\delta}\hat{\mathbf{e}})\,drdz \\
&+\Delta t\pi\int_r\int_z\left(2\mu\frac{\partial\mathbf{w}_v}{\partial\boldsymbol{\chi}}:\mathbf{e}_i\right)\mathbf{N}_r\boldsymbol{\delta}\hat{\mathbf{v}}drdz \\
&-2\pi\int_r\int_z 2\mu r_n\frac{\Delta t}{2}\left(\frac{\partial\mathbf{w}_v}{\partial\boldsymbol{\chi}}\mathbf{e}_i^T\right):\frac{\partial\mathbf{N}}{\partial\boldsymbol{\chi}}\boldsymbol{\delta}\hat{\mathbf{v}}-2\pi\int_r\int_z(\mathbf{w}_v\cdot p_i)drdz \\
&-2\pi\int_r\int_z(\mathbf{w}_v\cdot\mathbf{N}_r)\delta pdrdz+2\pi\int_r\int_z\frac{2\mu}{r_n}(\mathbf{w}_v\cdot v_i)_rdrdz \\
&-2\pi\int_r\int_z\frac{2\mu\Delta t}{r_n^2}(\mathbf{w}_v\cdot v_i)_r\mathbf{N}_r\boldsymbol{\delta}\hat{\mathbf{v}}drdz+2\pi\int_r\int_z\frac{2\mu}{r_n}(\mathbf{w}_v\cdot\mathbf{N}_r)\boldsymbol{\delta}\hat{\mathbf{v}}drdz
\end{aligned} \tag{7.43}$$

which may be written as:

$$\begin{aligned}
\mathbf{R}_v(\mathbf{v},\hat{\mathbf{v}},p) = & \frac{1}{\Delta t}\left(\mathbf{M_{vv}}\boldsymbol{\delta}\mathbf{v}+\mathbf{M_{v\hat{v}}}\boldsymbol{\delta}\hat{\mathbf{v}}\right)+\left(\mathbf{K_{v\hat{v}}}+\mathbf{K^{axi}_{v\hat{v}}}\right)\boldsymbol{\delta}\mathbf{v}\boldsymbol{\delta}\hat{\mathbf{v}}+\left(\mathbf{K_{vp}}+\mathbf{K^{axi}_{vp}}\right)\delta p \\
&+\left(\mathbf{K_{vp\hat{v}}}+\mathbf{K_{v\hat{v}}}+\mathbf{K^{axi}_{v\hat{v}}}+\mathbf{K^{fr}_{vv}}+\mathbf{K^{fr}_{vm}}+\mathbf{K^{fr}_{vp}}\right)\boldsymbol{\delta}\hat{\mathbf{v}} \\
&+\mathbf{f_{vv}}+\mathbf{f^{axi}_{vv}}+\mathbf{f_{vm}}+\mathbf{f_{vp}}+\mathbf{f^{axi}_{vp}}
\end{aligned} \tag{7.44}$$

As previously stated each mesh node has five degrees of freedom, the change in fluid velocity ($\boldsymbol{\delta}\mathbf{v}$) in the horizontal and vertical directions, the change in pressure (δp) and the change in mesh velocity ($\boldsymbol{\delta}\hat{\mathbf{v}}$). Three noded triangular elements are used, therefore there are fifteen degrees of freedom associated with each element. The system of governing equations given above is shown diagrammatically in Figure 7.6, this represents the element stiffness matrix, force vector and vector of unknowns. The terms referenced in Figure 7.6 are explained below. Note the absence of $\mathbf{A}_{\hat{\mathbf{v}}\hat{\mathbf{v}}}$ and $\mathbf{F}_{\hat{\mathbf{v}}}$ which refer to the stiffness and force term respectively of the mesh quality optimisation equation,

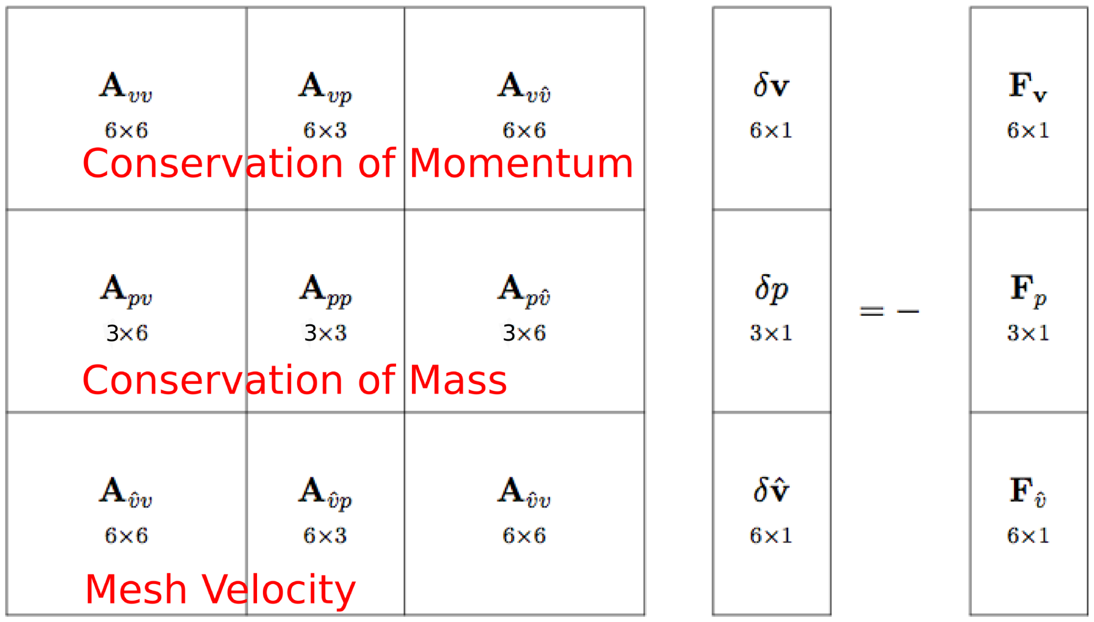

$\mathbf{A}_{vv}$ 6×6
$\mathbf{A}_{vp}$ 6×3
$\mathbf{A}_{v\hat{v}}$ 6×6
Conservation of Momentum
$\mathbf{A}_{pv}$ 3×6
$\mathbf{A}_{pp}$ 3×3
$\mathbf{A}_{p\hat{v}}$ 3×6
Conservation of Mass
$\mathbf{A}_{\hat{v}v}$ 6×6
$\mathbf{A}_{\hat{v}p}$ 6×3
$\mathbf{A}_{\hat{v}v}$ 6×6
Mesh Velocity
$\delta\mathbf{v}$ 6×1
δp 3×1
$\delta\hat{\mathbf{v}}$ 6×1
= −
$\mathbf{F}_{\mathbf{v}}$ 6×1
$\mathbf{F}_p$ 3×1
$\mathbf{F}_{\hat{v}}$ 6×1

Figure 7.6: Element stiffness matrix, vector of unknowns and force vector

which are explained in the following section.

$$\begin{aligned}
\mathbf{A_{vv}} &= \frac{\mathbf{M_{vv}}}{\Delta t} + \mathbf{K_{vv}} + \mathbf{K_{vv}^{axi}} \\
\mathbf{A_{vp}} &= \mathbf{K_{vp}} + \mathbf{K_{vp}^{axi}} \\
\mathbf{A_{v\hat{v}}} &= \frac{\mathbf{M_{v\hat{v}}}}{\Delta t} + \mathbf{K_{v\hat{v}}} + \mathbf{K_{v\hat{v}}^{axi}} + \mathbf{K_{vpv}} + \mathbf{K_{vm}^{fr}} + \mathbf{K_{vp}^{fr}} \\
\mathbf{A_{pv}} &= \mathbf{K_{pv}} + \mathbf{K_{pv}^{axi}} \\
\mathbf{A_{pp}} &= 0 \\
\mathbf{A_{p\hat{v}}} &= \mathbf{K_{p\hat{v}}} + \mathbf{K_{p\hat{v}}^{fr}} \\
\mathbf{A_{\hat{v}p}} &= 0 \\
\mathbf{F_v} &= \mathbf{f_{vv}} + \mathbf{f_{vv}^{axi}} + \mathbf{f_{vm}} + \mathbf{f_{vp}} + \mathbf{f_{vp}^{axi}} \\
\mathbf{F_p} &= \mathbf{f_{pv}} + \mathbf{f_{pv}^{axi}}
\end{aligned}$$

7.3.5.3 The mesh optimisation equations

In this section, the process of implementing mesh optimisation as part of the FE solution process is discussed. In the current implementation of this computational framework, the entries in the element stiffness matrix, $\mathbf{A_{\hat{v}\hat{v}}}$, and in the element force vector, $\mathbf{F_{\hat{v}}}$, associated with the change in mesh velocity are calculated using Laplacian smoothing. The goal of this project is to calculate these terms using mesh optimisation due to the many issues already discussed associated with Laplacian smoothing. In Section 3.2.5 element quality measures are introduced and in Section 3.3 the objective functions used to optimise meshes based on these quality measures are described.

The mesh optimisation process is summarised here. For each element in a mesh, a matrix of second derivatives, or the Hessian, and vector of first derivatives, or the gradient, of the quality measure is calculated. Using an objective function, the system of equations used to optimise a mesh is assembled. This system of equations is then solved using a Newton-Raphson iterative solver. This process closely mirrors the typical FE solution process, however instead of solving for physical degrees of freedom, new nodal positions associated with the optimised mesh are solved for.

In the case of the micro-fluid droplet, both physical degrees of freedom and mesh degrees are freedom are being solved for. Therefore, the entries in the element stiffness matrix and force vector, illustrated in Figure 7.6, associated with the change in mesh velocity are assembled using the exact same method as for mesh optimisation, that is using the element quality objective function. However, as the change in mesh velocity is being solved for, and not displacement for which these equations were derived for, the entries in the Hessian matrix must be multiplied by $0.5\Delta t$. In Chapters 4 and 6, the log-barrier objective function was found to be both effective and efficient at optimising meshes, therefore, it will be used here. In Section 7.3.5.2, it may be seen that the mass matrix terms are divided by $2\Delta t$. For reasons relating to numerical stability, rather than dividing by a very large number, it was chosen to multiply all entries in the system of equations by $2\Delta t$, including the mesh optimisation terms. The stiffness and force terms relating to mesh optimisation are defined as:

$$\mathbf{A}_{\hat{\mathbf{v}}\hat{\mathbf{v}}} = \left(\nabla q_k \left[\frac{1}{1-\gamma} - \frac{1}{(q_k-\gamma)^2} \right] \nabla q_k^T + \left[\frac{q_k}{1-\gamma} - \frac{1}{q_k-\gamma} \right] \nabla^2 q_k \right) \Delta t^2 \tag{7.45}$$

where q_k is the quality of element k, γ is the barrier term described in Section 3.3, Δt is the time-step, $\nabla^2 q_k$ is the Hessian of the element quality measure and ∇q_k is the gradient of the element quality measure.

$$\mathbf{F}_{\hat{\mathbf{v}}} = \left(\left[\frac{q_k}{1-\gamma} - \frac{1}{q_k-\gamma} \right] \nabla q_k \right) 2\Delta t \tag{7.46}$$

Choice of quality measure

In equations 7.45 and 7.46, the q_k, ∇q_k and $\nabla^2 q_k$ terms may be calculated using any suitable quality measure, including anisotropic quality measures. There are two main criteria to be considered in choosing the mesh quality measure. These are efficiency and effectiveness. In Chapter 3, three quality measures were introduced, the area-length (AL) quality measure, the ideal weight inverse mean ratio (IMR) quality measure and the sine quality measure. In terms of efficiency, the sine measure was found to be the most computationally expensive, thus this is eliminated. The choice is therefore between the AL quality measure and the IMR quality measure, both of which were found in Chapters 4 and 6 to be very close in terms of performance and efficiency. The AL quality measure was chosen for convenience reasons. This measure and its derivatives was derived by the author and then implemented in a general manner allowing it to be easily integrated into multiple projects. The IMR quality measure was derived and developed by the Mesquite development team and is implemented using custom data types. This means that a wrapper is required in order to use the Mesquite implementation. This would involve many memory allocations, deallocations and copys, operations which are computationally very expensive. It is for these reasons that the AL quality measure was chosen.

7.3.5.4 Newton-Raphson iterative solver

The Newton-Raphson iterative procedure is an effective and efficient means of solving a system of non-linear algebraic equations resulting from a discretisation of a partial differential equation [58]. Such a solver is used as part of an implicit method. This procedure replaces $\mathbf{F}(\mathbf{X})$ by its first order expansion about the point $\mathbf{F}(X_{t_n})$

$$\mathbf{F}(\mathbf{X_{t_n}}) + \mathbf{K}(\mathbf{X_{t_n}})\left[X_{t_{n+1}} - X_{t_n}\right] = 0 \tag{7.47}$$

where $\mathbf{K}$ is the stiffness matrix which is defined by the directional derivative formula introduced in Section 3.2.5.2.

$$\mathbf{K}(\mathbf{X_{t_n}})[\mathbf{U}] = \frac{\partial}{\partial \epsilon}\bigg|_{\epsilon=0} \mathbf{F}(\mathbf{X}_{t_n} + \epsilon \mathbf{U}) \tag{7.48}$$

Given the solution at time t_n, the solution at time t_{n+1} may be iteratively calculated using this scheme.

7.3.5.5 Boundary conditions

There are four different types of boundary nodes present in this simulation. These are illustrated in Figure 7.7 and are described below.

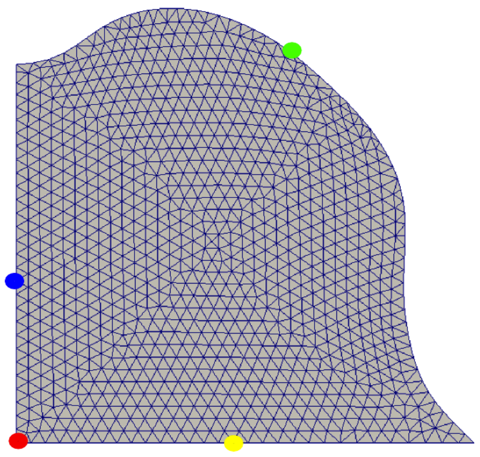


Figure 7.7: Boundary Conditions.

- Red: node is fully fixed.
- Blue: node is fixed in the x direction.
- Yellow: node is fixed in the y direction.
- Green: surface node.

The application of boundary conditions is described in Chapter 5 and the exact same process is used here. It is possible to optimise the positioning of the surface nodes using the constrained optimisation algorithm developed in Chapter 5. This algorithm constrains nodes to complex surfaces based only on information obtained from the mesh. However, the use of this algorithm was found to be very expensive. Therefore, in the initial implementation of the coupling process, surface nodes will be treated as Lagrangian, thus the mesh velocity at each surface node is equal to the fluid velocity. If the coupling process is successful, then this algorithm will be implemented, thus enabling the optimisation of surface nodes.

In the current implementation, the mesh velocity of the surface nodes is taken to be equal to the fluid velocity, thus enforcing the Lagrangian condition. Therefore the $\mathbf{A}_{\hat{\mathbf{v}}\mathbf{v}}$ entries of the element stiffness matrix enforce the condition that the mesh velocity is equal to the fluid velocity or $\delta\mathbf{v} = \delta\hat{\mathbf{v}}$ for all surface nodes.

7.3.5.6 Adaptive time-step algorithm

An adaptive time-step is used in this analysis so that the time-step can either increase or decrease depending on the degree of non-linearity present. At the end of every time-step, the magnitude of the next time-step is calculated using a relationship developed by Mackenzie [42]:

$$\Delta t_{n+1} = \Delta t_n \left(\frac{\iota_d}{\iota_n + 1} \right) \tag{7.49}$$

where Δt_n is the magnitude of the previous time-step, i_n is the number of Newton iterations required for the previous time-step and ι_d is the user defined desired number of Newton iterations. More sophisticated adaptive time-step algorithms are presented by Volker and Rang [60].

7.3.5.7 Re-meshing algorithm

Between time-steps, it may be necessary to re-mesh all or parts of the domain due to the complex, constantly evolving domain, irrespective of the effectiveness of the mesh updating algorithm used. The re-meshing algorithm used in this analysis is a two step process. Firstly, areas of the mesh of high or low nodal density relative to the rest of the mesh are targeted. In areas of high nodal density, unnecessary nodes are removed. Conversely, in areas of low nodal density, new nodes are inserted. The second stage involves computing a new triangulation of the domain.

Node insertion and deletion operations

The node insertion and deletion algorithm used in this analysis is based on the same premise as Laplacian smoothing: that mesh composed of equidistant nodes are of good quality. A new node is inserted at the middle of any edge if the length of the edge is greater than 1.5 times the average edge length. If the length of an edge is less than 0.5 times the average edge length, one of its nodes is deleted along with all the edges which are connected to that node. This is illustrated in Figure 7.8. The resulting cavity is then re-meshed during the second stage of the re-meshing process.

Calculation of the new triangulation

A new triangulation is calculated after every time-step, even if no node insertions or deletions have been performed. This is because the current triangulation may not be the optimal triangulation for the new nodal positions. Therefore, the mesh connectivity is deleted and the nodal coordinates are passed to the computational geometry

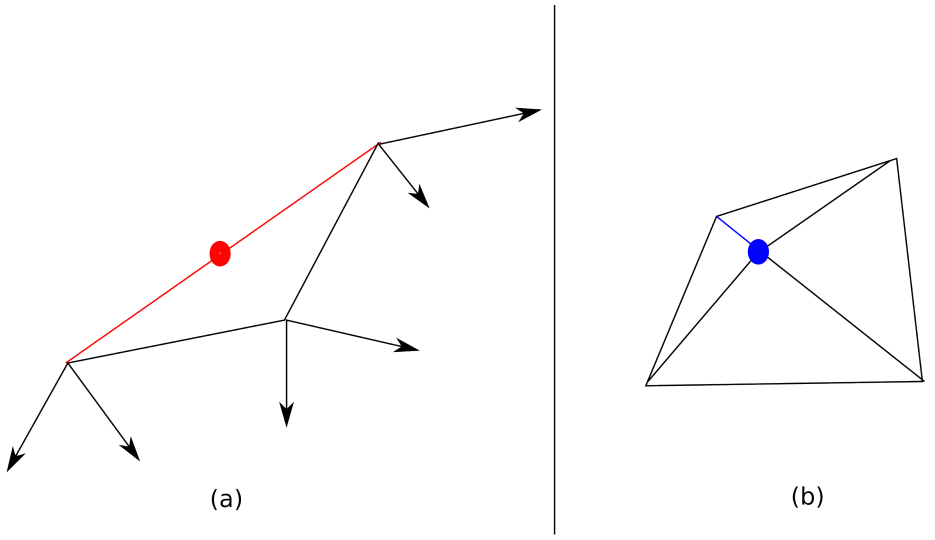


Figure 7.8: Node insertion and deletion. (a) A new node is inserted at the mid-point of the long edge marked in red. (b) One of the nodes on the short edge, marked in blue, is removed.

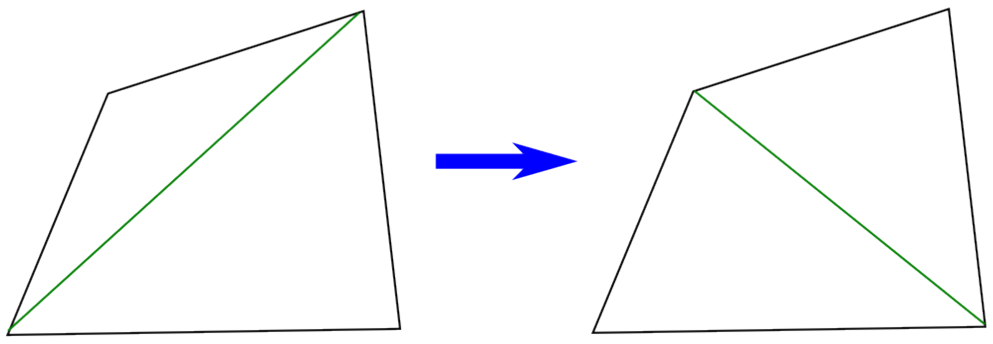


Figure 7.9: The edge marked in green is flipped, resulting in a higher quality mesh.

library CGAL [27] which computes the new triangulation. CGAL returns the Delaunay triangulation of the set of nodes passed to it. In a Delaunay triangulation of a set of nodes, no node may lie within the circumcircle of a triangle, they are only permitted to lie on the boundary of a circumcircle [61]. Delaunay triangulations are typically very high quality triangulations [61]. Although the entire domain is re-meshed after every time-step, in practice changes to the mesh between time-steps are minimal. Changes are generally reserved for regions of the mesh where node insertions or deletions have been performed. However, edge flips, illustrated in Figure 7.9, may also be performed in areas where no nodal insertions or deletions have occurred. This is because the change in nodal positions during a time-step may result in a mesh which is no longer a Delaunay triangulation.

Solution adaptive re-meshing

Solution adaptive re-meshing, introduced in Section 2.1.2, is a process where either the entire domain, or selected parts of it, are re-meshed periodically based on an estimation of the solution error. This aims to give as accurate a solution as possible while limiting the number of of degrees of freedom. This estimation may be used to calculate the nodal density throughout the mesh, as used by Saksono and Perić [62], where the norm of the gradient of the fluid velocity is used to determine which part of the mesh requires refining or coarsening. Alternatively, the method described in Section 2.1.3, where the shape, size and orientation of the ideal element at each point in the domain is calculated by defining a metric field over the domain. In cases where the mesh is used to solve for multiple variables, such as the problem under consideration here, the method of combining multiple metric tensors described in Section 2.1.4.2 may be used. The mesh optimisation process described in this chapter may then be combined with a quality measure which measures the size, shape and orientation of an element, such as that described in Section 3.2.6.

The overall goal of the mesh optimisation and re-meshing process is to limit the introduction of errors caused by the mesh. The source of these errors is the interpolation of variables using unsuitable elements and the transfer of data from one mesh to another. Therefore, it is highly advantageous to limit re-meshing as much as possible, as each re-mesh implies the transfer of data. In this problem, the direction of anisotropy is not known a-priori and is subject to rapid change, meaning that a mesh which is ideal at one time-step may not be ideal during subsequent time-steps. Therefore, isotropic elements, whilst not ideal, may in fact reduce the amount of re-meshing required. However, there is definite merit in the further investigation of the both types of solution adaptive re-meshing described in this section to later stages of this project.

7.3.6 Results

An analysis using the computational framework described in the previous sections was performed on the micro-fluid droplet using the mesh optimisation algorithm described above. For comparison, an analysis using Laplacian smoothing was run using the same parameters. In this section the results of both analysis are compared to gauge the effectiveness of the mesh optimisation algorithms. To-reiterate, mesh optimisation is being used due to the problems associated with Laplacian smoothing. For example, analysis were found to be terminating prematurely due to issues with the mesh and

	Laplacian Smoothing	**Mesh Optimisation**
Edge flips	271	167
Nodes added	121	76
Nodes removed	141	34
Area re-meshed	149.34%	81.83%
Volume change	-0.139%	-0.036%
Range of angles	$8.07° - 163.06°$	$23.5° - 115.03°$

Table 7.1: Comparison of Laplacian smoothing and mesh optimisation

volume loss was observed. Laplacian smoothing also requires the use of an artificial mesh viscosity parameter, which is not required when mesh optimisation is used. This parameter was the source of many problems, thus a method which does not require its use is desirable. It is also desirable to limit re-meshing as much as possible as this was shown in Chapter 2 to cause numerical errors.

Each analysis is compared using the following criteria to give as broad a picture as possible of the advantages and disadvantages of each method of calculating the change in mesh velocity. Firstly the degree of re-meshing which occurred during the inter time-step re-meshing process is assessed. This is done by tracking the cumulative number of nodes added and removed and the number of edges flipped over the course of the analysis. The cumulative area re-meshed is also calculated and is expressed as a percentage of the area of the domain. If an edge is flipped, the area of the two triangles on that edge is added to the total area re-meshed. If a node is added or removed, the area of any elements defined by that node contribute to the cumulative area re-meshed. The change in volume of the domain and the magnitude of the average time-step were also used for comparison. It is desirable to have as large a time-step as possible as the larger the time-step, the less time taken to run an analysis. Finally, to determine the quality of the meshes over the course of the analysis, the range of angles observed in the triangular elements over the course of each analysis was calculated. These results are shown in Table 7.1.

It can be seen that the use of mesh quality optimisation instead of Laplacian smoothing has reduced the amount of re-meshing required by almost half. Volume loss has been reduced to a quarter of its previous value and there is a much tighter bound on the range of angles present during the analysis. However, all this has come at a very large cost - the magnitude of the time-step is less than one percent of its previous value. These analyses typically take from several days to several weeks to terminate, depending on the particular analysis. With this new reduced time-step, running an

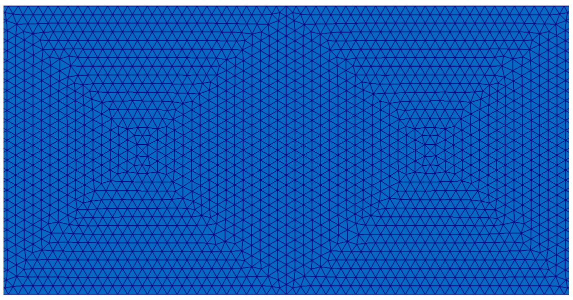


Figure 7.10: Initial droplet shape.

analysis until it reaches its equilibrium position could take several years. This is clearly not practicable, even though all of the initial goals have been achieved.

Careful scrutiny of the results of the analysis revealed that the mesh optimisation process was too aggressive. The adaptive time-step algorithm described in Section 7.3.5.6 determines the magnitude of the next time-step based on the number of iterations the previous time-step took to reach convergence. The greater the number of iterations required, the greater the degree of non-linearity of the system of equations. The aggressive mesh optimisation algorithms applied to this computational framework introduced very large non-linearities into the system, thus causing the time-step to plummet. It is clear that these algorithms in their current form are not at all suited to this application. Therefore, a modified, less aggressive form of this algorithm which introduces lesser non-linearities into the system, thus having a lesser effect on magnitude of the time-step, is required. Such an algorithm is described in the next section. To ensure that the time-stepping problem was not specific to the particular quality measure used or analysis performed, several different analyses were performed using both the IMR quality measure and AL quality measure. The results obtained were very similar to those presented in Table 7.1.

7.3.7 Maintaining mesh quality

It was shown in the previous section that using mesh quality optimisation to calculate the change in mesh velocity in an ALE analysis is quite effective, however this comes at the expense of a significant reduction in the magnitude of the time-step. This means that the time taken to perform an analysis increases drastically. In this section, a modified version of the mesh optimisation algorithm is presented, which aims to preserve the quality of the mesh during the deformation process, rather than optimising it, thus having a lesser effect on the magnitude of the time-step.

The initial mesh of the fluid droplet is generated using Cubit [24], a powerful mesh generation package. All analysis involving the fluid droplet begin from an initial non-equilibrium position, for example that shown in Figure 7.10. The initial mesh are always of good quality due to the powerful meshing algorithms utilised by Cubit and the simple domain shape. In Section 7.3.5.7, the inter time-step re-meshing process is described. Therefore, the mesh is generally also of good quality at the beginning of each time-step, and thus, only minor numerical and performance benefits may be gained from aggressively optimising it. This means that instead of optimising the mesh, it is only necessary to maintain the quality of the mesh during the deformation process. This is achieved by re-defining the quality of an element so that it is a function of its quality at the beginning of the time-step, Equation 7.50.

$$q_k = \frac{q_k^i}{q_k^0} \tag{7.50}$$

where q_k^i is the quality of element k at the i^{th} iteration and q_k^0 is the quality of element k at the beginning of the time-step. The rest of the process remains identical to that described for the mesh optimisation case. As before, the quality measure is combined with the log-barrier objective function, Figure 7.11. If the quality of an element remains unchanged during a time-step, then its quality is one and no effort is made to improve it. If it improves during a time-step, its quality is greater than one and if it deteriorates, its quality is less than one. Therefore, at the first iteration of every time-step, the mesh velocity equations make no contribution to the element stiffness matrix and force vector. This means that there are no residuals associated with mesh velocity at the beginning at each time-step. It is only when changes are made to the mesh that these entries are non-zero. There is a slight disadvantage associated with this approach. If the quality of an element is improved during a time-step, this algorithm will attempt to restore it to its previous quality. However, from Figure 7.11 it can be seen that the non-linearities associated with restoring an element which has been improved to its original quality are lesser than those generated when the quality of an element deteriorates. These non-linearities are unnecessary and counter-productive. Nevertheless, the punishment for improving an element is far less than that for damaging it, so whilst a disadvantage, it is not debilitating.

7.3.7.1 Results

Similar to the previous section, the results obtained using mesh quality optimisation and Laplacian smoothing are compared in this section. To reiterate, the aim of this

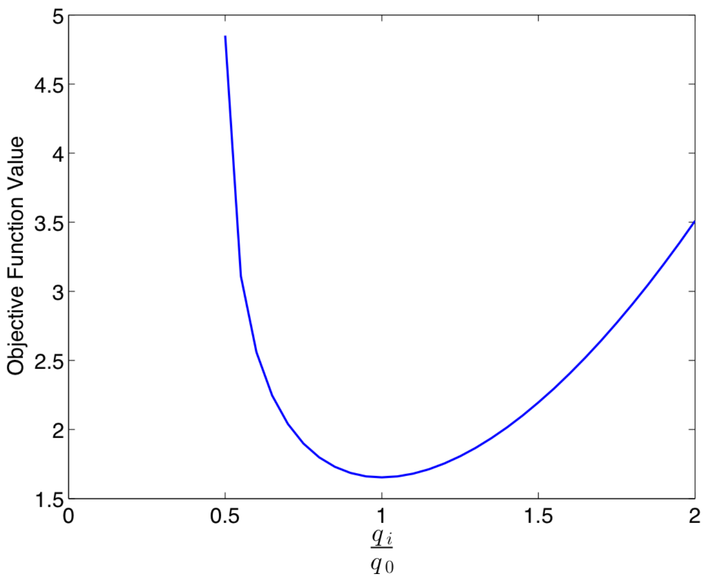
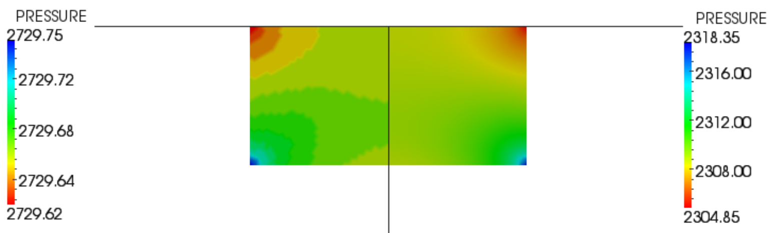


Figure 7.11: The log-barrier function used to maintain mesh quality

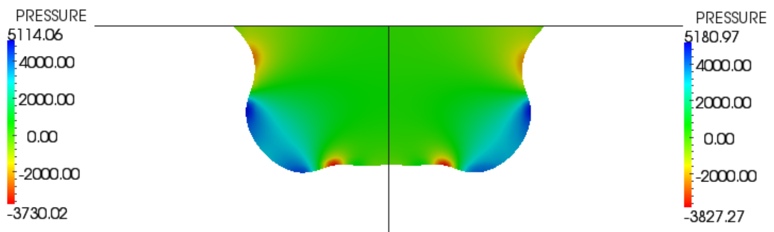
process is to limit the amount of re-meshing required throughout the analysis and to minimise the change in volume of the domain. It is also desired to maximise the minimum angle and minimise the maximum angle present in the mesh and to remove the need for the mesh viscosity parameter, all whilst having as large a time-step as possible. The results are presented in the following pages with the shapes of the two droplets tracked throughout the entire analysis. When presenting the results, the two droplets shown side by side could be either those from the same times during the analysis or when they are at similar positions in the evolution of their shapes towards their respective equilibrium positions and the shapes assumed by the droplets closely coincide. The decision was made to present the two droplets side by side at similar stages of their evolution towards equilibrium, for example approaching the maximum/minimum height as this allows for comparison between the magnitudes of their amplitudes, the differences in the shapes assumed by the droplets and the pressure distribution. It can frequently be seen that the range of pressures in both droplets at similar stages of the simulation differ extensively. However, strong conclusions cannot be drawn from this as these pressures vary substantially between successive time-steps so the range of pressures may be very different between successive time-steps even if the shape of the droplet has only slightly changed. It was also decided to omit the mesh from this diagram as the density of the mesh is such that it is impossible to distinguish individual elements unless magnified by a great deal as each mesh contains approximately two thousand elements. The plotting of the mesh also makes the pressure distribution impossible to see. Therefore, it was decided that more information could be conveyed in the diagram by omitting the mesh. The progression of the droplet throughout the entire analysis has been presented so as to demonstrate the immense changes the domain undergoes and thus clearly show the magnitude of the problem of maintaining mesh quality.

From the results it is clear that the use of mesh optimisation/preservation has had a very large effect on the results of the analysis. The shape of the droplet, the amplitude of its oscillations, the pressure fluctuations, the change in volume and the time taken for it to reach its equilibrium position have all been affected by the addition of the mesh quality preservation equations. It is clear that this computational framework has a strong mesh dependence, although the differences in the results cannot be attributed to a mesh dependence alone. The degree of re-meshing is determined in the same manner as before, by tracking the number of nodes added and removed and by counting the number of edge flips performed. The cumulative area re-meshed is also calculated and expressed as a percentage of the area of the domain. The cumulative area of the domain which is re-meshed in the analysis using mesh optimisation is 5.51 times greater than that re-meshed by the analysis using Laplacian smoothing. As shown in Chapter 2, the transfer of data from one mesh to another, which is required anytime re-meshing is performed, leads to numerical errors. These certainly will have contributed to difference between the two solutions. The inter time-step re-meshing algorithm is re-examined in the following section to understand the reasons for such a dramatic increase in the cumulative area being re-meshed.

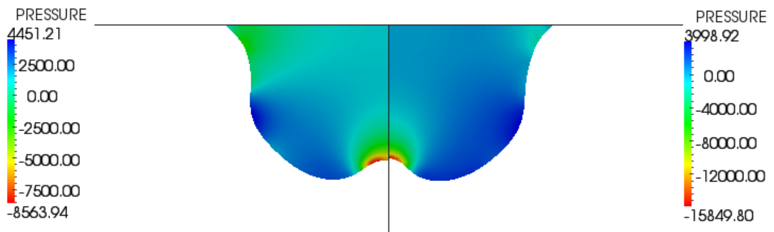
Laplacian Smoothing | **Mesh Optimisation**



Laplacian Smoothing	Mesh Optimisation
t = 0.0s	t = 0.0s
flips = 0	flips = 0
Nodes Added = 0	Nodes Added = 0
Nodes Removed = 0	Nodes Removed = 0
Re-meshed = 0.0%	Re-meshed = 0.0%
Volume Change = 0.0%	Volume Change = 0.0%

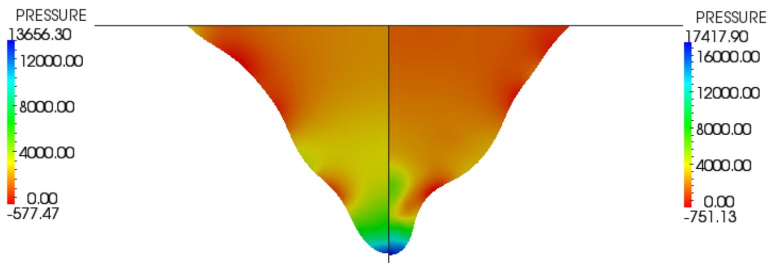


t = 1.71×10^{-5}s	t = 5.62×10^{-6}s
flips = 6	flips = 6
Nodes Added = 1	Nodes Added = 1
Nodes Removed = 0	Nodes Removed = 0
Re-meshed = 4.66%	Re-meshed = 3.21%
Volume Change = 4.000×10^{-7}%	Volume Change = 1.343×10^{-5}%

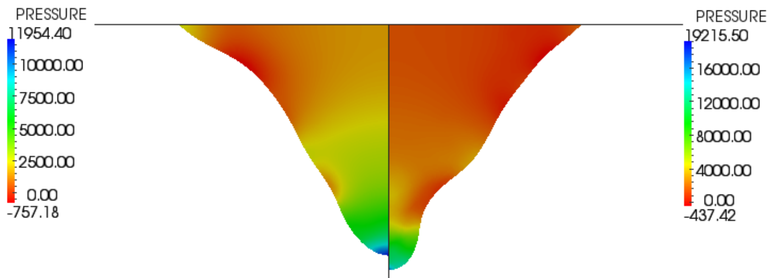


t = 2.95×10^{-5}s	t = 1.07×10^{-5}s
flips = 17	flips = 6
Nodes Added = 1	Nodes Added = 1
Nodes Removed = 2	Nodes Removed = 0
Re-meshed = 13.51%	Re-meshed = 3.21%
Volume Change = 1.000×10^{-7}%	Volume Change = 2.686×10^{-5}%

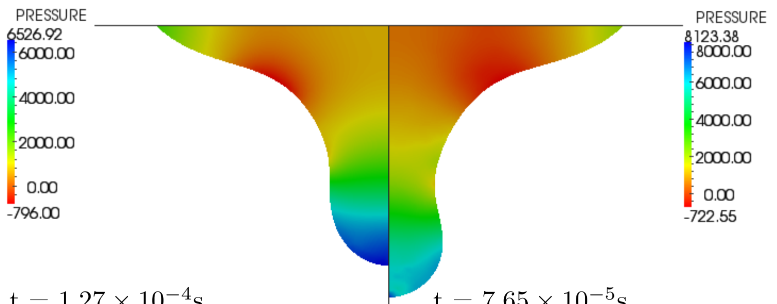
Laplacian Smoothing	**Mesh Optimisation**



t = 6.90×10^{-5}s	t = 2.88×10^{-5}s
flips = 48	flips = 12
Nodes Added = 3	Nodes Added = 2
Nodes Removed = 5	Nodes Removed = 0
Re-meshed = 36.61%	Re-meshed = 7.04%
Volume Change = 2.005×10^{-7}%	Volume Change = 0.732×10^{-4}%

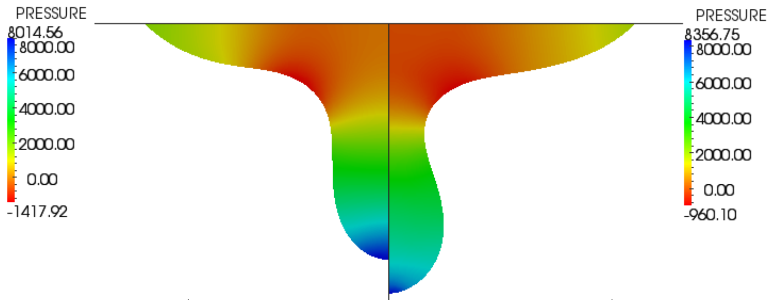


t = 7.98×10^{-5}s	t = 3.42×10^{-5}s
flips = 63	flips = 38
Nodes Added = 4	Nodes Added = 3
Nodes Removed = 7	Nodes Removed = 0
Re-meshed = 46.00%	Re-meshed = 31.06%
Volume Change = 1.002×10^{-6}%	Volume Change = 0.843×10^{-4}%

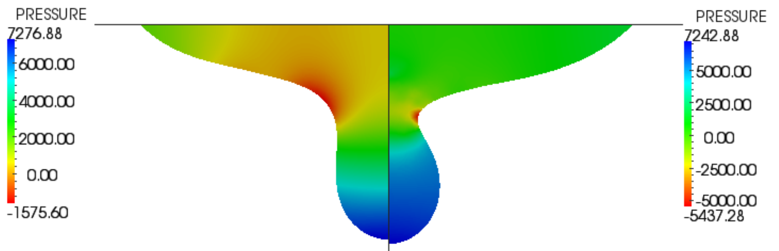


t = 1.27×10^{-4}s	t = 7.65×10^{-5}s
flips = 122	flips = 76
Nodes Added = 9	Nodes Added = 6
Nodes Removed = 10	Nodes Removed = 1
Re-meshed = 85.35%	Re-meshed = 61.79%
Volume Change = 1.938×10^{-6}%	Volume Change = 1.918×10^{-4}%

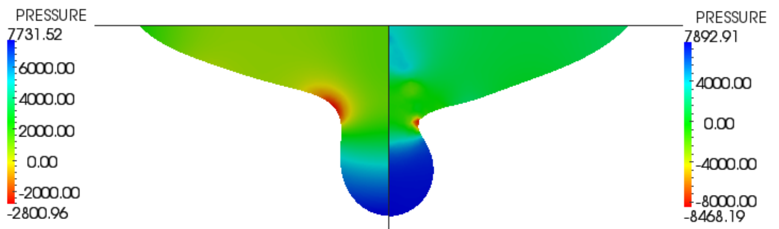
Laplacian Smoothing	**Mesh Optimisation**



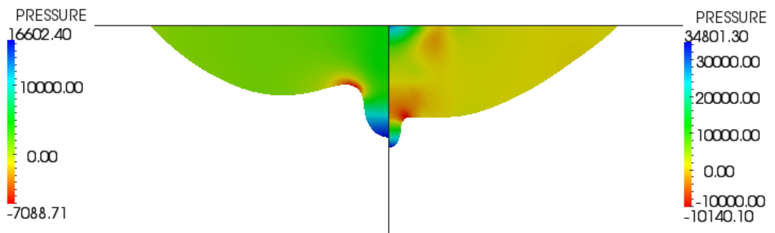
t = 1.74×10^{-4}s	t = 1.31×10^{-4}s
flips = 199	flips = 126
Nodes Added = 16	Nodes Added = 13
Nodes Removed = 13	Nodes Removed = 2
Re-meshed = 138.30%	Re-meshed = 92.02%
Volume Change = 3.675×10^{-6}%	Volume Change = 3.384×10^{-4}%



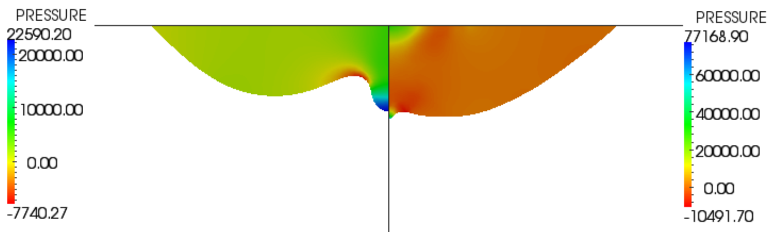
t = 2.02×10^{-4}s	t = 2.17×10^{-4}s
flips = 234	flips = 620
Nodes Added = 19	Nodes Added = 32
Nodes Removed = 15	Nodes Removed = 6
Re-meshed = 164.05%	Re-meshed = 558.99%
Volume Change = 4.477×10^{-6}%	Volume Change = 5.434×10^{-4}%



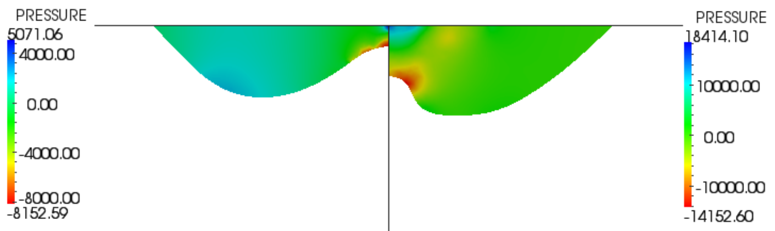
t = 2.26×10^{-4}s	t = 2.63×10^{-4}s
flips = 301	flips = 1166
Nodes Added = 26	Nodes Added = 49
Nodes Removed = 18	Nodes Removed = 10
Re-meshed = 204.23%	Re-meshed = 1042.40%
Volume Change = 5.279×10^{-6}%	Volume Change = 6.646×10^{-4}%



Laplacian Smoothing	**Mesh Optimisation**
t = 2.73×10^{-4}s	t = 3.00×10^{-4}s
flips = 436	flips = 2172
Nodes Added = 40	Nodes Added = 94
Nodes Removed = 22	Nodes Removed = 23
Re-meshed = 289.25%	Re-meshed = 1873.32%
Volume Change = 0.668×10^{-5}%	Volume Change = 0.907×10^{-3}%

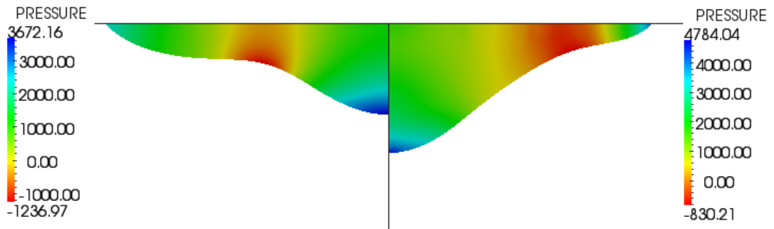


t = 2.81×10^{-4}s	t = 3.18×10^{-4}s
flips = 476	flips = 3129
Nodes Added = 46	Nodes Added = 111
Nodes Removed = 23	Nodes Removed = 29
Re-meshed = 308.12%	Re-meshed = 2759.01%
Volume Change = 0.695×10^{-5}%	Volume Change = 1.138×10^{-3}%

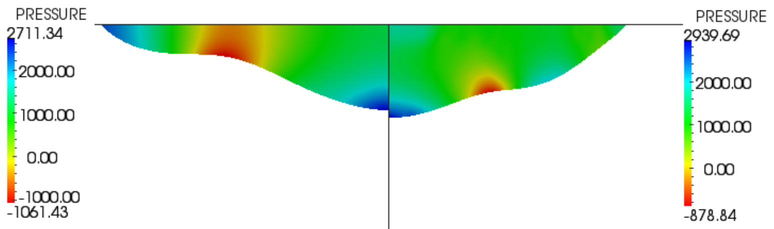


t = 2.93×10^{-4}s	t = 3.35×10^{-4}s
flips = 605	flips = 4061
Nodes Added = 62	Nodes Added = 136
Nodes Removed = 33	Nodes Removed = 33
Re-meshed = 375.36%	Re-meshed = 3675.41%
Volume Change = 1.331×10^{-2}%	Volume Change = 1.206×10^{-3}%

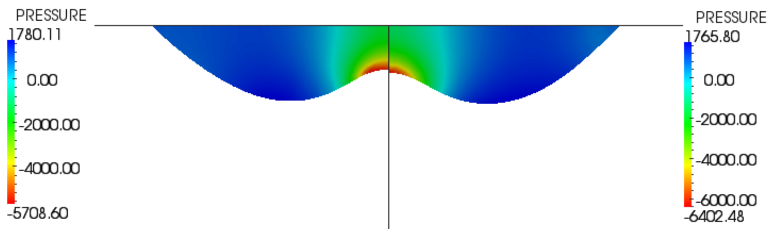
Laplacian Smoothing **Mesh Optimisation**



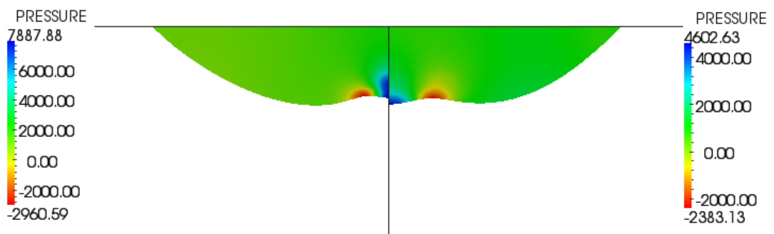
Laplacian Smoothing	Mesh Optimisation
t $= 4.13 \times 10^{-4}$s	t $= 3.82 \times 10^{-4}$s
flips = 1301	flips = 5942
Nodes Added = 138	Nodes Added = 201
Nodes Removed = 63	Nodes Removed = 43
Re-meshed = 805.36%	Re-meshed = 5439.77%
Volume Change $= 0.803 \times 10^{-1}$%	Volume Change $= 1.298 \times 10^{-3}$%



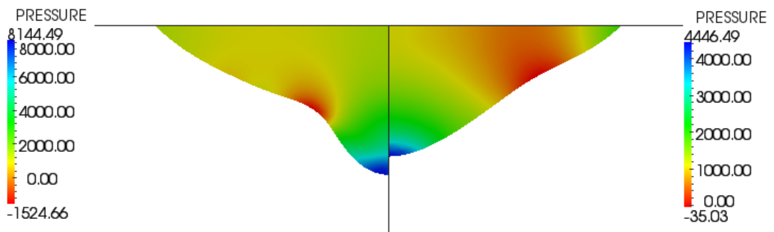
Laplacian Smoothing	Mesh Optimisation
t $= 4.55 \times 10^{-4}$s	t $= 3.97 \times 10^{-4}$s
flips = 1568	flips = 6471
Nodes Added = 168	Nodes Added = 223
Nodes Removed = 83	Nodes Removed = 50
Re-meshed = 961.03%	Re-meshed = 5930.77%
Volume Change $= 0.803 \times 10^{-1}$%	Volume Change $= 1.338 \times 10^{-3}$%



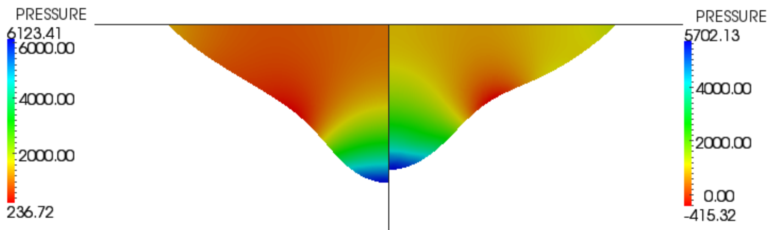
Laplacian Smoothing	Mesh Optimisation
t $= 5.73 \times 10^{-4}$s	t $= 4.63 \times 10^{-4}$s
flips = 2261	flips = 9416
Nodes Added = 235	Nodes Added = 335
Nodes Removed = 137	Nodes Removed = 78
Re-meshed = 1370.94%	Re-meshed = 8543.45%
Volume Change $= 0.939 \times 10^{-1}$%	Volume Change $= 2.183 \times 10^{-3}$%



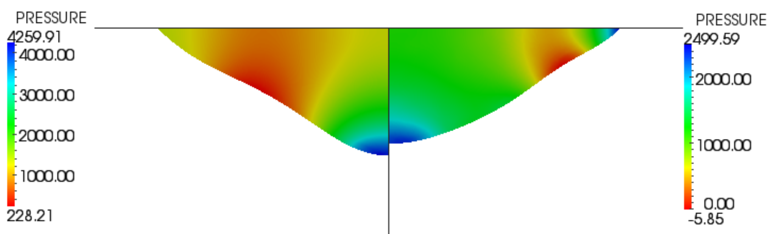
Laplacian Smoothing	Mesh Optimisation
t = 5.84×10^{-4}s	t = 4.73×10^{-4}s
flips = 2312	flips = 9713
Nodes Added = 239	Nodes Added = 346
Nodes Removed = 143	Nodes Removed = 79
Re-meshed = 1403.76%	Re-meshed = 8820.28%
Volume Change = 0.939×10^{-1}%	Volume Change = 2.199×10^{-3}%



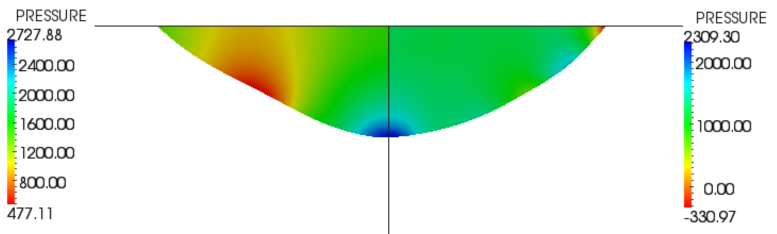
Laplacian Smoothing	Mesh Optimisation
t = 6.16×10^{-4}s	t = 5.05×10^{-4}s
flips = 2459	flips = 10968
Nodes Added = 255	Nodes Added = 397
Nodes Removed = 150	Nodes Removed = 95
Re-meshed = 1490.96%	Re-meshed = 9873.47%
Volume Change = 0.939×10^{-1}%	Volume Change = 2.364×10^{-3}%



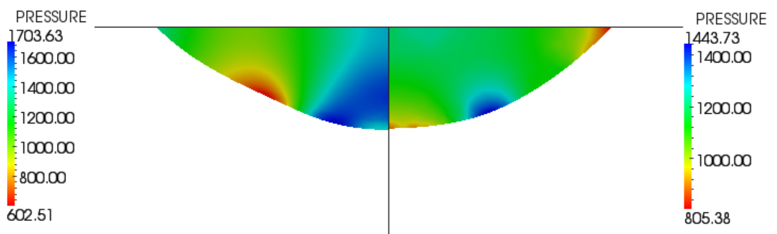
Laplacian Smoothing	Mesh Optimisation
t = 7.91×10^{-4}s	t = 5.72×10^{-4}s
flips = 3266	flips = 13310
Nodes Added = 320	Nodes Added = 478
Nodes Removed = 232	Nodes Removed = 131
Re-meshed = 1990.75%	Re-meshed = 12008.71%
Volume Change = 1.624×10^{-1}%	Volume Change = 2.402×10^{-3}%



Laplacian Smoothing	Mesh Optimisation
t = 8.04×10^{-4}s	t = 5.89×10^{-4}s
flips = 3406	flips = 14023
Nodes Added = 329	Nodes Added = 490
Nodes Removed = 248	Nodes Removed = 140
Re-meshed = 2074.05%	Re-meshed = 12749.40%
Volume Change = 1.696×10^{-1}%	Volume Change = 2.438×10^{-3}%



Laplacian Smoothing	Mesh Optimisation
t = 8.21×10^{-4}s	t = 6.18×10^{-4}s
flips = 3938	flips = 15283
Nodes Added = 363	Nodes Added = 510
Nodes Removed = 311	Nodes Removed = 154
Re-meshed = 2412.84%	Re-meshed = 13367.20%
Volume Change = 1.701×10^{-1}%	Volume Change = 2.454×10^{-3}%



Laplacian Smoothing	Mesh Optimisation
t = 8.43×10^{-4}s	t = 6.05×10^{-4}s
flips = 3957	flips = 14746
Nodes Added = 364	Nodes Added = 532
Nodes Removed = 314	Nodes Removed = 161
Re-meshed = 2422.89%	Re-meshed = 13792.30%
Volume Change = 1.701×10^{-1}%	Volume Change = 2.589×10^{-3}%

Re-meshing

The inter time-step re-meshing process is described in Section 7.3.5.7. This re-meshing algorithm is a two stage process consisting of node insertion and deletions followed by the computation of the Delaunay triangulation of the nodes. Whilst a Delaunay triangulation is generally of high quality, some of the changes made between time-steps, for example edge flips, were observed to provide only marginal mesh quality improvements. It was frequently observed that a flipped edge is flipped again after the following time-step. This means that two unnecessary re-meshing operations are performed, each introducing numerical errors. Therefore, it is recommended that the frequency of re-meshing be reduced, perhaps only re-meshing when the quality drops below a certain threshold. It is also recommended that each alteration to the mesh be examined to ensure that it improves the mesh by a significant amount.

The node insertion/deletion algorithm used in this analysis is based on the same premise as Laplacian smoothing: that meshes composed of equidistant nodes are of good quality. Therefore, it has a strong bias for meshes which are optimised using Laplacian smoothing. The basic premise of this algorithm is fundamentally flawed in that it is purely based on heuristics, unlike mesh quality measures which are based on strong mathematical foundations. This means that many node insertions and deletions may be unnecessary. A more sophisticated insertion/deletion algorithm based on mesh quality measures should be far more effective and would eliminate many unnecessary mesh modification operations. It must be noted that in the analysis using Laplacian smoothing, the number of mesh nodes remained relatively constant throughout, whereas with quality preservation the number of nodes fluctuated greatly. It is therefore strongly recommended that future versions of this model and similar models utilise node insertion/deletion algorithms based on mesh quality measures and not on empirical quantities.

Re-meshing based on the analysis results

The re-meshing algorithm in its current form is clearly not fit for purpose due to the reasons described in the previous section. Solution adaptive re-meshing could lead to significant reductions in the level of re-meshing required as well as a reduction in numerical errors caused by the mesh. The results displayed on the previous pages could be used in a solution adaptive re-meshing scheme. For example, in Figure 7.12, the distribution of the pressure is shown on the left and the distribution of the magnitude

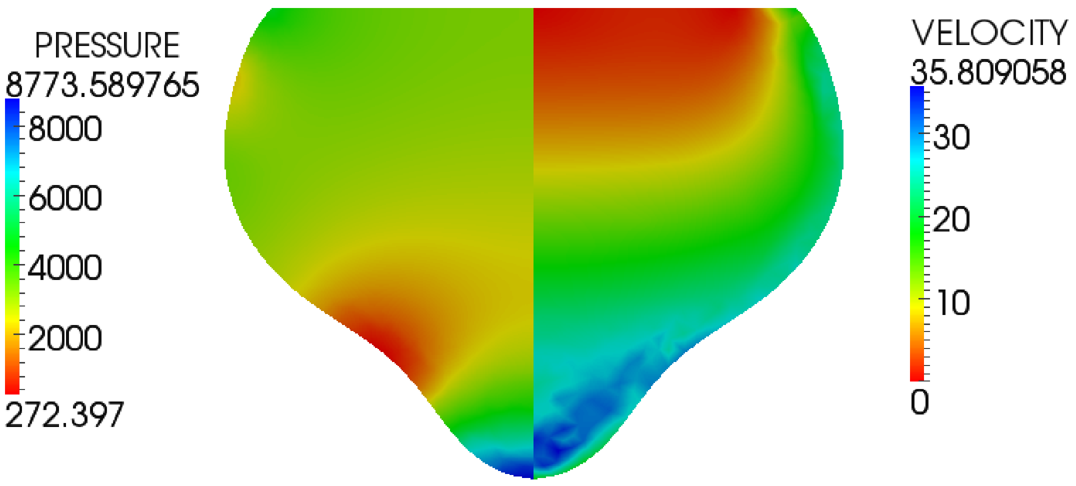


Figure 7.12: The magnitude of the pressure is plotted on the left and the magnitude of the fluid velocity is plotted on the right.

of the fluid velocity is shown on the right. The pressure is relatively constant over much of the domain. Large, isotropic elements would be very well-suited for the interpolation of the pressure in this area. The magnitude of the fluid velocity is relatively constant in the red and green areas, where again large isotropic elements would be suitable. However, the region of rapid change, marked by the yellow boundary between the two areas would require would require finer elements to ensure accurate interpolation of the data.

The x and y components of the fluid velocity are plotted in Figure 7.13. Correctly oriented anisotropic elements could be very effective in the blue region of the y velocity. This is because the x velocity is relatively constant in this area but the y velocity changes rapidly. Therefore, anisotropic elements, which are much longer in the x direction than in the y direction may be most suited here.

Time-Step

The use of quality preservation in place of optimisation has increased the average time-step from 1% of that of Laplacian smoothing to 10%. This is clearly a great improvement, however it remains far from ideal. Several solutions are proposed to alleviate the impact of the reduced time-step. Firstly, this simulation is implemented using only one processor, thus, parallelisation could greatly reduce the time required. This simulation is currently in development, thus the focus is on correctness and functionality with little thought for performance. The use of modern compilers with full

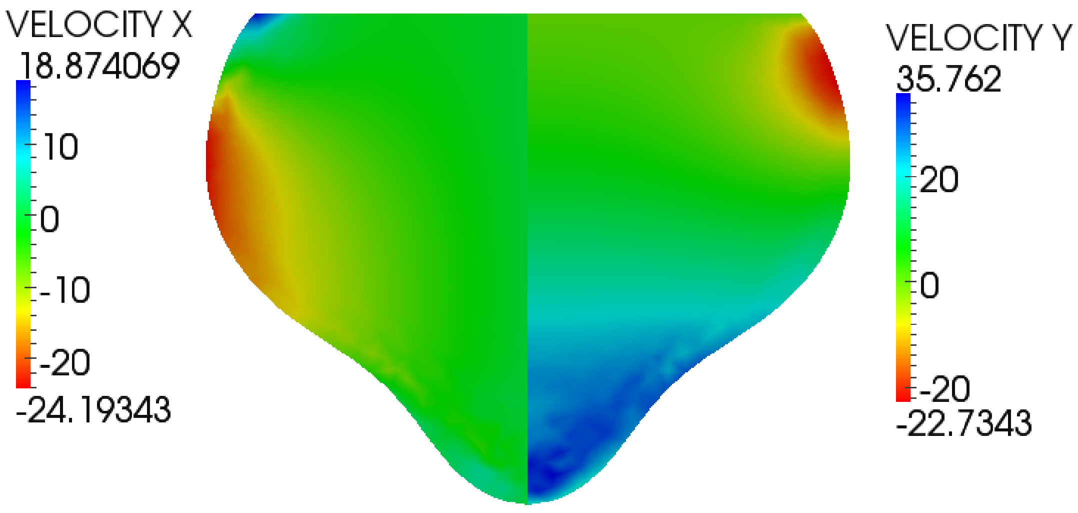


Figure 7.13: The x component of the fluid velocity is plotted on the left and the y component of the fluid velocity is plotted on the right.

optimisation enabled, experimentation with various implementations of mathematical libraries such as Atlas [63] or Intel's mkl [64] (both available freely for non-commercial use) which have been shown to give significant performance boosts and profiling tools to identity the most expensive sections of the code, thus enabling the optimisation of inefficient code, could also greatly improve performance. The more sophisticated adaptive time-step algorithms described by Volker and Rang [60] could also ensure that the most appropriate time-step is chosen, thus increasing performance.

Mesh quality

Figure 7.14 shows the range of angles in both simulations versus time. Both simulations have a very similar bound on the smallest angle in the mesh, although it may be seen that the range of angles for quality preservation is much smoother. Quality preservation has a higher bound on the upper angle in the mesh, although again it may be seen that this is much smoother. The sudden jumps in the range of angles correspond to changes in the mesh topology, i.e. node insertions/deletions or edge flips. It is clear that topological transformations are much more effective for Laplacian smoothing than for quality preservation due to the sudden changes in the minimum and maximum angles observed. Although much fewer topological transformation operations occur for Laplacian smoothing, they are clearly much more effective. This again confirms the suspicion that much of the re-meshing that is occurring for quality preservation is unnecessary and may be eliminated.

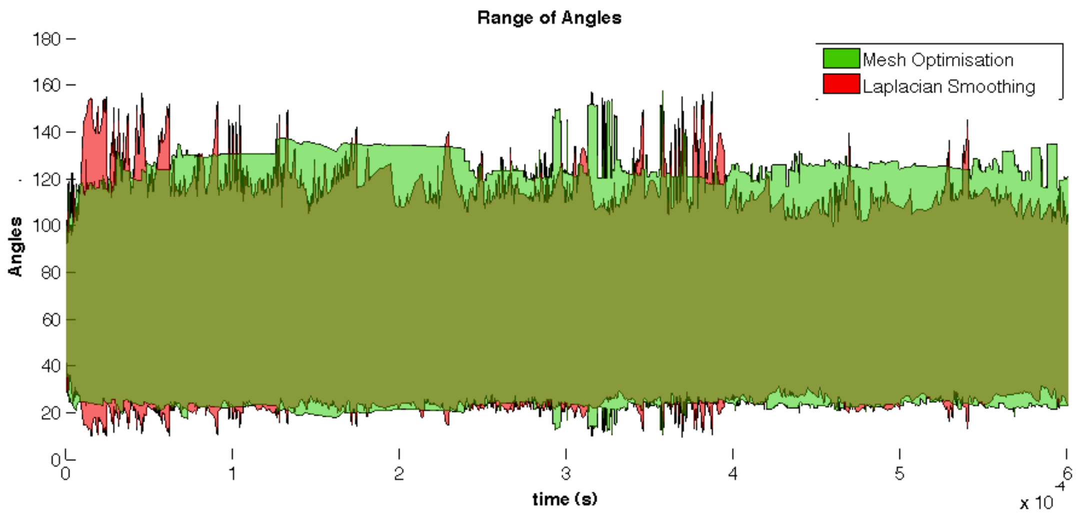


Figure 7.14: The range of angles in the droplet using Laplacian smoothing is shown in red and using quality preservation in green.

Volume change

The problem of volume change has been greatly reduced using quality preservation versus Laplacian smoothing. In fact with quality preservation the change in volume is 65 times less than that observed using Laplacian smoothing.

Stiffness matrix conditioning

Another goal of mesh quality optimisation is to reduce stiffness matrix conditioning as it was shown in Chapter 2 that a higher quality mesh correlates with a better conditioned stiffness matrix. However, this only refers to the physical portion of the stiffness matrix. The stiffness matrix in this analysis is composed of a physical portion and a mesh quality portion. Mesh optimisation introduces a coupling term between the x and y components of mesh quality entries in the stiffness matrix which is not present with Laplacian smoothing. This in turn increases the stiffness matrix conditioning. The size of the stiffness matrix in this simulation, although small by many standards, is still far too large to calculate the conditioning of at intermediate time-steps, thus the difference in conditioning of the stiffness matrices for both simulations cannot be quantified.

The optimisation of surface nodes

The results presented in the previous section show that the use of mesh quality optimisation has drastically increased the time taken for this simulation to run. Clearly, adding the additional computational cost associated with using the constrained optimisation algorithm presented in Chapter 5 is not feasible at this time, thus, it is recommended that efforts are focussed on alleviating the current efficiency issues before additional expensive operations are added.

7.3.7.2 Conclusions

It is clear from the results presented in the previous section that replacing Laplacian smoothing with mesh quality optimisation was not the silver bullet hoped for: it did not solve the many issues that it was hoped it would and it introduced several issues of its own. Using mesh quality preservation, the time-step size reduced to 10% of its previous value. If all the other indications were positive, this could be deemed acceptable as parallelisation and other performance boosts as described above could be implemented to negate the effect of the small time-step. However, several issues must firstly be investigated. In order to comment further on the amount of re-meshing performed with mesh quality preservation, more sophisticated node insertion/deletion algorithms must be implemented as well as tight control placed on the use of edge flips. This is because it is suspected that many unnecessary re-meshing operations are occurring. The current re-meshing algorithm is not at all compatible with analysis requiring non-uniform mesh refinement. This is yet another reason to change them. The vast differences in the shape assumed by the droplet throughout both simulations may be due to the degree of re-meshing occurring with quality preservation as well as a mesh dependence in the unmodified simulation. Four clear positives may be drawn from the use of mesh quality preservation. Firstly, the artificial mesh viscosity parameter has been eliminated. This is a clear advantage of this algorithm due to the many problems linked to this parameter. Secondly, the amount of volume change in the fluid droplet has been greatly decreased. Thirdly, this technique is transferable to three dimensions whereas Laplacian smoothing is not. Finally, this technique is also compatible with analysis which require non-uniform mesh refinement. However, it must be concluded that Laplacian smoothing is quite effective in two-dimensional, convex domains and may in fact be the best solution for such simulations.

Chapter 8

Conclusions

Many problems in Finite Element (FE) simulations have been tracked to poor quality meshes. In this thesis powerful algorithms to optimise these poor meshes have been developed and implemented. The impact of poor quality meshes is wide ranging in an analysis. For example, if a function is approximated on a poor quality mesh, many interpolation and gradient interpolation errors are introduced, negatively affecting the accuracy of the simulation. Stiffness matrices assembled from poor quality meshes may suffer from very poor conditioning as this is strongly linked the underlying mesh. Both of these consequences of poor quality meshes are discussed in depth in Chapter 2. The use of anisotropic elements and methods to calculate the ideal element size, shape and orientation at each location in a domain are also introduced.

As part of this thesis, a very effective mesh optimisation methodology has been developed and implemented. The powerful algorithms used as part of this methodology are described in detail in Chapter 3. The results obtained from optimising meshes based on this methodology are presented in Chapter 4. It is clear that the log-barrier objective function is a very powerful tool which should be used in all instances of mesh optimisation, both due to its effectiveness and its efficiency. Combined with each quality measure discussed in this thesis, it produced significantly better meshes in less time than any other objective function tried. This is a key recommendation of this thesis.

Second derivatives (Hessians) for the Area/Volume-Length quality measure, and Sine quality measure have been derived. No references to these were found in the literature. Also, the gradient and Hessian of these quality measures have been derived with respect to a pseudo gradient of deformation that relates the shape of an improved element to its original shape. Again no reference to this has been found and the derivation of these measures in this form represents a significant reduction in the effort required to derive and implement them in a computer code as well as providing an expression for them which is consistent with FE equations. A quality measure which is size, shape and rotation variant, thus suitable for optimising meshes based on an underlying anisotropic metric field derived from solution error estimates, is introduced in Chapter 3. This quality measure may be used with the mesh optimisation methodology developed in this chapter.

In Chapter 5, the challenges associated with optimising mesh nodes which lie on the boundary of the domain are discussed. Nodes which lie on planar surfaces and straight edges do not present significant difficulties, however, nodes which lie on non-planar edges or surfaces are much more difficult to optimise. Surface quadrics, the technique developed by Dr. Bryan Klingner for optimising such nodes whilst proven to be effective has the disadvantage of modifying the domain shape and volume. This disadvantage renders this technique unsuitable for use in many FE simulations. The technique presented introduces constraints to the optimisation process which preserving both domain shape and volume. The effectiveness of this novel approach has been clearly demonstrated in Chapter 6.

The techniques developed were applied to an existing monotonic Arbitrary Lagrangian Eulerian (ALE) FE simulation of surface tension on a micro-fluid droplet as a case study. No examples in the literature were found of this being done previously. The mesh optimisation equations were coupled with the physical equations in order to determine the new nodal positions at each iteration. The existing implementation used Laplacian smoothing to find the new nodal positions and there were several problems with this and this also required the use of an artificial mesh viscosity parameter which was proving to be very problematic. Mesh optimisation solved many of these problems, however this came at the expense of a significantly reduced time step, and therefore increased computational time. Although the main aims were achieved, the reduction in time step made the use of mesh optimisation unfeasible. A technique to preserve the quality of the mesh instead of optimising it was proposed instead. This increased the time step to about 10% of its former value, a sacrifice which would be acceptable if all the project goals were achieved. However, this resulted in large amounts of the domain being re-meshed and a different evolution of the fluid droplet towards its equilibrium position. This could be due to a mesh dependence in the underlying implementation or it could be due to the numerical errors introduced by the re-meshing of the domain.

The example chosen was in fact a very difficult case which presented many challenges.

For example, the physical processes being modelled required the mesh the adapt to a rapidly changing domain. Many other physical processes which are modelled using similar techniques to this case study are much less demanding on the mesh so may benefit greatly from the proposed approach. The re-meshing algorithms were also found to be very largely biased towards Laplacian smoothing and thus much of the re-meshing was unnecessary when mesh optimisation was used. In fact the re-meshing algorithms used were found to be largely unfit for purpose. This also greatly reduced the effectiveness of the proposed approach. With more sophisticated re-meshing algorithms, this technique could prove to be very effective. The application of adaptive re-meshing and the use of a quality measure derived from solution error estimates is proposed as a possible solution to the issues encountered. A variation of the proposed method whereby the optimisation algorithms are only applied to the poorest quality elements may also remedy the issues associated with this algorithm. However, the proposed method does have several very strong advantages. The artificial mesh viscosity parameter, which proved very problematic is no longer required and the volume loss observed was significantly reduced. This method is also applicable to three dimensional analysis, whereas Laplacian smoothing has been shown to be very ineffective in three dimensions. This technique is also compatible with simulations involving selective mesh refinement, something which Laplacian smoothing does not support. However, in some simulations, involving two-dimensional convex domains, it was found that Laplacian smoothing may be more effective than mesh optimisation at maintaining mesh quality.

8.1 Future work

The inclusion of the algorithms developed as part of this thesis into the release version of Mesquite would greatly increase the impact of this research. This would also enable other researchers to build upon this work, thus advancing the field of mesh optimisation further.

Constraining nodes to the domain boundary is very expensive as shown by the results presented in Chapter 6. Investigation of this has shown that a matrix inversion used as part of the algorithm is the cause of this. Investigation of other techniques for applying the constraints or the use of a more efficient algorithm for inverting the matrix would be very useful.

The application of mesh optimisation techniques to an ALE simulation posed many problems. A great deal of work remains to be done in order to eliminate all of these problems such as the use of more sophisticated re-meshing algorithms and the investigation of the merits of focussing only on elements whose quality is below a certain threshold. This analysis could also greatly benefit from the use of adaptive re-meshing.

Appendix A

Derivation of $\mathcal{F}$ and $\mathcal{T}$

Derivation of $\mathcal{F}$

An expression for $\mathbf{H} \otimes \mathbf{H} : \mathbf{U} - \mathbf{H}\mathbf{U}^T\mathbf{H}$ which is linear in $\mathbf{U}$ is required in order to derive the second derivative of the AL/VL quality measure. Both $\mathbf{H}$ and $\mathbf{U}$ are second order tensors and $\mathbf{H} \otimes \mathbf{H}$ is a fourth order tensor. In the following sections, a fourth order tensor, $\mathbf{A}$, is printed as follows

$$\left[\begin{array}{ccc|cc|cc}
a_{0,0,0,0} & a_{0,0,0,1} & a_{0,0,0,2} & a_{0,1,0,0} & \cdots & \cdots & a_{0,2,0,2} \\
a_{0,0,1,0} & a_{0,0,1,1} & a_{0,0,1,2} & a_{0,1,0,0} & \cdots & \cdots & a_{0,2,1,2} \\
a_{0,0,2,0} & a_{0,0,2,1} & a_{0,0,2,2} & a_{0,1,0,0} & \cdots & \cdots & a_{0,2,2,2} \\
\hline
a_{1,0,0,0} & a_{1,0,0,1} & a_{1,0,0,2} & a_{1,1,0,0} & \cdots & \cdots & a_{1,2,0,2} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
a_{2,0,2,0} & a_{2,0,2,1} & a_{2,0,2,2} & a_{2,1,0,0} & \cdots & \cdots & a_{2,2,2,2}
\end{array}\right]$$

where $a_{i,j,k,l}$ is member i, j, k, l of $\mathbf{A}$ expressed using indicial notation. Therefore, $\mathbf{H} \otimes \mathbf{H} : \mathbf{U}$ may be expressed as follows:

$$\left[\begin{array}{ccc|ccc|ccc}
H_{00}H_{00} & H_{00}H_{01} & H_{00}H_{02} & H_{01}H_{00} & H_{01}H_{01} & H_{01}H_{02} & H_{02}H_{00} & H_{02}H_{01} & H_{02}H_{02} \\
H_{00}H_{10} & H_{00}H_{11} & H_{00}H_{12} & H_{01}H_{10} & H_{01}H_{11} & H_{01}H_{12} & H_{02}H_{10} & H_{02}H_{11} & H_{02}H_{12} \\
H_{00}H_{20} & H_{00}H_{21} & H_{00}H_{22} & H_{01}H_{20} & H_{01}H_{21} & H_{01}H_{22} & H_{02}H_{20} & H_{02}H_{12} & H_{02}H_{22} \\
\hline
H_{10}H_{00} & H_{10}H_{01} & H_{10}H_{02} & H_{11}H_{00} & H_{11}H_{01} & H_{11}H_{02} & H_{12}H_{00} & H_{12}H_{01} & H_{12}H_{02} \\
H_{10}H_{10} & H_{10}H_{11} & H_{10}H_{12} & H_{11}H_{10} & H_{11}H_{11} & H_{11}H_{12} & H_{12}H_{10} & H_{12}H_{11} & H_{12}H_{12} \\
H_{10}H_{20} & H_{10}H_{21} & H_{10}H_{22} & H_{11}H_{20} & H_{11}H_{21} & H_{11}H_{22} & H_{12}H_{20} & H_{12}H_{12} & H_{12}H_{22} \\
\hline
H_{20}H_{00} & H_{20}H_{01} & H_{20}H_{02} & H_{21}H_{00} & H_{21}H_{01} & H_{21}H_{02} & H_{22}H_{00} & H_{22}H_{01} & H_{22}H_{02} \\
H_{20}H_{10} & H_{20}H_{11} & H_{20}H_{12} & H_{21}H_{10} & H_{21}H_{11} & H_{21}H_{12} & H_{22}H_{10} & H_{22}H_{11} & H_{22}H_{12} \\
H_{20}H_{20} & H_{20}H_{21} & H_{20}H_{22} & H_{21}H_{20} & H_{21}H_{21} & H_{21}H_{22} & H_{22}H_{20} & H_{22}H_{12} & H_{22}H_{22}
\end{array}\right]$$

$$: \begin{bmatrix} \mathbf{U}_{00} & \mathbf{U}_{01} & \mathbf{U}_{02} \\ \mathbf{U}_{10} & \mathbf{U}_{11} & \mathbf{U}_{12} \\ \mathbf{U}_{20} & \mathbf{U}_{21} & \mathbf{U}_{22} \end{bmatrix} \tag{A.1}$$

and $\mathbf{H}\mathbf{U}^T\mathbf{H}$ may be expressed as follows:

$$\left[\begin{array}{ccc|ccc|ccc}
H_{00}H_{00} & H_{01}H_{00} & H_{02}H_{00} & H_{00}H_{01} & H_{01}H_{01} & H_{02}H_{01} & H_{00}H_{02} & H_{01}H_{02} & H_{02}H_{02} \\
H_{00}H_{10} & H_{01}H_{10} & H_{02}H_{10} & H_{00}H_{11} & H_{01}H_{11} & H_{02}H_{11} & H_{00}H_{12} & H_{01}H_{12} & H_{02}H_{12} \\
H_{00}H_{20} & H_{01}H_{20} & H_{02}H_{20} & H_{00}H_{21} & H_{01}H_{21} & H_{02}H_{21} & H_{00}H_{22} & H_{01}H_{22} & H_{02}H_{22} \\
\hline
H_{10}H_{00} & H_{11}H_{00} & H_{12}H_{00} & H_{10}H_{01} & H_{11}H_{01} & H_{12}H_{01} & H_{10}H_{02} & H_{11}H_{02} & H_{12}H_{02} \\
H_{10}H_{10} & H_{11}H_{10} & H_{12}H_{10} & H_{10}H_{11} & H_{11}H_{11} & H_{12}H_{11} & H_{10}H_{12} & H_{11}H_{12} & H_{12}H_{12} \\
H_{10}H_{20} & H_{11}H_{20} & H_{12}H_{20} & H_{10}H_{21} & H_{11}H_{21} & H_{12}H_{21} & H_{10}H_{22} & H_{11}H_{22} & H_{12}H_{22} \\
\hline
H_{20}H_{00} & H_{21}H_{00} & H_{22}H_{00} & H_{20}H_{01} & H_{21}H_{01} & H_{22}H_{01} & H_{20}H_{02} & H_{21}H_{02} & H_{22}H_{02} \\
H_{20}H_{10} & H_{21}H_{10} & H_{22}H_{10} & H_{20}H_{11} & H_{21}H_{11} & H_{22}H_{11} & H_{20}H_{12} & H_{21}H_{12} & H_{22}H_{12} \\
H_{20}H_{20} & H_{21}H_{20} & H_{22}H_{20} & H_{20}H_{21} & H_{21}H_{21} & H_{22}H_{21} & H_{20}H_{22} & H_{21}H_{22} & H_{22}H_{22}
\end{array}\right]$$

$$: \begin{bmatrix} \mathbf{U}_{00} & \mathbf{U}_{01} & \mathbf{U}_{02} \\ \mathbf{U}_{10} & \mathbf{U}_{11} & \mathbf{U}_{12} \\ \mathbf{U}_{20} & \mathbf{U}_{21} & \mathbf{U}_{22} \end{bmatrix} \tag{A.2}$$

There are many equal terms in both fourth order tensors. The following fourth order tensor is obtained by subtracting A.2 from A.1:

$$
\left[\begin{array}{ccc|ccc|ccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & H_{00}H_{11} - H_{01}H_{10} & H_{00}H_{12} - H_{02}H_{10} & H_{01}H_{10} - H_{00}H_{11} & 0 & H_{01}H_{12} - H_{02}H_{11} & H_{02}H_{10} - H_{00}H_{12} & H_{02}H_{11} - H_{01}H_{12} & 0 \\
0 & H_{00}H_{21} - H_{01}H_{20} & H_{00}H_{22} - H_{02}H_{20} & H_{01}H_{20} - H_{00}H_{21} & 0 & H_{01}H_{22} - H_{02}H_{21} & H_{02}H_{20} - H_{00}H_{22} & H_{02}H_{21} - H_{01}H_{22} & 0 \\
\hline
0 & H_{10}H_{01} - H_{11}H_{00} & H_{10}H_{02} - H_{12}H_{00} & H_{11}H_{00} - H_{10}H_{01} & 0 & H_{11}H_{02} - H_{12}H_{01} & H_{12}H_{00} - H_{10}H_{02} & H_{12}H_{01} - H_{11}H_{02} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & H_{10}H_{21} - H_{11}H_{20} & H_{10}H_{22} - H_{12}H_{20} & H_{11}H_{20} - H_{10}H_{21} & 0 & H_{11}H_{22} - H_{12}H_{21} & H_{12}H_{20} - H_{10}H_{22} & H_{12}H_{21} - H_{11}H_{22} & 0 \\
\hline
0 & H_{20}H_{01} - H_{21}H_{00} & H_{20}H_{02} - H_{22}H_{00} & H_{21}H_{00} - H_{21}H_{00} & 0 & H_{21}H_{02} - H_{22}H_{01} & H_{22}H_{00} - H_{20}H_{02} & H_{22}H_{01} - H_{21}H_{02} & 0 \\
0 & H_{20}H_{11} - H_{21}H_{10} & H_{20}H_{12} - H_{22}H_{10} & H_{21}H_{10} - H_{21}H_{10} & 0 & H_{21}H_{12} - H_{22}H_{11} & H_{22}H_{10} - H_{20}H_{12} & H_{22}H_{11} - H_{21}H_{12} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right]
$$

$$
: \begin{bmatrix} \mathbf{U}_{00} & \mathbf{U}_{01} & \mathbf{U}_{02} \\ \mathbf{U}_{10} & \mathbf{U}_{11} & \mathbf{U}_{12} \\ \mathbf{U}_{20} & \mathbf{U}_{21} & \mathbf{U}_{22} \end{bmatrix} \qquad \text{(A.3)}
$$

Each of the terms in this tensor is the determinant of a sub-tensor of $\mathbf{H}$. Therefore, this may be simplified as follows:

$$\left[\begin{array}{ccc|ccc|ccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \det H^s_{(2,2)} & \det H^s_{(1,2)} & -\det H^s_{(2,2)} & 0 & \det H^s_{(0,2)} & -\det H^s_{(1,2)} & -\det H^s_{(0,2)} & 0 \\
0 & \det H^s_{(2,1)} & \det H^s_{(1,1)} & -\det H^s_{(2,1)} & 0 & \det H^s_{(0,1)} & -\det H^s_{(1,1)} & -\det H^s_{(0,1)} & 0 \\
\hline
0 & -\det H^s_{(2,2)} & -\det H^s_{(1,2)} & \det H^s_{(2,2)} & 0 & -\det H^s_{(0,2)} & \det H^s_{(1,2)} & \det H^s_{(0,2)} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \det H^s_{(2,0)} & \det H^s_{(1,0)} & -\det H^s_{(2,0)} & 0 & \det H^s_{(0,0)} & -\det H^s_{(1,0)} & -\det H^s_{(0,0)} & 0 \\
\hline
0 & -\det H^s_{(2,1)} & -\det H^s_{(1,1)} & \det H^s_{(2,1)} & 0 & -\det H^s_{(0,1)} & \det H^s_{(1,1)} & \det H^s_{(0,1)} & 0 \\
0 & -\det H^s_{(2,0)} & -\det H^s_{(1,0)} & \det H^s_{(2,0)} & 0 & -\det H^s_{(0,0)} & \det H^s_{(1,0)} & \det H^s_{(0,0)} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right]$$

$$: \begin{bmatrix} \mathbf{U}_{00} & \mathbf{U}_{01} & \mathbf{U}_{02} \\ \mathbf{U}_{10} & \mathbf{U}_{11} & \mathbf{U}_{12} \\ \mathbf{U}_{20} & \mathbf{U}_{21} & \mathbf{U}_{22} \end{bmatrix} \tag{A.4}$$

where $\mathbf{H}^s_{(a,b)}$ is sub-tensor of $\mathbf{H}$ with column a and row b removed. Using indicial notation, this may be expressed as:

$$\mathcal{F} = \det(\mathbf{H}^s_{(3-j-l),(3-i-k)})\gamma_{i,k}\gamma_{j,l} \tag{A.5}$$

where

$$\gamma_{a,b} = \begin{cases} 1, & \text{if } a \leq b \\ -1, & \text{if } b \leq a \\ 0, & \text{if } a = b \end{cases}$$

Derivation of $\mathcal{T}$

Similarly in the derivation of the second derivatives of the AL/VL quality measures with respect to the gradient of deformation tensor, an expression for $\mathbf{UA}$ in the form $\mathcal{T}(\mathbf{A}) : \mathbf{U}$ is required, where $\mathbf{U}$ and $\mathbf{A}$ are second order tensors and $\mathcal{T}(\mathbf{A})$ is a fourth order tensor. The following expression when contracted is equal to **UA**

$$\left[\begin{array}{ccc|ccc|ccc}
A_{00} & A_{10} & A_{20} & A_{01} & A_{11} & A_{21} & A_{02} & A_{12} & A_{20} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
A_{00} & A_{10} & A_{20} & A_{01} & A_{11} & A_{21} & A_{02} & A_{12} & A_{20} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
A_{00} & A_{10} & A_{20} & A_{01} & A_{11} & A_{21} & A_{02} & A_{12} & A_{20}
\end{array}\right] : \left[\begin{array}{ccc}
\mathbf{U}_{00} & \mathbf{U}_{01} & \mathbf{U}_{02} \\
\mathbf{U}_{10} & \mathbf{U}_{11} & \mathbf{U}_{12} \\
\mathbf{U}_{20} & \mathbf{U}_{21} & \mathbf{U}_{22}
\end{array}\right] \tag{A.6}$$

References

[1] Jonathan Richard Shewchuk. What is a Good Linear Element ? Interpolation , Conditioning , and Quality Measures. In *Proceedings of the 11th International Meshing Roundtable*, 2002.

[2] Jessica Schoen. Robust, Guaranteed-Quality Anisotropic Mesh Generation. Master's thesis, University of California, Berkeley, 2009.

[3] J Donea, A Huerta, J Ponthot, and A Rodr. *Encyclopedia of Computational Mechanics*. John Wiley & Sons, Ltd, Chichester, UK, November 2004.

[4] Weizhang Huang. Metric tensors for anisotropic mesh generation. *Journal of Computational Physics*, 204(2):633 – 665, 2005.

[5] Xiangrong Li, Mark S. Shephard, and Mark W. Beall. 3d anisotropic mesh adaptation by mesh modification. *Computer Methods in Applied Mechanics and Engineering*, 194(48):4915 – 4950, 2005. Unstructured Mesh Generation.

[6] R. Löhner. *Applied Computational Fluid Dynamics Techniques: An Introduction Based on Finite Element Methods*. Wiley, 2001.

[7] Hassan O. and E. J. Probert. Grid control and adaptation. In Joe F. Thompson, Bharat K. Soni, and Nigel P. Weatherhill, editors, *Handbook of Grid Generation*. CRC Press, 1999.

[8] Klaus Jürgen Bathe and Arthur P. Cimento. Some practical procedures for the solution of nonlinear finite element equations. *Computer Methods in Applied Mechanics and Engineering*, 22(1):59 – 85, 1980.

[9] Metric tensor. `http://mathworld.wolfram.com/MetricTensor.html`. Accessed: 21-06-2014.

[10] P. Frey and P.L. George. *Mesh Generation*. ISTE. Wiley, 2010.

[11] P.J. Frey and F. Alauzet. Anisotropic mesh adaptation for {CFD} computations. *Computer Methods in Applied Mechanics and Engineering*, 194(49):5068 – 5082, 2005. Unstructured Mesh Generation.

[12] Francois Courty, David Leservoisier, Paul-Louis George, and Alain Dervieux. Continuous metrics and mesh adaptation. *Applied Numerical Mathematics*, 56(2):117 – 145, 2006.

[13] Xiangrong Li, Jean-Francois Remacle, Nicolas Chevaugeon, and Mark S. Shephard. Anisotropic Mesh Gradient Control. In *Proceedings of the 13th International Meshing Roundtable*, 2004.

[14] Xiangmin Jiao, Andrew Colombi, Xinlai Ni, and John C. Hart. Anisotropic mesh adaptation for evolving triangulated surfaces. In *Proceedings of the 15th International Meshing Roundtable*, 2006.

[15] Gustavo C. Buscaglia and Enzo A. Dari. Anisotropic mesh optimization and its application in adaptivity. *International Journal for Numerical Methods in Engineering*, 40(22):4119–4136, 1997.

[16] Ward Cheney and David Kincaid. *Numerical Mathematics and Computing*. International Thomson Publishing, 4th edition, 1998.

[17] Patrick Knupp, Lori Freitag-Diachin, and Boyd Tidwell. Mesh quality improvement toolkit user's guide. Technical report, Sandia National Laboratories, 2012.

[18] B Klingner. *Tetrahedral mesh Improvement*. PhD thesis, University of California at Berkeley, November 2008.

[19] Patrick Knupp, Lori Freitag-Diachin, and Jason Kraftcheck. Mesh Quality Improvement Toolkit User's Guide. Technical report, Sandia National Laboratories, 2008.

[20] Interoperable Technologies for Advanced Petascale Simulations (ITAPS), 2010.

[21] Timothy J. Tautges, Jason A. Jason A. Kratfcheck, Brandon M. Smith, and Hong-Jun Kim. Mesh Oriented Database Version 4.0 User's Guide, June 2011.

[22] Lori a. Freitag, Mark Jones, and Paul Plassmann. An Efficient Parallel Algorithm for Mesh Smoothing. In *Proceedings of the 4th International Meshing Roundtable*, 1995.

[23] Lori a. Freitag and Carl Ollivier-Gooch. Tetrahedral mesh improvement using swapping and smoothing. *International Journal for Numerical Methods in Engineering*, 40(21):3979–4002, November 1997.

[24] Sandia National Laboratories. Cubit 13.2 User Documentation, 2012.

[25] Richard E. Ladner. K-D Trees. Technical report, University of Washington, 2002.

[26] Lori A Freitag and Carl Ollivier-Gooch. A Comparison of Tetrahedral Mesh Improvement Techniques. In *Proceedings of the 5th International Meshing Roundtable*, 1996.

[27] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[28] Qiang Du and Max Gunzburger. Grid generation and optimization based on centroidal voronoi tessellations. *Applied Mathematics and Computation*, 133(2):591 – 607, 2002.

[29] Patrick M. Knupp. Algebraic Mesh Quality Metrics. *SIAM Journal on Scientific Computing*, 23(1):193, 2001.

[30] Todd Munson. Mesh Shape-Quality Optimization Using the Inverse Mean-Ratio Metric. *Mathematical Programming*, 110(3):561–590, 2004.

[31] Javier (Swansea University) Bonet and Richard (Swansea University) Wood. *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press, Cambridge, 2nd edition, 2008.

[32] John Barlow. More on optimal stress points, reduced integration, element distortions and error estimation. *International Journal for Numerical Methods in Engineering*, 28(7):1487–1504, 1989.

[33] M Scherer, R Denzer, and P Steinmann. On a solution strategy for energy-based mesh optimization in finite hyperelastostatics. *Computer Methods in Applied Mechanics and Engineering*, 197(6-8):609–622, January 2008.

[34] François Labelle and Jonathan Richard Shewchuk. Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. *ACM Transactions on Graphics*, 26(3):57.1–57.10, July 2007. Special issue on Proceedings of SIGGRAPH 2007.

[35] L. Kaczmarczyk and C.J. Pearce. Efficient numerical analysis of bone remodelling. *Journal of the Mechanical Behavior of Biomedical Materials*, 4(6):858 – 867, 2011. Bone Remodeling.

[36] Lukasz Kaczmarczyk, Mohaddeseh Mousavi Nezhad, and Chris Pearce. Three-dimensional brittle fracture: configurational-force-driven crack propagation, 2014. http://arxiv.org/abs/1304.6136.

[37] Carl Olivier-Gooch. GRUMMP - Generation and Refinement of Unstructured Mixed Element Mesh in Parallel.

[38] Bryan Matthew Klingner and Jonathan Richard Shewchuk. Agressive tetrahedral mesh improvement. In *Proceedings of the 16th International Meshing Roundtable*, pages 3–23, Seattle, Washington, October 2007.

[39] Divergence theorem. `http://mathworld.wolfram.com/DivergenceTheorem.html`. Accessed: 3-07-2014.

[40] Mark Ainsworth. Essential boundary conditions and multi-point constraints in finite element analysis. *Computer Methods in Applied Mechanics and Engineering*, 190(48):6323–6339, September 2001.

[41] Lukasz Kaczmarczyk, Chris J. Pearce, and Nenad Bicanic. Scale transition and enforcement of rve boundary conditions in second-order computational homogenization. *International Journal for Numerical Methods in Engineering*, 74(3):506 – 522, 2008.

[42] R. J. D. Mackenzie. *A Computational Framework For Modelling Micro-scale Fluids Subject to Surface Acoustic Waves.* PhD thesis, University of Glasgow, August 2014.

[43] Arbtip Dheeravongkit and Kenji Shimada. Inverse adaptation of a hex-dominant mesh for large deformation finite element analysis. *Computer-Aided Design*, 39(5):427 – 438, 2007. Geometric Modeling and Processing 2006 Geometric Modeling and Processing 2006.

[44] Harm Askes, Ellen Kuhl, and Paul Steinmann. An ALE formulation based on spatial and material settings of continuum mechanics. Part 2: Classification and applications. *Computer Methods in Applied Mechanics and Engineering*, 193(39-41):4223–4245, October 2004.

[45] W. K. Liu & B. Moran T. Belytschko. *Nonlinear Finite Elements for Continua and Structures.* John Wiley & Sons, 2000.

[46] Carlos A Felippa. Review of Continuum Mechanics. In *Nonlinear Finite Element Methods*, chapter 7. University of Colorado at Boulder, 2012.

[47] Patrick Knupp, Len G. Margolin, and Mikhail Shashkov. Reference jacobian optimization-based rezone strategies for arbitrary lagrangian eulerian methods. *Journal of Computational Physics*, 176(1):93 – 128, 2002.

[48] S. Giuliani. An algorithm for continuous rezoning of the hydrodynamic grid in arbitrary lagrangian-eulerian computer codes. *Nuclear Engineering and Design*, 72(2):205 – 212, 1982.

[49] J. Sarrate and A. Huerta. An improved algorithm to smooth graded quadrilateral meshes preserving the prescribed element size. *Communications in Numerical Methods in Engineering*, 17(2):89–99, 2001.

[50] Water dripping. Still images taken from a video produced by Julien Reboud, John Cooper, Rab Wilson et al. Bioelectronics Group, University of Glasgow.

[51] D. J. Acheson. *Elementary Fluid Dynamics*. Oxford University Press, 1st edition, 1990.

[52] Richard Fitzpatrick. Surface tension. `http://farside.ph.utexas.edu/teaching/336L/Fluid.pdf`. Accessed: 21-07-2014.

[53] Steve Tobias. The Material Derivative. In *MATH3454*, chapter 2. University of Leeds, 2014.

[54] Leibniz integral rule. `http://mathworld.wolfram.com/LeibnizIntegralRule.html`. Accessed: 29-06-2014.

[55] G. K. Batchelor. Equations governing the motion of a fluid. In *An Introduction to Fluid Dynamics*, pages 131–173. Cambridge University Press, 2000. Cambridge Books Online.

[56] M. Kit, Kevin. Mse 443: Polymer processing. Accessed: 12-06-2014.

[57] Andrew Hazel. In *Numerical Computation*, chapter Incompressible Fluid Flow. University of Manchester. Accessed: 03-07-2014.

[58] P. H. Saksono and D. Peric. On finite element modelling of surface tension variational formulation and applications part i: Quasistatic problems. *Computational Mechanics*, 38(3):265–281, 2006.

[59] Ross Mackenzie, Lukasz Kaczmarczyk, and Chris Pearce. An axisymmetric pressure stabilised predictive model of surface tension in micro-fluids. In *Proceedings of the Interational Conference on Computational Mechanics (CM13)*, 2013.

[60] Volker John and Joachim Rang. Adaptive time step control for the incompressible navier stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 199(9–12):514–524, 2010.

[61] Jonathan Richard Shewchuk. Delaunay Refinement Algorithms for Triangular Mesh Generation. *Computational Geometry: Theory and Applications*, 22(1-3):21–74, 2002.

[62] P. H. Saksono, W. G. Dettmer, and D. Peric. An adaptive remeshing strategy for flows with moving boundaries and fluidstructure interaction. *International Journal for Numerical Methods in Engineering*, 71(9):1009–1050, 2007.

[63] R. Clint Whaley and Antoine Petitet. Minimizing development and maintenance costs in supporting persistently optimized BLAS. *Software: Practice and Experience*, 35(2):101–121, February 2005.

[64] Intel math kernel library. http://software.intel.com/en-us/intel-mkl.