Mesh Improvement Methodology for 3D Volumes with Non-planar Surfaces

Alan Kelly, Lukasz Kaczmarczyk, and Chris Pearce

University of Glasgow, Glasgow, Scotland
a.kelly.2@research.gla.ac.uk,
{Lukasz.Kaczmarczyk,Chris.Pearce}@glasgow.ac.uk

Summary. A mesh improvement methodology is presented which aims to improve the quality of the worst elements in 3D mesh with non-planar surfaces which cannot be improved using traditional methods. A numerical optimisation algorithm which specifically targets the worst elements in the mesh but is a smooth functions of nodal positions is introduced. A method of moving nodes on curved surfaces whilst maintaining the domain geometry and preserving mesh volume is proposed and implemented and is shown to be very effective at improving mesh which traditional mesh improvers cannot improve.

1 Introduction

In the context of the Finite Element Method (FEM), high quality meshes can be crucial to obtaining accurate results. The quality of an element can be described as a numerical measure which estimates the effect that the shape of an element will have on the accuracy of an analysis, [1]. It can be shown that poor quality elements can result in both discretisation 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.

The field of mesh optimisation is complex and has now become an area of research in its own right. 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 mesh quality is the objective function and the constraints include, for example, the domain geometry, maximum element size, etc.

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 for the analyst, 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. In doing this, we are attempting to simplify a very complex process; this paper will explain the problems encountered and the solutions to these problems.

1.1 Motivation

The motivation for this project has come from the problems encountered by the authors and their colleagues. 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 geometries which consequently require an evolving mesh that will enable the problems to be solved accurately and efficiently. For complex three-dimensional geometries, automatic mesh generators do not always create meshes of sufficient quality to ensure a sufficient level of accuracy in the solution. This is further complicated by the need to have an adapting mesh that can resolve the evolving geometry. In the problems mentioned above, many 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 authors. Therefore, the decision was then made to develop a new set of tools, either via the modification of existing open-source software or through the development of new tools.

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. 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 a Eulerian manner where the mesh is fixed and the continuum moves with respect to the mesh or in some arbitrary combination of these, [2]. In certain micro-fluids analysis, the positions of the surface nodes are determined using a Lagrangian formulation and the interior nodes using an Eulerian formulation. The surface node positioning is determined by the physics of the problem being analysed and therefore any alteration of their positioning must be compatible with the physics of the problem, i.e. the geometry and the volume of the domain must be preserved. Therefore a method of improving mesh quality by moving surface nodes but without changing the geometry or volume of the domain is necessary. Such a method has been developed and is described in Section 3.3.

1.2 Implementation

This work was implemented using the Mesh Quality Improvement Toolkit (Mesquite) as a platform, [3]. The architecture of this library makes it ideal

to use as a base for the development and testing of new algorithms. In addition, the Mesh Oriented Database (MOAB) [4] was used for mesh operations not available in Mesquite, CBLAS and Lapack were used for numerical operations. Both Mesquite's native algorithms [3] and Stellar [5] a mesh optimisation program, were used to assess the results obtained. Stellar, whilst a very powerful program, has restrictions that mean it has limited application to the kind of problems that motivated this work. The main issue is that the user has limited control over the optimisation process. However, as an academic package, it is very powerful as it shows what can be achieved in mesh optimisation, thus making it an ideal tool for comparison of results.

1.3 Optimisation-Based Mesh Smoothing

Mesh smoothing is the process of improving mesh quality without changing the mesh topology [6]. There are many existing mesh smoothing algorithms, the most famous of which is Laplacian smoothing. Laplacian smoothing involves moving a vertex to the average of its connected neighbours' positions and is applied to each mesh vertex in sequence and is repeated several times. It has been shown to be somewhat effective with 2D triangular meshes, but is much less effective in 3D [5]. Although Laplacian smoothing is computationally cheap, there is no guarantee of mesh improvement. It is even possible that inverted elements will be created [7] when the domain is not convex [2]. Much more sophisticated mesh smoothing algorithms have been developed which are based on numerical optimisation techniques. Techniques such as these are referred to as optimisation-based smoothers. These methods require a means of expressing the quality of an element numerically and of combining the qualities of every element in the mesh into a single numerical value. A numerical measure which effectively captures mesh qualities requirements is described in Section 2.1.

Another requirement of numerical optimisation techniques is the use of an objective function, which is a means of combining the qualities of a group of elements into a scalar-valued function. For example, one could express the quality of a mesh as the sum of the qualities of every element. This objective function would then be minimised, or maximised, depending on the choice of quality measure, to improve the quality of the mesh. However, as previously stated, one poor element may render a problem unsolvable. A simplistic objective function such as the one described above would be very good at improving average element quality, but would not improve the worst element since one poor quality element does not stand out in this case. Such an objective function may even invert some elements as one negative number may not sufficiently influence the objective function. Therefore, it is desirable to use an objective function that targets the quality of the worst element.

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 element's qualities, 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 [8], which enabled the improvement of the worst element in a mesh. This algorithm achieved very high quality results and it is this algorithm which is utilised in Stellar [5]. This approach works by calculating the search directions for the nodes of the worst element and attempting to predict the distance each node may be moved in this direction until the element is no longer the worst element in the mesh. As element quality is a function of nodal positions, a first order Taylor Series of the quality of every affected element may be used to approximate the point at which the element being improved is no longer the worst element.

A genuinely smooth objective function which penalises the worst element in a mesh to such an extent that the improvement process focuses on this element should, in theory, yield better results in a shorter analysis time since the objective function contains information about the qualities of all elements so there is no requirement to approximate the point at which the quality of the worst element changes. An objective function which meets the smoothness criteria and which also adequately penalises the worst element is described in Section 2.2.

2 Mesh Improvement Methodology

The focus of this research is to produce practical tools which can be used in a variety of problems. These tools must be easy to use, powerful, efficient and easy to integrate into existing codes. In the case of domains which are constantly evolving, it is often necessary to perform mesh improvement during an analysis. Therefore, algorithms which can quickly improve the worst elements of large meshes are required. Two components are required to achieve this: the development of a element quality modification function and the identification of the worst elements in a mesh.

2.1 Quality Measure

Finding a suitable quality measure that provides an accurate estimate of an element's effects in terms of discretisation/interpolation error and stiffness

matrix condition is quite difficult and is a very active area of research in itself. There are many measures in existence and these may be further studied in [1]. It was decided to use the Volume-Length quality measure as it has been shown to be very effective at improving stiffness matrix conditioning and interpolation errors [1],[5]. Also, since this measure is a smooth function of vertex positions, the function and its gradient are straightforward and computationally cheap to calculate. This measure is normalised so that an equilateral element has quality 1 and a degenerate element (zero volume) has quality 0.

$$q = 6\sqrt{2}\frac{V}{l_{rms}^3} \tag{1}$$

where V is the volume of a tetrahedral element, l_{rms} is the root mean square of the element's edge lengths and q is the element quality.

2.2 Worst Element Improvement Algorithms

The Log-Barrier Objective Function

This section describes an objective function which both satisfies the smoothness criteria described in Section 1.3 and punishes the worst element in the mesh. This is achieved by expressing the quality of every element as a function of the worst element, equation (2), and is referred to as a log-barrier function. This equation was developed iteratively until a function which had the required characteristics was found, i.e. one which punishes harshly the worst element. It has been found that choosing the barrier constant term C in the range 0.75-0.95 is most effective. The optimisation process starts with a lower value of C and becomes more aggressive with C increasing. Smaller values of C tend to increase average element quality as the worst elements are not punished as harshly, whereas higher C tends to improve the quality of the worst element. The gradient and Hessian of the quality measure are also shown in Equations (3) and (4). Expressing the quality of a group of elements in this manner ensure that the optimisation process is always focused on the worst element.

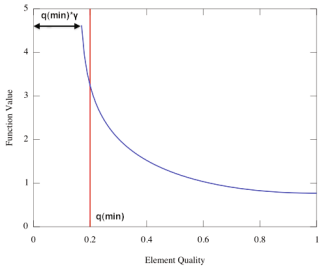
$$I = \frac{q^2}{2(1-\gamma)} - log(q-\gamma) \tag{2}$$

$$\nabla I = (\frac{q}{1-\gamma} - \frac{1}{q-\gamma})\nabla q \tag{3}$$

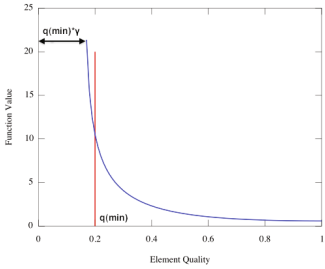
$$\nabla^2 I = \nabla q[\frac{1}{1-\gamma} - \frac{1}{(q-\gamma)^2}]\nabla q + [\frac{q}{1-\gamma} - \frac{1}{q-\gamma}]\nabla^2 q \tag{4}$$

Where q is the element quality, γ the barrier which is a constant, C, times the quality of the worst element in the mesh, q_{min}, ∇q the gradient of the quality measure and $\nabla^2 q$ the Hessian of the quality measure.

Figure 1 shows the Log-Barrier function graphically. It can be seen that the function value rapidly increases as the quality of the element reduces, thus achieving our aim of punishing the worst element. Figure 1b demonstrates that this effect is further magnified by squaring the Log-Barrier function.



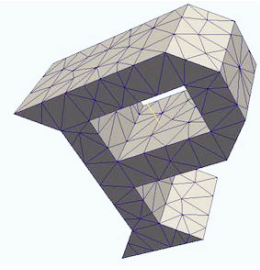
(a) Log-Barrier



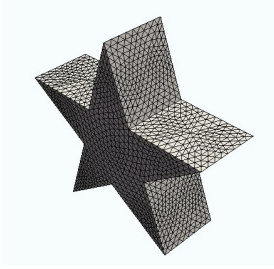
(b) Log-Barrier Squared

Fig. 1. Plot of the Log-Barrier function (equilateral element has quality 1 and a degenerate element (zero volume) has quality 0) (C=0.8)

The optimisation process is repeated several times for the Log-Barrier function as the parameters change as the worst element changes. Unlike traditional mesh optimisation, which is allowed to run until it is deemed to have converged or some other termination criteria has been achieved, Log-Barrier optimisation performs one pass over each patch and then the worst element quality is re-calculated and γ updated to reflect this. This ensures that the optimisation process is always aggressively tackling the worst element.



(a) 'P', Klingner [5]



(b) 'Star'

Fig. 2. Mesh used for Testing

The Log-Barrier function also has several other very useful features. It comes with an invertibility guarantee - if the initial mesh is valid, that is to say a mesh without any inverted or negative volume elements, is input, no inverted elements will be created. If the initial mesh is invalid, the Log-Barrier function can untangle it as the quality is always chosen to be worse than the worst element. The form of the Log-Barrier function adopted here is different to the form describe by [9]. The differences between both methods should be investigated to compare their respective merits.

2.3 Comparison of Measures

Mesh optimisation was performed on the two meshes shown in Figure 2, 'P' containing 926 elements and 'Star' containing 34880 elements. The Log-Barrier function combined with the Volume-Length quality measure was compared with Mesquite's Ideal Weight Inverse Mean Ratio quality measure combined with an Infinity Norm objective function. The Infinity Norm was found to be the most effective objective function in Mesquite at improving the worst element in the mesh. Each smoothing algorithm was run until convergence, with no restrictions on time, to measure the highest quality each smoother could achieve. The results may be seen in Table 1.

Table 1. Highest Quality Achievable

Mesh	Smoother	Highest Quality
'P'	Inverse-Mean	0.47391
	Log-Barrier	0.48496
	Stellar	0.466157
'Star'	Inverse-Mean	0.42576
	Log-Barrier	0.44392
	Stellar	0.359

As may be seen for both meshes, the Log-Barrier function achieved greater improvement, albeit slight. This demonstrates that the Log-Barrier function is as effective as the existing mesh smoothers in Mesquite. It can also be seen in Table 1 that the mesh was also improved using Stellar as well. For the purposes of this comparison, Stellar's topological transformation functionality was disabled so as to provide a meaningful comparison. In both cases, the other two smoothers performed slightly better.

It is also important to note that Stellar can achieve significantly better results (maximum quality of 0.714 for 'Star' mesh) when all of its functionality is enabled. However, this results in a mesh with approximately half the number of elements as the original mesh. This demonstrates the potential improvement that may be achieved by performing topological changes during

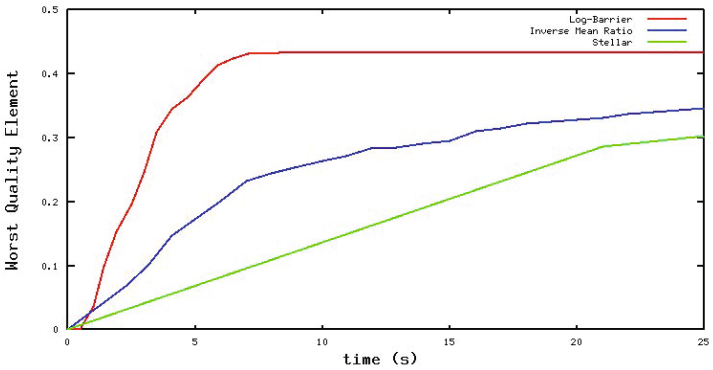


Fig. 3. Comparison of time versus quality achieved for the Log-Barrier function, Inverse Mean Ratio and Stellar

the mesh optimisation process, although in practice it would be necessary to restrict the reduction in the number of elements. Although not the subject of this paper, it is intended to add topological changes to Mesquite at a later date.

The performance of the Log-Barrier smoother in terms of mesh quality alone is not dramatic. However, as may be seen in Figure 3, it is significantly quicker than both other smoothers. This effect is further emphasised when it is combined with the selective patch-improvement procedure described in the next section.

2.4 Patch Improvement

As we are only interested in improving the worst elements in a mesh, it is inefficient to operate on all elements. Therefore a modified form of patch-based improvement is used. Patch improvement involves breaking the mesh up into smaller mesh patches and improving each patch individually. Since we only wish to operate on the worst elements, we select the patches containing these elements and improve them. Algorithm 1 explains how this is achieved. This method is also very effective at reducing the time taken to improve a mesh using other, non-barrier quality measures such as the Inverse Mean Ratio. However, in this case, a different approach is necessary whereby a target mesh quality is chosen by the user. All elements with quality worse than the target quality are selected for improvement. It has been found that the most efficient means of achieving this is to work towards the target quality, i.e. perform several iterations on elements with qualities much worse than the target and gradually work towards the target. Figure 4 shows the time taken to improve 'Star'. It is clear how big an impact this method has when these results are compared to the results presented in Figure 3.

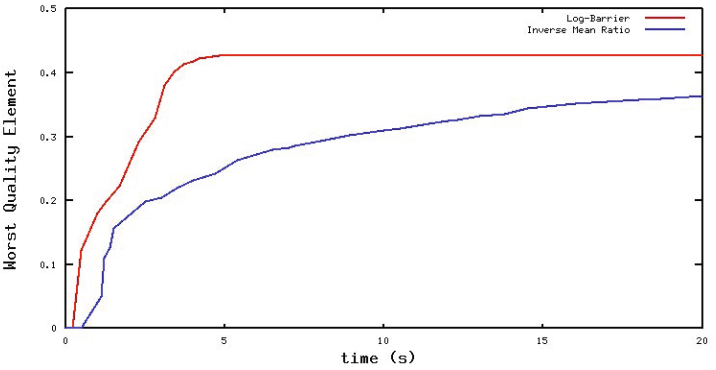


Fig. 4. Comparison of time versus quality achieved for the Log-Barrier function and Inverse Mean Ratio using patch based improvement

Algorithm 1. Patch Improvement

Assess quality of every element (q) and calculate standard deviation (σ)
Identify elements whose quality is $q_{min} + 3\sigma$ or worse.
Identify all the free nodes contained in this sub-group of elements and create mesh patches with a layer of two elements deep around these nodes.
Loop through each patch and improve.

3 Surface Mesh Optimisation

Meshes associated with complex domains often have complex boundaries. If the worst element of a mesh lies on a boundary, then it becomes difficult to improve the mesh without changing the geometry of the domain. Several methods have been developed to tackle this [3]. If the domain of the boundary is a straight line or a planar surface there are two possible options. Mesquite provides built in functionality which "snaps" nodes which have been moved from either a planar surface or a straight line back onto the correct domain. Another means of achieving this is to project the gradients of the quality measure onto the surface. Both of these methods are effective for surfaces which may be mathematically defined, i.e. an equation may be derived for the surface, but are not sufficient for more complex ones which cannot be defined mathematically.

3.1 Surface Quadrics

Klingner [5] developed a method of using surface quadrics as part of his PhD project and implemented this into Stellar. This method assigns an error to a

vertex which has been moved based on how far it has moved from the planes created by the original triangular faces that adjoined it [5]. This approach is summarised here. Let P be the set of planes created by the triangular faces adjoining a vertex, v. The quadric error for a point f relative to v is defined as

$$Q_v(x) = \Sigma\delta_i(x)^2 \tag{5}$$

This means that if a vertex moves along a surface, there is no quadric error. However, if a vertex moves perpendicular to a surface, the quadric error increases rapidly. By limiting the quadric error, the amount by which a vertex may move from a surface is limited [5]. A penalty function is used to trade the quality of an element off against its quadric error. Klingner [5] has shown that it is possible to achieve high quality improvement by making small changes to the surface of a mesh.

Although using surface quadrics has been shown to be effective, this method has the disadvantage that the geometry of the domain is being changed. It is always desired that the domain shape be determined by the physics of the problem and not by mesh quality requirements. Therefore, we wish to develop a method whereby surface vertex movement does not change the geometry of the domain, using only information which may be derived from the discretised domain.

3.2 Optimising Mesh Surface Using Boundary Representation

All modern CAD systems use a Boundary Representation or B-Rep solid model to store geometry. If this information is available, then Mesquite can optimise surface meshes whilst restricting surface vertices to their respective surfaces. Mesquite also contains a deforming domain class whereby the initial mesh of the undeformed domain is used to guide the optimisation of deformed mesh, [3]. However, there are many cases of domain deformation whereby this information will not be available such as crack propagation and dam break analysis to give two examples from our research group.

3.3 Generating Surface Geometry from Discretised Domain

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 our hypothesis that for a given shape, the surface area to volume ratio is a constant.

$$\frac{V}{A} = C \tag{6}$$

where V is the domain volume, A is the surface area of the domain and C is a constant.

From this expression the following equation may be derived:

$$\int_T \underrightarrow{N} n dT.\delta x_{i+1} = \int_T \underrightarrow{N}(x_0.n_0)dT - \int_T \underrightarrow{N}(x_i.n_i)dT \tag{7}$$

where the domain T is the surface triangles being integrated over, N is the matrix of element shape functions, δx is the change in nodal positions, x_0 is the nodal coordinates for the initial mesh and x_i for the mesh at the i^{th} iteration and n_0 is the unit outward normal evaluated at the element integration points. The derivation for this equation may be found in Appendix A.

Equation (7) allows us to express the domain geometry in terms of the discretised domain, which enables us to move surface nodes whilst maintaining the domain geometry. Furthermore, this method conserves mesh volume due to the underlying assumption. This is of great importance in some analysis such as in the case of microfluids where minute changes in volume can adversely affect the results of an analysis. This underlying equations for this method are derived from a continuous shape, not a discretised shape, meaning that this equation is valid for the continuous domain and not just the discretised domain. This therefore allows for the movement of surface nodes based on the actual domain geometry and not on the discretised geometry. Furthermore, this equation is valid for all element types and orders. The equation takes the following form and is expressed in this case using triangles as the surface element:

$$C\delta x = g \tag{8}$$

where
$C = \int_T N n dA$
and
$g = \int_T N x n dA$

C is a constraint matrix and g is a solution vector. A Newton based solver is used to optimise the mesh. The system of equations which is being minimised in order to optimise the nodal positions is expressed as

$$Sx = f \tag{9}$$

where S is the Hessian matrix, x is the nodal search directions which are required and f is the gradient vector. Using an expression developed by [10], this system of equation may be modified as follows:

$$S' = C^T C + Q^T S Q \tag{10}$$

$$f' = C^T g + Q^T (f - SRg) \tag{11}$$

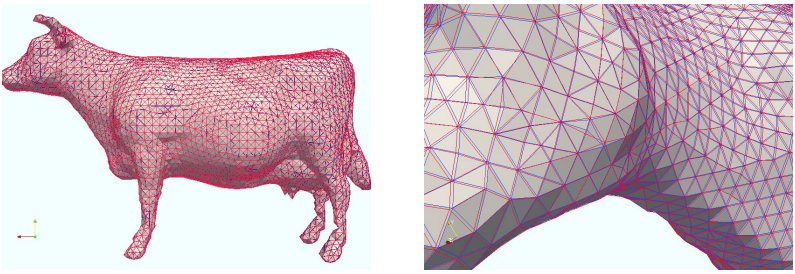
$$R = C^T(CC^T)^{-1} \tag{12}$$

$$Q = I - C^T(CC^T)^{-1}C \tag{13}$$

This modified system of equations,

$$S'x = f' \tag{14}$$

is then solved using an iterative solver to calculate nodal search directions. A line search algorithm is used to find the optimal nodal locations based on element quality and the surface constraints.



(a) 'Cow', Klingner [5] (b) Surface Node Movement

Fig. 5. Effectiveness of Surface Algorithm, red mesh is optimised and blue is original mesh

The mesh in Figure 5 was optimised using this algorithm. The original mesh and optimised mesh are overlain on each other so that it is possible to see the movement of surface nodes. As is often the case with complex mesh, many of the worst elements have all four nodes on the surface. This means that traditional optimisation-based smoothers will not be able to improve it due to the inability to move surface nodes. The results obtained from using this complex mesh demonstrate how effective this is. The movement of surface nodes may be seen in Figure 5b, which is an enlarged view of the neck region of the mesh. Even though the movement of surface nodes is slight, the quality of the worst element has improved by 15% from 0.237 to 0.273. This movement of surface nodes was achieved with a change in volume of 0.00017% and a change in the volume to surface ratio of 0.0014%, which is negligible. The same mesh was optimised using Stellar with its default settings enabled, including topology modification and surface quadrics. Stellar could not successfully improve the mesh. The mesh optimisation routines included in Mesquite could not improve this mesh due to all the nodes of the worst elements being on the mesh surface. The quality improvement achieved using

such a complex mesh demonstrates just how effective the combination of the algorithms presented in this paper are, with the movement of surface nodes enabling the other algorithms to improve the mesh.

This method is also effective when applied to planar surfaces - however as such surfaces may easily be defined mathematically there are much simpler and efficient means of optimising surface nodes, as shown in Section 3.

4 Conclusions and Future Work

We have developed and implemented a very effective mesh optimisation methodology. The combination of a log-barrier objective function, selective-patch based improvement and surface optimisation has enabled us to optimise mesh which previously would not have been possible, such as 'Cow' , in Figure 5. The log-barrier objective function both improves the quality of the mesh obtained and achieves this much quicker than other optimisation based smoothers. The optimisation of surface nodes is completely automated and integrated into Mesquite as are the log-barrier objective function and worst patch selector.

The surface optimisation algorithm works very well for curved surfaces. However, this algorithm breaks down on sharp edges. Therefore, a 2 dimensional version of this algorithm must be implemented in order to enable this method to be applied to mesh with sharp edges. This will enable nodes to move along curved lines without changing the mesh geometry. Routines which automatically detect which nodes lie on curved surfaces and which lie on curved lines must be developed, similar to routines which we have already developed and implemented into Mesquite which automatically classify nodes lying on planar surfaces and straight lines. This is a step towards this project's main goal of automating and simplifying as much as possible the mesh improvement process.

References

1. Shewchuk, J.R.: What Is a Good Linear Finite Element? Interpolation, Conditioning, Anisotropy, and Quality Measures (2002)
2. Donea, J., Huerta, A., Ponthot, J., Rodr, A.: Encyclopedia of Computational Mechanics. John Wiley & Sons, Ltd, Chichester (2004)
3. Knupp, P., Freitag-Diachin, L., Tidwell, B.: Mesh quality improvement toolkit user's guide. Technical report (2012)
4. Tautges, T.J., Kratfcheck, J.A., Smith, B.M., Kim, H.-J.: Mesh Oriented Database Version 4.0 User's Guide (June 2011)
5. Klingner, B.: Tetrahedral mesh Improvement. PhD thesis, University of California at Berkeley (November 2008)
6. Freitag, L.A., Ollivier-Gooch, C.: A Comparison of Tetrahedral Mesh Improvement Techniques. In: 5th International Meshing Roundtable (1996)

7. Chen, Z., Tristano, J.R., Kwok, W.: Combined Laplacian and Optimisation-Based Smoothing for Quadratic Mixed Surface Meshes. In: 12th International Meshing Roundtable, pp. 360–370 (2003)
8. Freitag, L.a., Jones, M., Plassmann, P.: An Efficient Parallel Algorithm for Mesh Smoothing. In: Proceedings of the Fourth International Meshing Roundtable (1995)
9. Sastry, S.P., Shontz, S.M., Vavasis, S.A.: A Log-Barrier Method for Mesh Quality Improvement
10. Ainsworth, M.: Essential boundary conditions and multi-point constraints in finite element analysis. Computer Methods in Applied Mechanics and Engineering 190(48), 6323–6339 (2001)

A Derivation of Surface Constraint Equation

Our hypothesis is that for a given shape, the volume to surface area ratio is a constant. Therefore:

$$\frac{V}{A} = C \tag{15}$$

where V is the domain volume, A is the surface area of the domain and C is a constant.

Therefore

$\int_V dV = C \int_A dA$

$\frac{1}{3} \int_V div(x) dV = C \int_A dA$

Using Green's theorem this may be rewritten as:

$\int_A n.x dA = C_3 \int_A dA$

where n is unit outward pointing normal, x is the domain surface and C_3 is C times 3.

localise by dropping integral

$n.x dA = C_3 dA$

$n.x = C_3$

$\forall x \epsilon$ Surface

Multiple both sides by the element shape functions and integrate over the surface triangle and sum for all surface triangles.

$\int_T N n dA.x = \int_T N C_3 dA$

Expand the right hand side using a Taylor Series and truncate after the linear term, gives the linearised version

$\int_T \underrightarrow{N} n dA.\delta x_{i+1} = \int_T \underrightarrow{N}(X_0.n_0)dA - \int_T \underrightarrow{N}(X_i.n_i)dA$

where *0* is the initial mesh and i is the mesh at the i^{th} iteration.