ALL-HEXAHEDRAL MESH SMOOTHING WITH A NODE BASED MEASURE OF QUALITY.

N. A. CALVO and S. R. IDELSOHN

Centro Internacional de Métodos Computacionales en Ingeniería - CIMEC, Güemes 3450, (3000) Santa Fe, Argentina

SUMMARY

This research work deals with the analysis and test of a normalized-Jacobian metric used as a measure of the quality of all-hexahedral meshes. Instead of element qualities, a measure of node quality was chosen. The chosen metric is a bound for deviation from orthogonality of faces and dihedral angles. We outline the main steps and algorithms of a program that is successful in improving the quality of initially invalid meshes to acceptable levels. For node movements, the program relies on a combination of gradient driven and simulated annealing techniques. Some examples of the results and speed are also shown.

KEY WORDS: unstructured mesh; hexahedral mesh; smoothing; Jacobian metric; all-hexahedral mesh

1. INTRODUCTION

The optimization of a given mesh for Finite Element Analysis implies that the elements must satisfy certain quality conditions. Many methods are currently in use[1,2,3], and there is also an excellent compendium of matrix norms useful for hexahedra, cited in reference 4; but it is still difficult to give a concrete meaning to the word "quality" for a mesh.

Even tough the actual algorithm we use combines orthogonality with aspect ratio, we will only concentrate here on the angular aspect of quality. This is because traditional methods give us meshes with negative-angled elements. Therefore, within the scope of this paper, any hexahedral element must have its edges and faces as orthogonal as possible.

We will be using the term "Jacobian" for the matrix involved in the mapping between any tetrahedron over the canonical one. A tetrahedral element can be mapped with a single Jacobian matrix whereas hexahedral elements have a different Jacobian at each vertex.

Mesh improvement involves two main approaches, which are sometimes used together: smoothing and topological operations. It is a common practice to use the term "smoothing" when only changes in node positions are involved. When there are some changes in the connectivity among the nodes, changing also the quantities of nodes or elements, these operations are referred to as topological changes.

Whereas tetrahedral meshes can readily undertake local topological changes, they are still unimplemented for all-hexahedral meshes. This is because of the lack of algorithms to handle and operate with dual surfaces or sheets of elements[5,6,7]. We are currently smoothing, i.e. repositioning nodes without changing the topology of the mesh.

Until recently, the only known smoothing technique applied to hexahedra was the Laplacian smoothing[8]: a movement of each node towards the centroid of its neighboring nodes. The drawback of the Laplacian method is the compression and even inversion (negative Jacobian) of elements near concave areas of the boundary. Even the newest methods[3,4] can't successfully untangle a mesh with inverted hexahedral elements.

2. THE GEOMETRY

2.1. Metric

We are using a technique known as 'optimization-based smoothing'[1,9] where each selected node is moved in order to maximize a quality-related function called quality metric.

The algorithm is intended to move nodes; therefore, instead of handling element qualities, we have defined a node quality metric. The subtle difference between node and element quality has proven useful at the implementation stage. It is easier to evaluate the impact of each movement on the quality of neighboring nodes using a node based quality measure.

In order to improve meshes with some concave elements (with a negative Jacobian), we need a metric pushing the mesh towards (positive) orthogonality. After we tested dihedral angles and edge angles as objective functions, we realized that the normalized Jacobian was a simple bound for both of them.

As elements define a trihedron on each node, we will deal mostly with trihedra rather than elements.

Figure 1 shows a mesh element. The segments PA, PB and PC define a trihedron with P as its vertex.

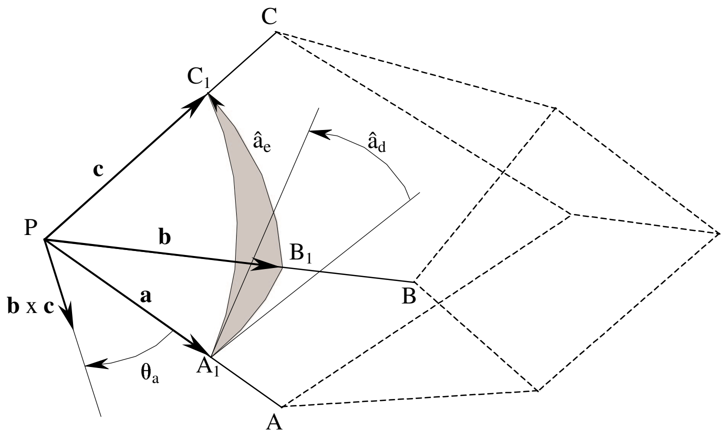

Figure 1: A trihedron in a hexahedral mesh element.

The intersection of S^2_P (the unit sphere centered at P) and the element is the unit spherical triangle shown grayed. S^2_P intersects the edges at A_1, B_1 and C_1, with **a**, **b** and **c** the respective unit vectors so defined.

The angle between the edges PB and PC was labeled $\hat{a}_e$, $\hat{a}_d$ is the dihedral angle between the triangles BPA and CPA; and θ_a is the angle between the edge PA and the normal **b** x **c** to the triangle BPC. Letters a, b and c can be cyclically used for all the angles between faces and edges.

In order to avoid any problem with the signs, we choose angles such as $\hat{a}_e$ and $\theta_a \in [0, 180°)$ and dihedrals such as $\hat{a}_d \in (-180°, 180°]$. Thus, when thetrihedron becomes inverted, all the θ's are over 90° and all the dihedrals are negative.

The mapping of these unit vectors in the canonical tetrahedron is made with a Jacobian matrix whose determinant is j = (**a b c**). It can take values between –1 and 1, and is the normalized version of the Jacobian determinant J=(**PA PB PC**) for the element edges.

The normalized Jacobian is an orthogonality measure for each trihedron at P. This measure is a lower bound on the sine of the angle between edges:

$$j = (\mathbf{a}\ \mathbf{b}\ \mathbf{c}) = \mathbf{a} \bullet (\mathbf{b} \times \mathbf{c}) = \cos(\theta_a)\ \sin(\hat{a}_e) \leq \sin(\hat{a}_e). \tag{1}$$

The dihedral angle $\hat{a}_d$ is the angle between $\mathbf{a} \times \mathbf{b}$ and $\mathbf{a} \times \mathbf{c}$, as seen from A to P, then:

$$\sin(\hat{a}_d) = \frac{(\mathbf{a} \times \mathbf{b}) \times (\mathbf{a} \times \mathbf{c})}{\|\mathbf{a} \times \mathbf{b}\|\ \|\mathbf{a} \times \mathbf{c}\|} \bullet \mathbf{a} = \frac{(\mathbf{a}\ \mathbf{b}\ \mathbf{c})\ \|\mathbf{a}\|}{\|\mathbf{a} \times \mathbf{b}\|\ \|\mathbf{a} \times \mathbf{c}\|} = \frac{(\mathbf{a}\ \mathbf{b}\ \mathbf{c})}{\sin(\hat{c}_e)\sin(\hat{b}_e)} \geq (\mathbf{a}\ \mathbf{b}\ \mathbf{c}) = j\ ; \tag{2}$$

Where double bars mean the module of the enclosed vector was taken. This equation shows the given metric as a lower bound for the sine of the dihedral angles.

As the previous equations are equally applied to the angles on the other edges: *the angles between faces and between edges of a trihedron are closer to 90º than the arcsine of j.*

We now have a metric to quantify the quality: *The quality of a node is the least valued j from the set of trihedra sharing that node. The quality of the mesh is the worst node quality.*

Frequently some sort of mean-value is used instead of the worst one, relations between inner an outer spheres or something alike. These values may be suitable for triangles or tetrahedrons, but are rather dangerous for quads and hexes. They are prone to give inverted elements, and a single one turns the whole mesh useless.

2.2. Internal Nodes

In order to improve the quality of a node, we move it while keeping the adjacent ones fixed. With the exception of a few special points, *j* is a differentiable function of node position. The element to which the worst *j* belongs, changes as the node moves.

In the two-dimensional (plane) case, we can easily show that problem. In Figure 2, we have isolated one of the quads sharing some node P, and showed the non-adjacent (and irrelevant) edges in dotted lines.

Fixing the nodes A and B while P moves in the plane, we can plot *j* as a function of the position of the moving node. In this case, *j* is the component of $\mathbf{a} \times \mathbf{b}$ in a direction that is perpendicular to the plane.

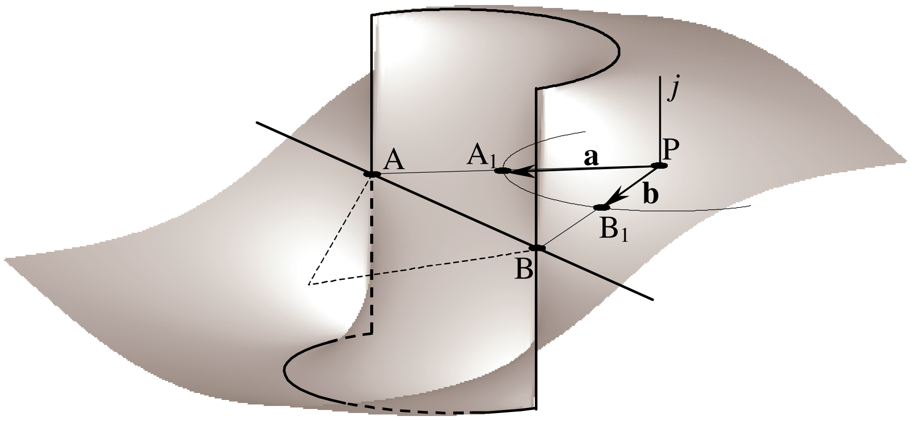

Figure 2: Dependence of two-dimensional metric on the node position, for one of the elements attached to the moving node

The line joining the fixed nodes is the border between positive and negative half-planes. The metric value is zero on that line, and one or minus one at the circle with the fixed points diametrically opposite (the diameter of the circle subtend a right angle).

For each element attached to the (moving) node, we have one of these surfaces. From that set of surfaces we make a single one choosing the lowest value at each point. This is the surface we can see viewing the set of intersecting surfaces from below. We must find the maximum value of that function.

In our three-dimensional case, we have three fixed nodes (such as A, B and C in Fig.1) for each element attached to the moving node P. The plot of j, as a function of the three node coordinates, is a 4D surface for each attached element. The planes through each triplet of fixed nodes, are the borders between positive and negative half-spaces. For each attached trihedron, any pair of fixed points (say A and B) defines a sphere (in which A and B are diametrically opposite). If the node were on the surface of that sphere, the edges (PA and PB) will form a right angle. In order the three edge angles are right (j equals one or minus-one), the node must be at the intersection between the three spheres so defined (by AB, BC and CA). This intersection exists as long as the fixed nodes are forming an acute triangle (ABC).

When the node has a negative metric, the gradient of the function may move it in the wrong direction. In Figure 2, on the left, the surface is negative and outside the minus-one semicircle. Here, the gradient moves the point away from the line; causing j to grow as it asymptotically approaches to zero. Such movement increases the metric value but in the wrong direction. In the positive half-plane, we don't have such a problem.

Thus, when the metric is negative we use J (the triple product of the edge vectors) without dividing it by the edge lengths. It doesn't require a new calculation; when it is negative, it is not divided by the edge lengths.

The corresponding surface in 2D, is a half-plane bounded by the dividing line and tangent to the negative half circle of fig. 2 (all the triangles/tetrahedrons with the same base and the same height have the same area/volume).

This combination of metric values is one of the key approaches to successfully untangle meshes having nodes with negative Jacobian.

2.2.1. Influence of the Set of Neighboring Nodes

The position of a node affects the quality of its neighboring nodes.

The kernel of a polygon is the intersection of all its internal angles[10]. It is a convex polygon included in the former one. So defined, the kernel is the set of points from which all the nodes of the polygon can be viewed without interference. If the starting polygon is convex, then the kernel coincides with it.

Figure 3 a) displays a node P from a two-dimensional planar mesh and the quadrilatera sharing that node. The adjacent nodes are labeled A_i, and the diagonally opposite ones are labeled D_i. Shown in b) is the enveloping polygon of the adjacent nodes. *The quality of* P *is positive only if the node is inside the kernel of that polygon.* If the point is moved out of the kernel, at least one of the attached elements will be inverted.

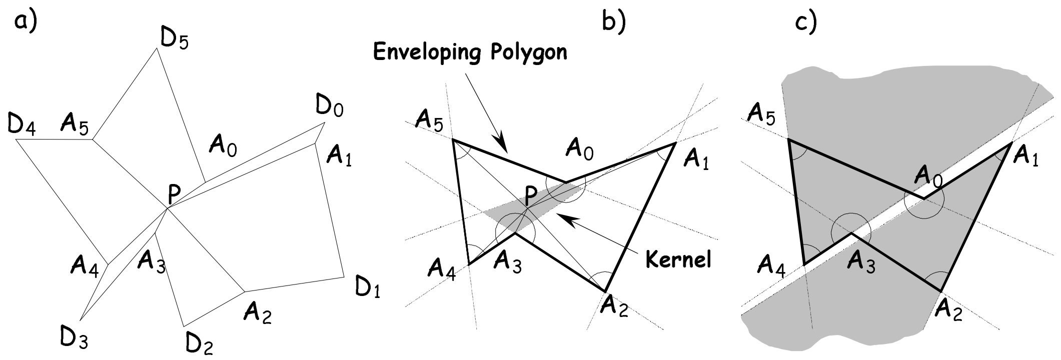

Figure 3: Envelope and Kernel of the nodes adjacent to a node P

In a polygon that has more than one concavity, there may be no kernel at all and no place with positive metric. In Figure 3 c), we moved the node A_0 to void the intersection between the internal angles at A_4 and A_1, in this case there is neither kernel nor a suitable position for the inner node.

In the three dimensional case we have an enveloping polyhedron, trihedral solid angles and a convex kernel polyhedron yielding positive metric to all the trihedra attached to the node.

In order to avoid futile attempts to improve a node whose quality can't be bettered any more, we need a value for the maximum quality attainable by each node. The positions of the neighboring nodes severely restrict that value, but these positions will change in successive steps. The valence of the node (the number of attached edges), in turn, limits the quality and this limit will not change.

In 2D the minimum angle at a node with *n* elements attached can't be over 360º/*n.* In 3D the general function relating the upper bound on *j* with the number of edges is the same problem as the optimal placing of points in a spherical surface. It is an unsolved combinatorial geometry problem [11,12], so we currently don't provide the program with such a value.

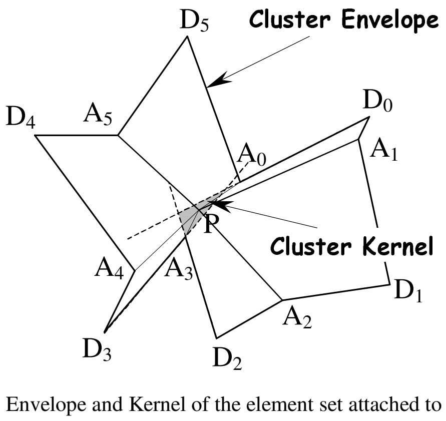

Figure 4: Envelope and Kernel of the element set attached to a node P

If we analyze Figure 3 b) and Figure 4 we can easily see that, by increasing the quality of a node, we may negatively affect that of its neighbors. Even in the kernel of the adjacent nodes, where the node P has positive-metric, some of the attached nodes can take negative metric values.

The cluster-enveloping polygon, formed with the attached elements, has a smaller kernel (with a higher probability of being void) than the adjacent nodes envelope. In the cluster kernel, the node has positive metric and does not force a negative metric on its neighbor nodes. It is evident that the centroid of the cluster kernel is a better position for the node than the commonly used (Laplacian) centroid of the envelope. As the kernel is difficult to obtain in 2D, obviously our 3D case is even harder. Any algorithm doing that will be welcomed.

2.3. Boundary Nodes

The boundary of the hexahedral mesh is a closed quadrilateral mesh with fixed nodes. We will show how the quality of the hexahedral mesh is affected by the quality of the prescribed boundary.

Our hexahedral mesh generator makes one element for each face of the outer mesh. As was explained in previous works[5,6], a layer of elements adjacent to the boundary composed by one hexahedron for each quad, makes it possible to correct many potential problems with the outermost elements.

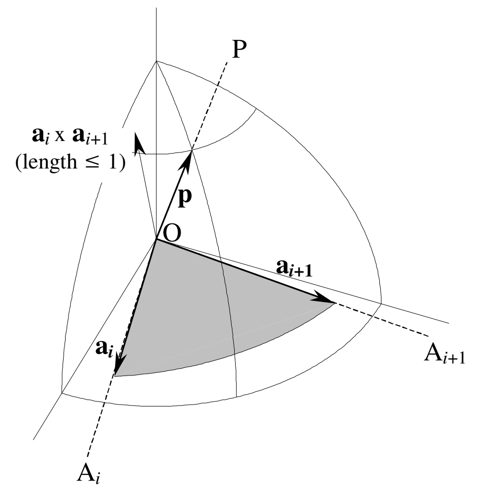

Figure 5: Fixed boundary node

Figure 5 above displays an external node O with one of the *n* trihedra sharing the segment OP. In addition, we can see $S^2{}_O$ (unit sphere at O), the grayed sector left by $S^2{}_O$ at the i^{th} external quad, and the corresponding unit vectors. Translating the origin of coordinates to O, we have (as P moves) the metric of the i^{th} trihedron:

$$j_i = (\mathbf{a}_i\ \mathbf{a}_{i+1}\ \mathbf{p}) = (\mathbf{a}_i \times \mathbf{a}_{i+1}) \bullet \mathbf{p} \Rightarrow \nabla j_i = \mathbf{a}_i \times \mathbf{a}_{i+1}, \text{ with the indices cycling modulus } n. \qquad (3)$$

The maximum j_i is for **p** perpendicular to the i^{th} face (actually, to the triangle A_i O A_{i+1}).

The direction **p** is *a-priori* unknown, so the worst j_i is also unknown but we can say that it is worse (lesser) than the worst $\mathbf{a}_i$ x $\mathbf{a}_{i+1}$. Therefore, the quality of the mesh is conditioned from the beginning by the worst (two-dimensional) j_i of the quads, i.e. the quality of the exterior mesh.

This simple bound is a poor approximation, as the scalar product will lower the actual quality in acute polyhedral angles.

The calculation of the limit on the orthogonality reachable by any border node has the same difficulty level as the calculation of the optimal position of any node and. So we don't perform it, however it is a source of time consumption when the program makes futile attempts to improve beyond imposed restrictions.

3. PROGRAMMING

Our goal is to move each node to a better position. To compare positions, we use the quality of the node as well as the worst quality from the set made up of the node and its neighbors.

So, in our algorithm, *a new position for the node P is considered to be better if the worst quality of the set* {P, {A*i*}} *improves or if it stays the same and the quality of the node improves*.

Every previous attempt we made by using only analytical ways invariantly ended up with some locking positions, often with negative metric.

In order to select the nodes to be moved, a list in reverse quality order is made. The first nodes on the list are selected and tested one at a time. This sub-list is repeatedly traversed. The quantity of tested nodes is a function of the overall quality of the mesh, starting with a few nodes and expanding to the entire list as the quality rises, or when there is no significant improvement.

To change a node position, the program has three different approaches:

1) Gradient driven:

As was explained before, a node movement in a direction improving the worst trihedron quality (∇j) may lower the quality of the neighbor nodes. Thus, the program makes a minimum movement in that direction; if it is a better position (in the sense already defined), the maximum successful step in that direction is found.

The obvious choice for the reference step is:

$$\|\Delta \mathrm{P}\| = \frac{1-j}{\|\nabla j\|} \Rightarrow \Delta \mathrm{P} = \frac{1-j}{\|\nabla j\|^2}\nabla j\,. \tag{4}$$

This step carries j to 1 (orthogonal trihedron), but it has proven less efficient than the mean length of the edges attached to the node. Labeling a_i the ith edge length $\|\mathrm{A_i\,P}\|$, and considering there are n of such attached edges, we choose the reference movement length:

$$\|\Delta \mathrm{P}\| = \langle a_i \rangle = \frac{\Sigma \|\mathrm{A}_i\,\mathrm{P}\|}{n}\,. \tag{5}$$

The first (and minimum) attempt is 1/128 of this. If it is successful, we test the whole reference step. If it doesn't succeed, we halve it recursively until a movement improving the node quality is found.

If the minimum movement doesn't succeed, the control passes to the next algorithm.

2) Simulated annealing:

The node is randomly moved and tested. The movement steps randomly takes different directions and their magnitudes are random fractions of the mean edge length (previously calculated):

$$\|\Delta \mathrm{P}\| = f\,\langle a_i \rangle, \text{ where } 0 < f \leq 1 \tag{6}$$

The number of attempts is a linear function of quality; between 0 for quality 1 and 50 attempts when quality falls to 0 or negative.

3) Forced movement: If all the previous tricks have failed and:

a) The quality of the node is 0 or negative,

b) This is the sixth time with this node and is still unchanged,

c) The node has at least one positive-metric neighbor node,

Then, the node and its neighbors are forced to move randomly as defined in the previous algorithm.

This movement is meant to prevent severe locking of a node. The causes may vary but it actually happens and this solution works out well.

The third condition lies in the fact that almost every example we tested has a near-smooth layer of many negative-metric nodes (mostly at the domain frontier). It is useless to work inside this layer so we chose to move only those nodes at the interface with those ones with a good position.

Finally, the program halts if the required quality is reached, or some prescribed time has passed or the list of nodes is complete and the quality doesn't improve.

Even though all the factors and functions chosen were experimentally tested, we don't expect them to be the best (fastest) values for every situation. We can only say that these values gave us positive metric meshes in all our examples and benchmarks. Many of them may seem superfluous, inefficient or extremely conservative, but they are necessary in order to guarantee the result. Regardless of whether it takes a minute or an hour, the speed must not interfere with the robustness.

4. EXAMPLES:

The examples were tested in an ALPHA station 500, 333 MHz with 320 MB of RAM and Digital UNIX V4.0B operating system.

We have tested mostly benchmark, few-element-meshes without any engineering sense, however we have selected three examples of concave meshes to be shown here.

In order to test the robustness of the process, all the runs were made with the same set of parameters.

4.1. Curved Beam

The first example is a test case of an acutely curved beam, forcing bad results from Laplacian-only smoothing. The resulting hexahedral mesh has 1586 nodes and 1310 elements.

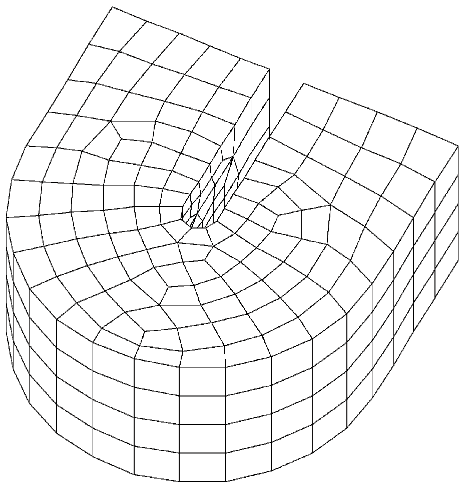
Figure 6: Mesh with an acute concavity

The running test started with 150 negative-quality nodes. They reached the 0-quality status in a minute and the overall quality rose to an acceptable level > 0.1 (about 6° or 174°) in less than two minutes.

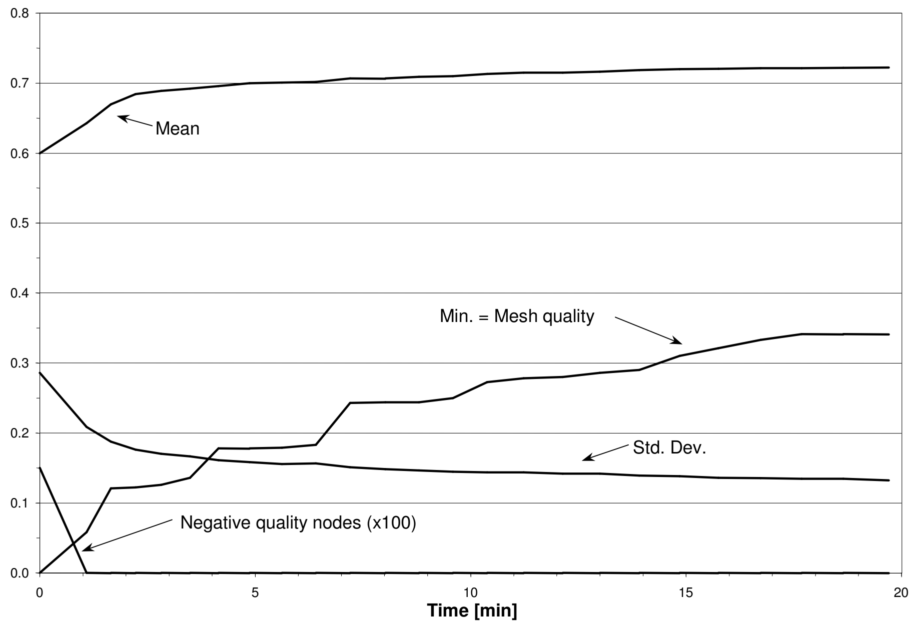

Figure 7: Run plot of quality for the curved beam

In the plots, mean, minimum and standard deviations are over the set of node qualities, being 1 for a perfectly orthogonal trihedron, 0 when squashed on a plane and negative when inverted.

4.2. Gear tooth

The second and more complex example shown in Figure 8 is a test problem of a gear tooth from a contact problem. The planar mesh was made by using a planar paving[13] mesh-generator and then extended and refined in the 3rd dimension with the help of a CAD program. The all-hexahedral mesh has 9286 nodes and 8210 elements.

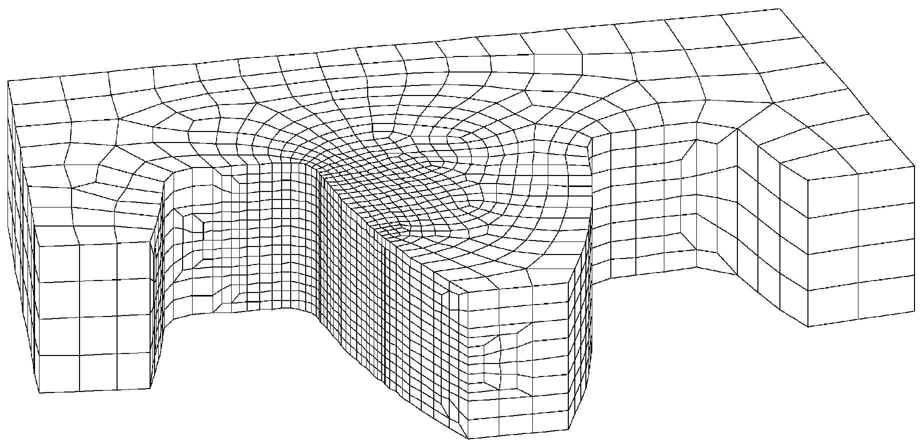

Figure 8: Mesh for a gear tooth

The initial situation with 404 negative nodes was corrected in about 10 minutes and the mesh quality reached an acceptable level in 15 min.

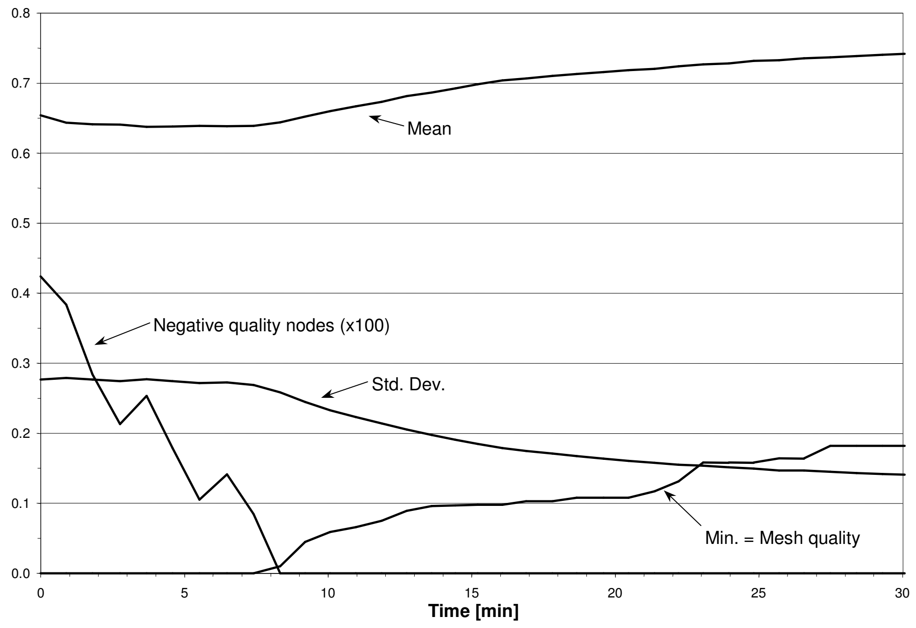

Figure 9: Run plot of quality for the tooth.

4.3. Spherical Bearing

The external mesh for the spherical bearing shown in Figure 10 was entirely made with a standard PC CAD program. The corresponding all-hexahedral mesh has 68689 nodes and 60112 elements.

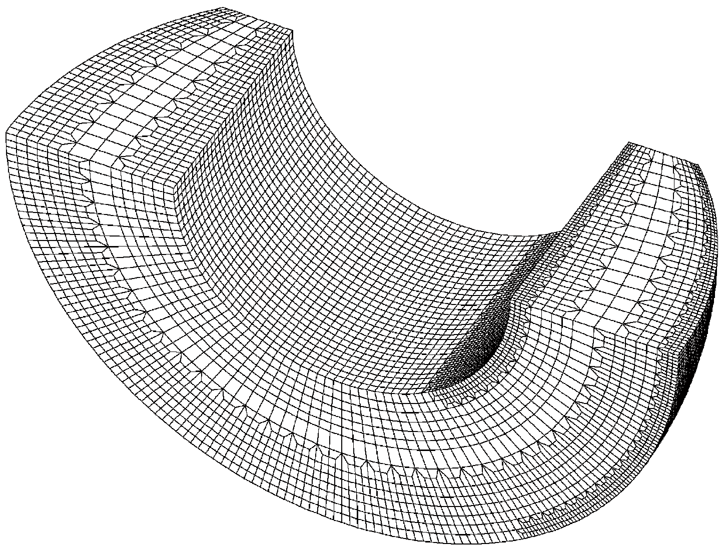

Figure 10: CAD Mesh for a Spherical Bearing

Starting with the Laplacian smoothed mesh, the initially negative 56 nodes were corrected in less than a minute, and it took a total of 11 min to improve the overall quality.

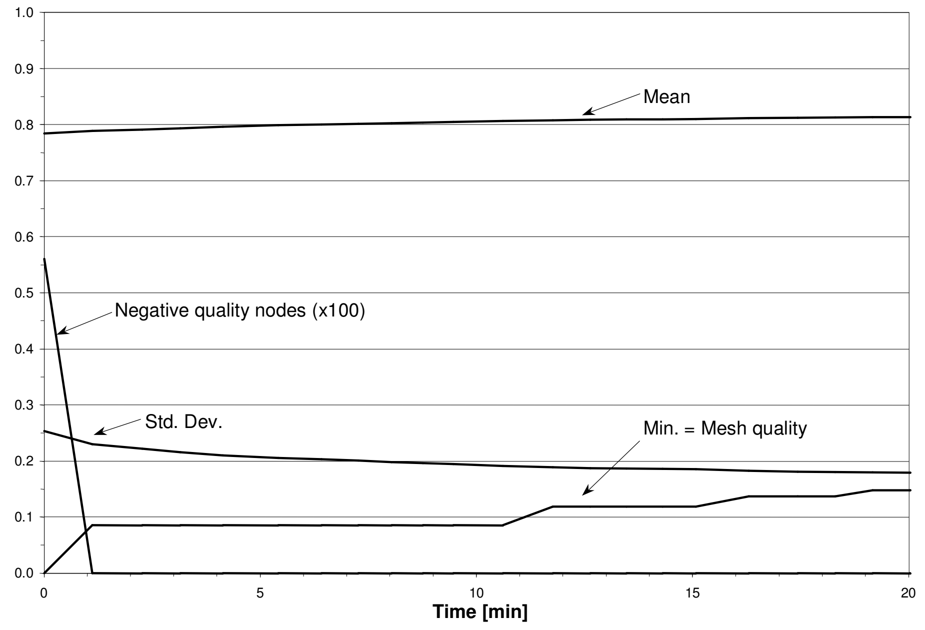

Figure 11: Run plot of the spherical bearing, starting with the Laplacian-smoothed mesh.

5. CONCLUSIONS:

The problem of finding the best shape for the elements/nodes of an all hexahedral mesh has proven intractable by some of the standard means used with tetrahedrons.

Although we still have lack of consensus about the best quality measure for hexahedral elements, we made some progress towards the improvement of angular quality.

Surely this metric alone is not the solution. It must be combined with suitable values for the aspect ratio and the planarity of the faces in order to boost the overall quality of the mesh. We actually use a product made up of the metric already presented and a node-based measure of aspect ratio related to the logarithm of the quotient between minimum and maximum lengths from the set of attached edges.

The method has proven to be robust and every test has given a positive result, so it can be easily extended to any quality measuring value. The very philosophy of the object-oriented programming allows us to replace the metric used by any other.

We are now in a position to make good quality meshes for the type of problems that can be successfully addressed by our all-hex generator.

REFERENCES

1. S. A. Canann, J. R. Tristano, and M. L. Staten, 'An Approach to Combined Laplacian and Optimization-Based Smoothing for Triangular, Quadrilateral and Quad-dominant Meshes', *7th International Meshing Roundtable,* Michigan, 1998.

2. S. J. Owen, *A Survey of Unstructured Mesh Generation Technology*, chapter 5: 'Mesh Post-Processing', http://www.andrew.cmu.edu/user/sowen/survey/postsurv.html

3 L. A. reitag and P. M. Knupp. 'Tetrahedral element Shape Optimization via the Jacobian Determinant and Condition Nmber', *8th International Meshing Roundtable,* California 1999.

4 P. Knupp. 'Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities', Parts I and II. Technical Report SAND 99-0709J, Sandia National Laboratories, 1999

5. N. A. Calvo and S. R. Idelsohn, 'Generador automático de mallas de hexaedros: presentación del método y avances en la implementación', *Mecánica Computaciona*l Vol XVI, Asociación Argentina de Mecánica Computacional (1996).

6. N. A. Calvo and S. R. Idelsohn, 'All-hexahedral element meshing by generating the dual mesh', *Proceedings of the IV World Congress on Computational Mechanics*, Buenos Aires 1998.

7 T. J. Tautges, T. D. Blacker, and S. A. Mitchell, 'The whisker weaving algorithm: a connectivity-based method for constructing all-hexahedral finite element meshes'*, Int. J. Num. Meth. Engng*. **39**, 3327-3349 (1996).

8 P. Knupp, 'Applications of mesh smoothing: copy morph, and sweep on unstructured quadrilateral meshes', *Int. J. Num. Meth. Engng*. **45**, 37-45 (1999).

9. L. Freitag, M. Jones, and P. Plassmann, 'An Efficient Parallel Algorithm for Mesh Smoothing'*, Proceedings of the 4th International Meshing Roundtable*, 47-58, (1995).

10. F. Preparata and M. Shamos, *Computational Geometry*. Springer-Verlag, New York 1985.

11 G. McCaughan, 'Circle Packings', http://www.pmms.cam.ac.uk/~gjm11/cpacking/info.html.

12 P. Bourke, 'Distributing Points on a Sphere' http://www.mhri.edu.au/~pdb/geometry/spherepoints/.

13 T. D. Blacker,. and M. B. Stephenson, 'Paving: a new approach to automated quadrilateral mesh generation', *Int. J. Num. Meth. Engng*. **32**, 811-847 (1991).