MECOM 99
Mendoza
6-10 de septiembre de 1999

ALL-HEXAHEDRAL MESH SMOOTHING WITH A NORMALIZED JACOBIAN METRIC COMBINING GRADIENT DRIVEN AND SIMULATED ANNEALING.

Nestor. A. Calvo y Sergio. R. Idelsohn
Centro Internacional de Métodos Computacionales en Ingeniería – CIMEC
Güemes 3450, (3000) Santa Fe, Argentina

RESUMEN

El presente trabajo trata sobre el análisis y prueba de la métrica del jacobiano normalizado, utilizada como medida de la calidad de mallas de hexaedros. Adoptamos una medida para la calidad de nodos en lugar de la calidad de elementos. La métrica del jacobiano normalizado es una cota de la no - ortogonalidad de las caras y los ángulos diedros. Delineamos los pasos y algoritmos fundamentales del programa que logró elevar hasta límites aceptables la calidad de mallas inicialmente inválidas. El programa se basa en una combinación de movimientos de nodos guiados por el gradiente y movimientos aleatorios utilizando la técnica conocida como "simulated annealing". Se muestran algunos ejemplos de resultados y tiempos de corrida.

ABSTRACT

The present 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 normalized-Jacobian metric is a bound for deviation from ortogonality of faces and dihedral angles. We outline the main steps and algorithms of a program that is successful in rising the quality of initially invalid meshes to acceptable levels. For node movements, the program relies on a combination of gradient driven and simulated annealing. Some example of the results and speed are also shown.

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], but it is still difficult to give a concrete meaning to the word "quality" for a mesh.

Mesh improvement involves two main approaches, sometimes used together, smoothing and topological operations. We are currently smoothing, i.e. repositioning nodes without changing the topology of the mesh. Connectivity changes in all-hexahedral meshes are still unimplemented because of the lack of algorithms to handle and operate between dual surfaces or sheets of elements[3,4,5].

Until the present development we relied on relaxed Laplacian smoothing, i.e. the repositioning of each node towards the centroid of its neighbor nodes. The drawback of the Laplacian method is the compression and even inversion (negative Jacobian) of elements near concave areas of the boundary.

THE GEOMETRY

Metric

We are using a technique known as optimization-based smoothing[1,6], where each selected node is moved in order to maximize (or minimize) a quality-related function called quality (or distortion) metric.

The algorithm is intended to move nodes; thus, 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, to evaluate the impact of each movement on the cluster of neighboring nodes.

In order to improve meshes with inverted elements, we chose a metric pushing the mesh towards ortogonality. We tested using dihedral angles as well as edge angles, and then we found a simple bound for both of them.

Figure 1 shows a node **P** with one of the elements attached to it. The segments **PA**, **PB** and **PC** are the three adjacent edges.

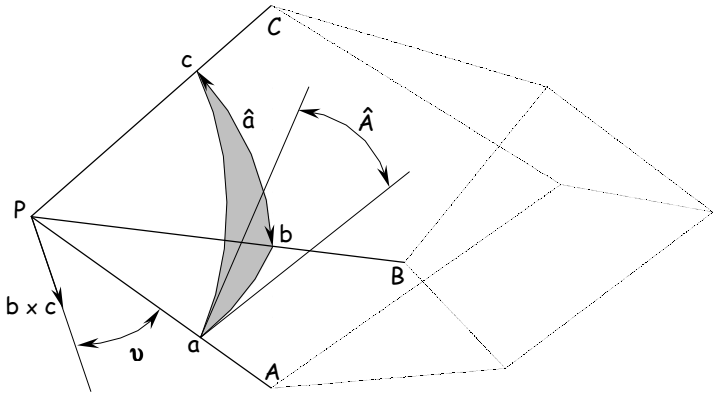

Figure 1: Elements of a trihedron belonging to a node

The intersection of the unitary sphere centered at **P** ($\mathbf{S}^2_\mathbf{p}$) and the element is the included unit spherical triangle shown in gray. The unitary vectors pointing to the direction of the adjacent nodes were labeled **a**, **b** and **c**.

As each element defines a trihedron on each of its nodes, we will deal mostly with trihedrons rather than with elements.

The metric used to quantify the quality of each trihedron at **P** is (**a b c**), the triple product between the directions of the attached segments.

This metric gives a lower bound on the sine of the angle between edges:

$$(\mathbf{a}\ \mathbf{b}\ \mathbf{c}) = \mathbf{a} \bullet (\mathbf{b} \times \mathbf{c}) = \cos(\upsilon)\ \sin(\hat{\mathbf{a}})[\ \sin(\hat{\mathbf{a}}). \tag{1}$$

The dihedral angle $\hat{\mathbf{A}}$ (between faces APB and APC) is the angle between **a** x **c** and **a** x **b**, so:

$$\sin(\hat{\mathbf{A}}) = \frac{\left\|(\mathbf{a}\times\mathbf{b})\times(\mathbf{a}\times\mathbf{c})\right\|}{\left\|\mathbf{a}\times\mathbf{b}\right\|\left\|\mathbf{a}\times\mathbf{c}\right\|} = \frac{(\mathbf{a}\,\mathbf{b}\,\mathbf{c})\left\|\mathbf{a}\right\|}{\left\|\mathbf{a}\times\mathbf{b}\right\|\left\|\mathbf{a}\times\mathbf{c}\right\|} = \frac{(\mathbf{a}\,\mathbf{b}\,\mathbf{c})}{\sin(\hat{\mathbf{c}})\sin(\hat{\mathbf{b}})} \geq (\mathbf{a}\,\mathbf{b}\,\mathbf{c})\ ; \tag{2}$$

thus, the given metric is also a lower bound for the sine of the dihedral angles.

In brief: **the angles between faces and between edges of a trihedron are closer to 90º than the arcsine of the metric value**.

The mapping of these three unitary vectors on the canonical tetrahedron needs a Jacobian matrix whose determinant is exactly that triple product. So we will label this metric as **j** = (**a b c**) in the three dimensional case, and ^{2}**j** = o**a** x **b**o as its two-dimensional version.

We now have a metric to quantify the quality: **The quality of a node is the worst j from the elements sharing that node, and 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 angles, and a single negative angle turns the whole mesh useless.

Internal Nodes

In order to rise the quality of a node **P**; we move that node while keeping fixed the adjacent ones. However, except from a few points, **j** is a differentiable function of **P**, the element to which the worst **j** belongs, changes as the position of the node changes, making this a local maximum problem.

In the two-dimensional case, we can easily show the problem: In we fixed two adjacent nodes and plotted ^{2}**j** against the node position.

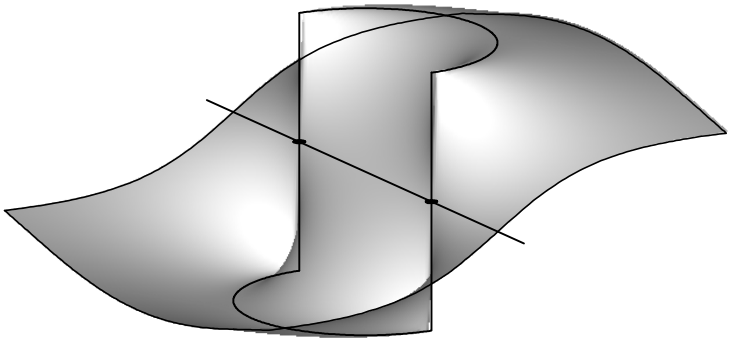

Figure 2: Dependence of two-dimensional metric on the node position

The line joining the fixed nodes divides the positive half-plane from the negative one. The metric value is zero at 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. We must find the maximum of the lower bound, which is the surface we can see viewing from below the set of intersecting surfaces.

In our three-dimensional case, the surfaces are difficult to see. We can say that the plane through each triplet of nodes (belonging to each attached element) divides the positive from the negative half-spaces. The metric takes a value of one or minus-one in the two points of intersection of the three spheres that have each pair of nodes diagonally opposite. The existence condition for this intersection is that the triangle formed with the adjacent nodes should have all its angles measuring less than 90º.

The **kernel** of a polygon is the intersection of all its internal angles[7], it is a convex polygon (without internal angles greater than 180º) included into the former one. So defined, the kernel is the set of points from which all the nodes of the polygon can be viewed. If the starting polygon is convex, then the kernel coincides with it.

Figure 3a) displays a node **P** from a two-dimensional mesh and the quadrilateral elements sharing that node. The adjacent nodes are labeled $\mathbf{A_i}$, and the diagonally opposite $\mathbf{D_i}$. 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.

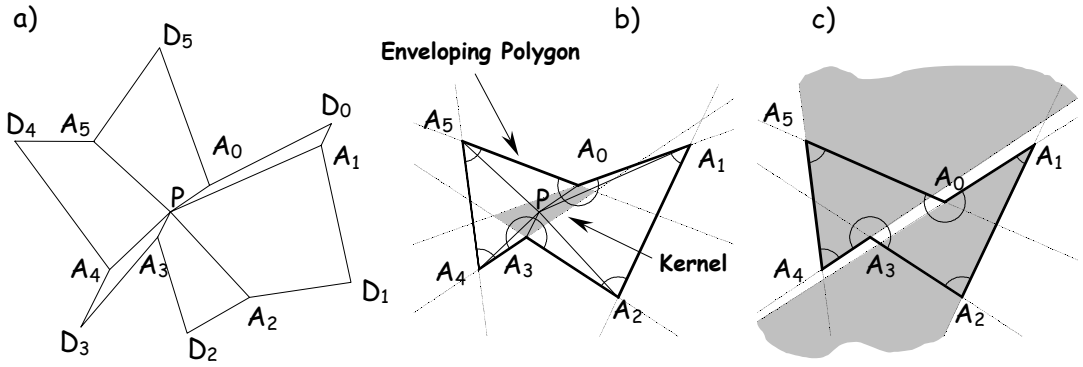

Figure 3: Envelope and Kernel of the adjacent nodes

In case the polygon has more than one concavity, there may be no kernel and no place with positive metric. In c), we moved the node $\mathbf{A_0}$ to void the intersection between the internal angles at $\mathbf{A_4}$ and $\mathbf{A_1}$, then 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.

The positions of the attached nodes severely restrict the maximum quality available for that node, but this position will change in successive steps. The valence of the node (the number of attached edges), in turn, imposes a limit in the quality that will not vary. The general function relating the upper bound on **j** with the number of edges has to do with the problem of optimal packing of circles in a sphere, it is an unsolved problem of combinatorial geometry[8,9].

Boundary Nodes

The boundary of the hexahedral mesh is a closed quadrilateral mesh with fixed nodes. We will show how the quality of the given external boundary affects the interior mesh.

Our hexahedral mesh generator makes one element for each face of the outer mesh. We have explained in previous works[3,4] that a layer of elements adjacent to the boundary (like an offset) makes it possible to correct many potential problems with the outermost hexahedral elements.

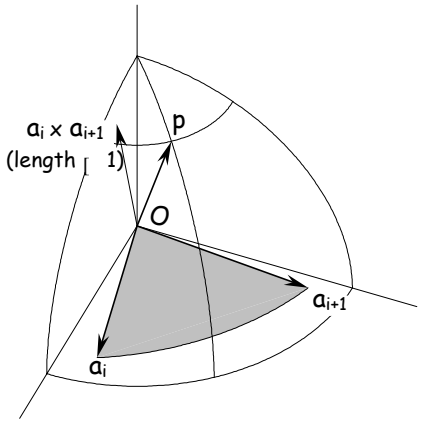

Figure 4: Fixed boundary nodes

The boundary mesh contains **n** quadrilateral elements sharing the node **O** (Figure 4); the corresponding **n** hexahedrons share the **OP** segment. Intersecting all this with $\mathbf{S}^2\mathbf{o}$, we have for the metric of the $\mathbf{i}^{th}$ trihedron:

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

Then, the maximum $\mathbf{j}_i$ is obtained when **p** is perpendicular to the $\mathbf{i}^{th}$ face (actually, to the triangle formed with **O** and the $\mathbf{i}^{th}$ quad diagonal don't having **O**).

The direction **p** is unknown, thus the worst $\mathbf{j}_i$ is also unknown; however, we can say that it is worse (lesser) than the worst $\mathbf{o}\mathbf{a}_i \times \mathbf{a}_{i+1}\mathbf{o}$. Therefore, the quality of the mesh is a priori conditioned by the quality of the exterior given mesh, being the worst two-dimensional $^2\mathbf{j}_i$ of the quads.

This simple bound is a poor approximation; the scalar product will lower too much the actual quality in acute polyhedral angles.

As we only need the direction **p**, the point of interest lies in $\mathbf{S}^2\mathbf{o}$. If we plot **j** in a two-dimensional chart for $\mathbf{S}^2\mathbf{o}$, may be against the spherical angles, we will find that each $\mathbf{j}_i$ gives a square sinusoidal wave with maximum at $\mathbf{a}_i \times \mathbf{a}_{i+1}$, semi-amplitude $^2\mathbf{j}_i$, and 2π wavelength.

We must find a maximum on the lower envelope of the set of these surfaces giving $\mathbf{j}_i$. This point will lie under the intersection of all the positive portions of the n surfaces. This intersection exists because the spherical polygon of the intersection of $\mathbf{S}^2\mathbf{o}$ with the quads sharing **O** has no crossing or coincident edges; thus the maximum angular distance between any two $\mathbf{a}_i \times \mathbf{a}_{i+1}$ will be less than $\boldsymbol{\pi}$.

The maximum point could belong to surface-maximum or to the intersection of two or three surfaces. Then as a local maximum problem, the calculation of the bound has the same difficulty level as the calculation of the optimal position of any node.

Drawbacks

There are two main drawbacks in using this metric.

An angular-only metric yields elements with other malformations. We can mix it with other aspects of quality to maximize a weighed-sum with aspect ratios, planarity of faces, size gradient, etc. However, we are paying attention to what happens with the ortogonality requirement only because traditional methods give us meshes with inverted elements.

The other problem has to do with the implementation: when the node has a negative metric, the gradient of the lower bound may address it to the wrong direction. To the left of Figure 2 the surface is negative and outside the minus-one semicircle. Here, the gradient tells the point to move away the line, where **j** grows asymptotically approaching zero, rising the metric value but in the wrong direction. In the positive portion, we don't have such a problem.

Thus, when the metric is negative we use the triple product without dividing it by the arm lengths. The corresponding surface, in 2D, is a plane crossing the dividing line and tangent to the negative arc (all the triangles/tetrahedrons with the same base and the same height have the same area/volume). This is not a new value, because we know it is negative before dividing it by the arm lengths. In any case, one doesn't care about the magnitude of a negative metric, it must be reverted.

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 formed by the node and its neighbors**.

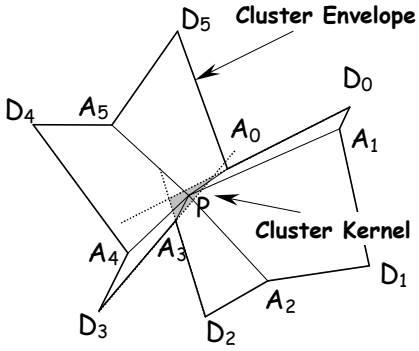

Figure 5: Envelope and Kernel of the attached elements

If we analyze Figure 3b) and Figure 5, we can easily see that rising the quality of a node may negatively affect that of its neighbors. Even in the kernel of the adjacent nodes, where the node **P** is positive-metric, some of the attached nodes can have 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-envelope's 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**. However, if it is difficult to obtain in 2D, our 3D case is even harder. Any algorithm to do that will be welcomed.

So, in our algorithm, **a position is considered as better if the worst quality of the cluster is better or if it is equal and that of the node is better**.

Every attempt we made (with serious meshes) by using analytical ways invariantly ended up with some locking positions often with negative metric.

In the actual program we sort the nodes by quality and sweep the list of worst nodes while there are changes greater than a given value.

To change a node position we make three main attempts:

1) **Gradient driven**: As was explained, the gradient direction improves the node quality but may lower that of the attached ones. Thus, we make a minimum movement in that direction, if it rises the quality we find the maximum successful step.

The obvious choice for the reference step is:

$$\|\Delta\mathbf{p}\| = \frac{1\text{-}\mathbf{j}}{\|\nabla\mathbf{j}\|} \Rightarrow \Delta\mathbf{p} = \frac{1\text{-}\mathbf{j}}{\|\nabla\mathbf{j}\|^2}\nabla\mathbf{j}\,. \tag{4}$$

This step approximates **j** to one, but it has proven less efficient than the mean arm length:

$$\|\Delta\mathbf{p}\| = \langle\mathbf{a}\rangle = \frac{\Sigma\|\mathbf{a}_i - \mathbf{p}\|}{n}\,. \tag{5}$$

The testing attempt is 1/128 of the reference, if it is successful, we test the reference step and then each attempt is half the precedent. The last one is the first to be successful.

2) **Simulated annealing**: If step 1 fails, the node is moved and tested randomly.

The movement steps are in random directions and its magnitudes are random fractions of the mean arm length:

$$\|\Delta \mathbf{p}\| = \langle \mathbf{a} \rangle \times \mathbf{f} \text{ , where } \mathbf{f} \in (0,1]. \quad (6)$$

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) There are five sweeps with this node unchanged,

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

then the node and the attached nodes are forced to move.

This movement is in order 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 at the domain frontier. It is useless to work inside this layer and we chose to move only those nodes at the interface with the good positioned ones.

All the factors and functions chosen were experimentally tested. Nevertheless, we don't expect that these are the best (fastest) values for every situation, however we can say that these values give us positive metric meshes in all our examples and benchmarks. Many of them may seems superfluous, inefficient or extremely conservative, but they are in order to guarantee the result, irrespective it takes a minute or an hour, the speed must no interfere with the robustness.

EXAMPLES:

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

However we tested mostly benchmark, few-element-meshes, without any engineering sense, we 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.

Curved Beam

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

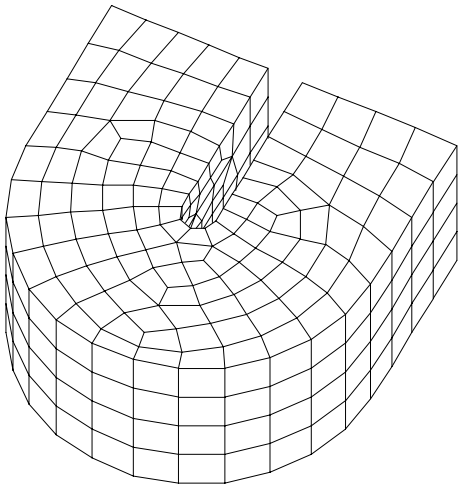

Figure 6: Mesh with a strong concavity

The running test plot shows that, starting with 150 inverted nodes, the situation was reverted in a minute and that the node quality rose to an acceptable level of quality > 0.1 (about 6º or 174º) in less than two minutes.

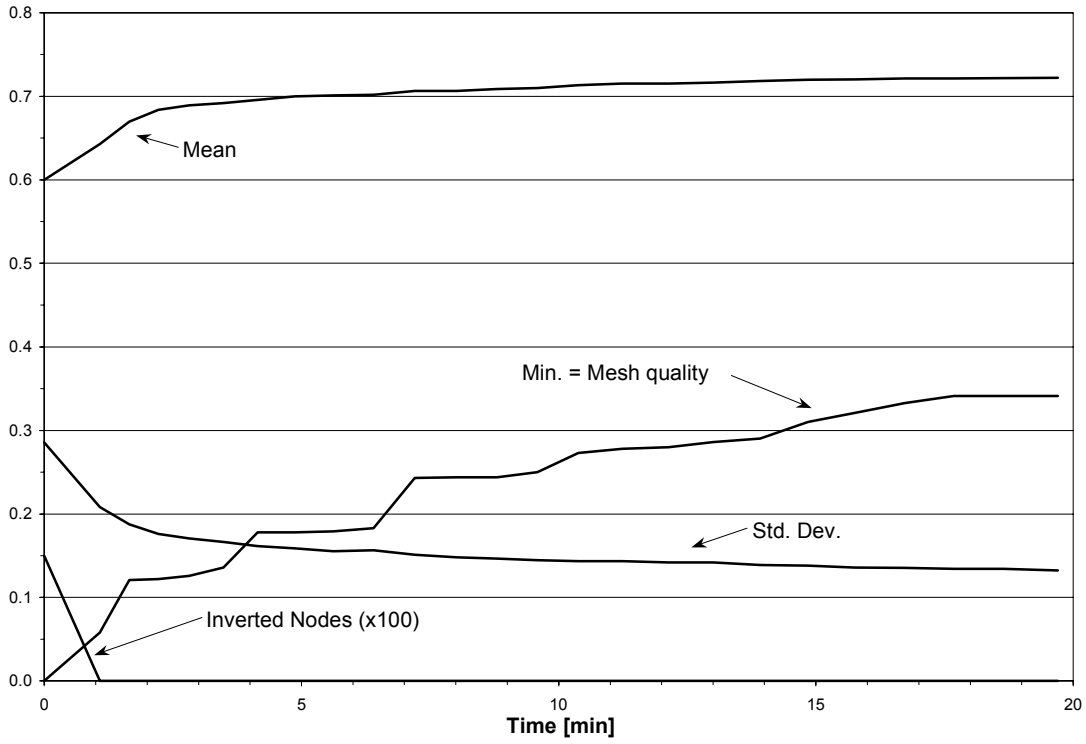

Figure 7: Run plot of quality for the curved beam

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 using a planar paving[10] mesh-generator and then extended to 2½-D and refined in the 3rd dimension whit the help of a CAD program. The all-hexahedral mesh has 9286 nodes and 8210 elements.

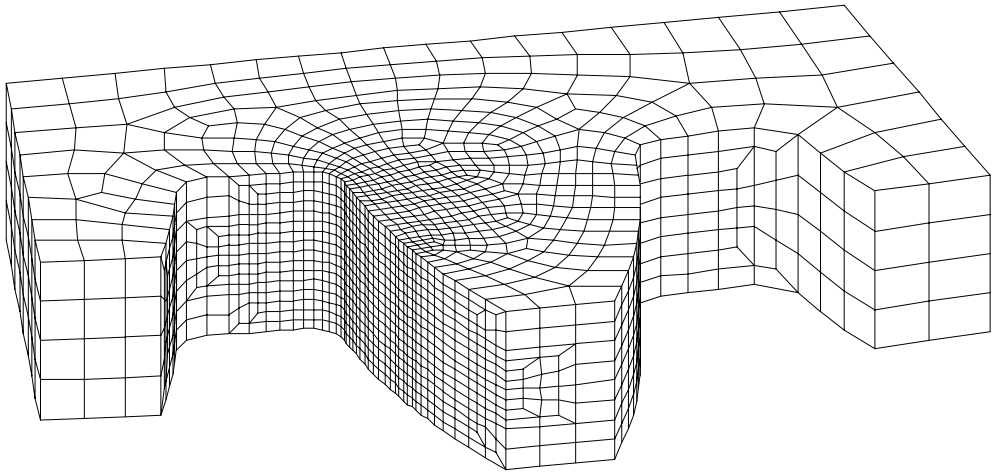

Figure 8: Mesh for a gear tooth

The chart shows that the initial situation with 404 inverted elements was reverted in about 10 minutes and the quality rose to acceptable level in 15 min.

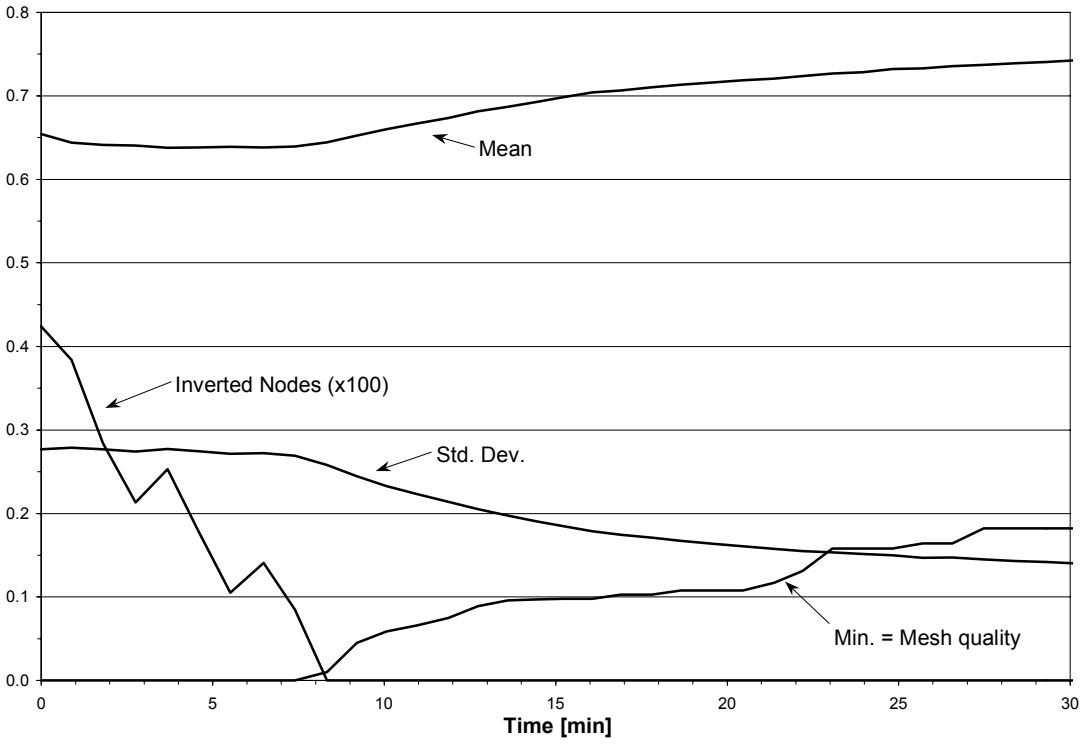

Figure 9: Run plot of quality for the tooth.

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.

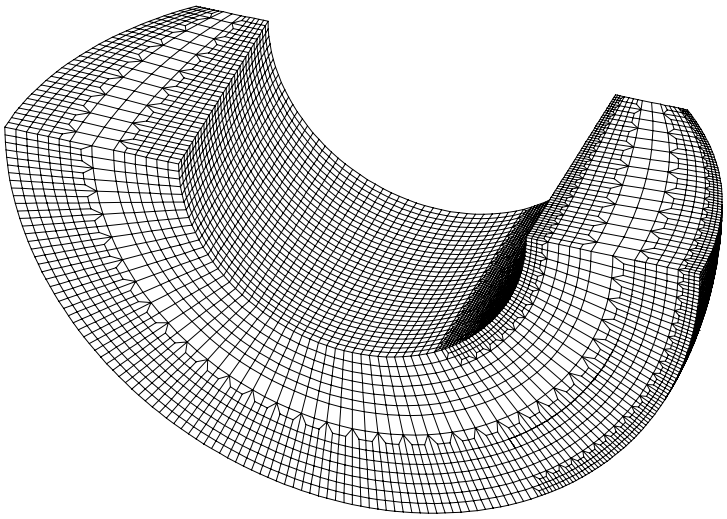

Figure 10: CAD Mesh for a Spherical Bearing

The following chart shows that, starting with the Laplacian smoothed mesh, the initially inverted 56 nodes were corrected in less than a minute, and it took a total of 11 min to rise the quality.

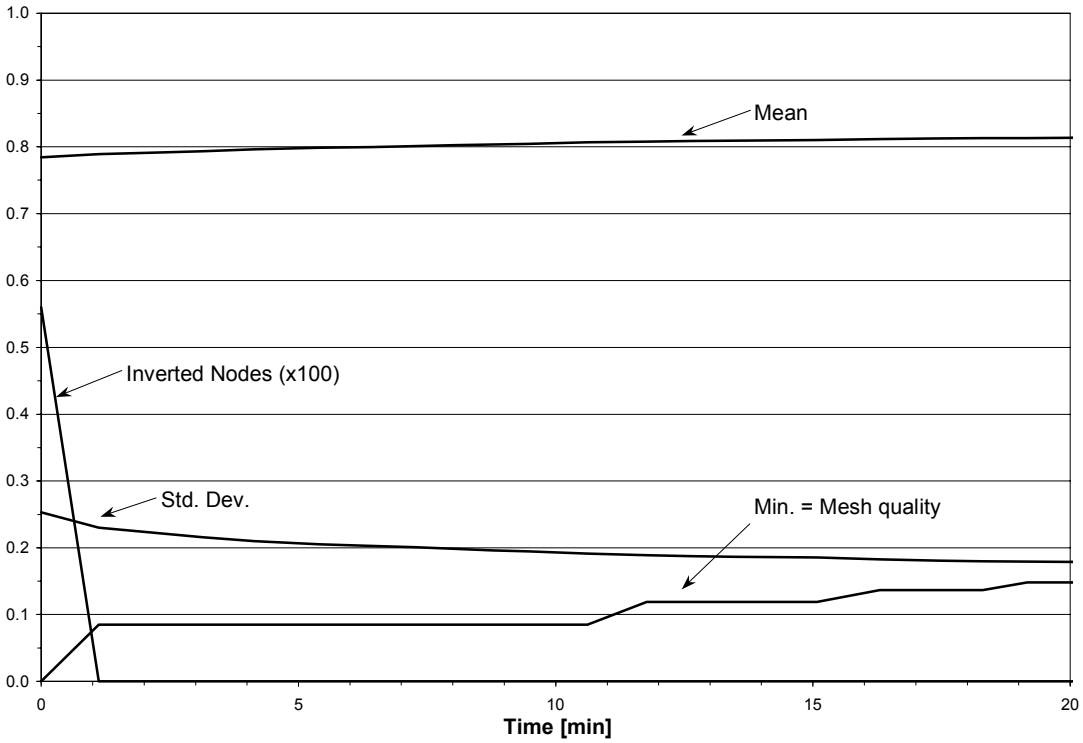

Figure 11: Run plot of quality for the spherical bearing starting with the Laplacian-smoothed mesh.

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.

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

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 attacked with 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', *Proceedings of the 7th International Meshing Roundtable,* 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] N. A. Calvo and S. R. Idelsohn, 'Generador automático de mallas de hexaedros: presentación del método y avances en la implementación', *Mecánica Computacion*al Vol XVI, Asociación Argentina de Mecánica Computacional (1996).

[4] 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.

[5] 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).

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

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

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

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

[10] 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).