A GEOMETRIC MESH SMOOTHING ALGORITHM RELATED TO DAMPED OSCILLATIONS

DIMITRIS VARTZIOTIS AND DORIS BOHNET

Abstract. We introduce a smoothing algorithm for triangle, tetrahedral and hexahedral meshes derived from a simple geometric triangle transformation. Additional to its good numerical performance, the geometric transformation can be modelled as a system of coupled damped oscillations which explains the effectivity of the algorithm.

1. Introduction

A modern product development process requires that more and more pieces of the final product are digitally tested in a very early stage of development to study the influence of the novelties on their essential properties and to detect failures before they occur. Digital tests are also aimed to replace the expensive testing on test stands. A reliable and fast simulation process, the centerpiece of digital testing, is certainly essential for the successful adoption of such a digital product development process. Nearly every simulation in engineering is based on solving partial differential equations with certain numerical solution schemes whose accuracy depends on a good discretization of the object in question. Therefore, every simulation process starts with the preprocessing of the initially given discretization, the so called mesh. This preprocessing should considerably improve the mesh quality where a good mesh quality should always imply faithful results and avoid artifacts of the numerical solution schemes (see e.g. the overview by [9] and the articles [4], [3] regarding computational fluid dynamics).

In order to improve the mesh quality, one usually relocates nodes (called smoothing) and changes the mesh connectivity (called reconnecting or modifying the topology). We focus in this article on mesh smoothing techniques where the two main approaches are the geometric and optimization-based approaches. The geometric approach directly moves the nodes such that the overall mesh quality ameliorates. One classical and widely used example is the Laplacian method where every node is recursively mapped onto the barycenter of its neighboring nodes. While it is usually fast and easily parallelizable, it can result in distorted or even inverted mesh elements in non-convex regions so that there exist several, mainly heuristic enhancements of this method (e.g. the already itself classical isoparametric Laplacian method established by [2]). In contrast to the geometric, the optimization-based approach relies on the computation of the local optimum by optimizational schemes of an objective function which expresses the global mesh quality and is therefore clearly effective as long as the objective function is convex and differentiable. As optimizational schemes one can employ classical methods as the conjugate gradient or newton method in the same way as more recent approaches like evolutionary algorithms (see e.g. [7], [3]). The main disadvantages of the optimization-based approach are its computational cost and the difficulty to choose an appropriate objective function.
In this article we present a geometric mesh smoothing algorithm for triangle, tetrahedral and hexahedral meshes which is based on one single simple triangle transformation. It is in line with the geometric transformation methods developed in a serial of articles (see [12] and citations within) and mathematically analyzed in [11]. Endowed with the good computational properties of a geometric method, its effectiveness can be mathematically proved for a certain subset of triangle meshes and also established by its correlation to a system of differential equations. The derivation of the algorithm and the presentation of its interesting mathematical properties are the main objective of our article and are the topic of §2. But we also give an outlook on its numerical smoothing results in §3.

2. The geometric triangle transformation

We call a triple \( \Delta = (x_0, x_1, x_2) \) of non-collinear points in the euclidean plane \( \mathbb{R}^2 \) or space \( \mathbb{R}^3 \) a triangle. A triangle transformation is then a map \( \theta : \mathbb{R}^n \to \mathbb{R}^n \) for \( n = (3 \times 2), (3 \times 3) \) which maps a triangle \( \Delta \) onto a triangle \( \theta(\Delta) \). We call it geometric if \( \theta \) is compatible with the isometry group actions of the euclidean plane (or space, respectively), that is, \( \theta \) commutes with reflections and translations. Any geometric triangle transformation can be generalized to a triangle mesh transformation by applying it separately to any element of the mesh and then mapping every vertex to the barycenter of its images under the triangle transformations. To guarantee a smoothing effect of such a triangle mesh transformation the geometric triangle transformation must fulfill certain criteria, first of all, \( \theta^n(\Delta) \) with \( n \to \infty \) should converge to an equilateral triangle. On the other hand, the convergence should not be too fast because - if applied to a mesh of triangles - this could disturb the global smoothing effect as obviously not any mesh topology admits a mesh of equilateral triangles.

In the following we present a geometric triangle transformation which has its offspring in the symmetry group of a triangle: an equilateral triangle is preserved by rotations around the centroid by 120° and reflections at the medians. One could imitate the rotational group action on an arbitrary triangle by rotating the vertices around the centroid. More precisely, if the triple \( (x_0, x_1, x_2) \), \( x_i \in \mathbb{R}^2 \) for \( i \in \mathbb{Z}^3 \), defines a planar triangle with centroid \( c = (0, 0) \), we rotate \( x_i \) counter-clockwisely around \( c \) onto the vector \( x_{i+1} \), and denote this new vertex by \( x_{i,new} \). We then push the triangle’s centroid back into the origin. If we repeat this procedure, one observes that the triangle converges to an equilateral one. See Figure 1 for an illustration. This simple transformation fulfills the criteria to be a suitable base for a mesh smoothing algorithm, i.e., it regularizes an arbitrary triangle, and – as we see in this article – has other nice mathematical properties.

![Figure 1. The first three iterations of the geometric element transformation represented as rotations: observe how the circle radii approach and the centroid moves to the circumcenter.](image-url)
2.1. Definition of the geometric element transformation. After these short introductory observations we proceed now with the precise definition of the geometric triangle transformation which is the main subject of this article: let $\Delta = (x_0, x_1, x_2) \in (\mathbb{R}^2)^3$ be a planar triangle, indexed counter-clockwisely, with centroid $c = (1/3)(x_0 + x_1 + x_2)$. Then we define

$$\theta(x_0, x_1, x_2) = (x_{0,\text{new}}, x_{1,\text{new}}, x_{2,\text{new}}), \quad x_{i,\text{new}} = r_i(x_i - c) + c, \quad i = 0, 1, 2,$$

where $r_i = \|x_{i-1} - c\|_2 / \|x_i - c\|_2^{-1}$ for $i \in \mathbb{Z}_3$. In order to keep the centroid fixed throughout the transformation, we map $x_{i,\text{new}}$ onto $x_{i,\text{new}} - c_{\text{new}} + c$, where $i \in \mathbb{Z}_3$ and $c_{\text{new}}$ the centroid of $\theta(\Delta)$. Combining these two transformations, we redefine $\theta$ by

$$(1) \quad x_{i,\text{new}} = (1/3)(2r_i(x_i - c) - r_{i+1}(x_{i+1} - c) - r_{i-1}(x_{i-1} - c)) + c, \quad i \in \mathbb{Z}_3.$$

Remark 0.1. Let $\Delta = (x_0, x_1, x_2) \in (\mathbb{R}^2)^3$ be a triangle and $E$ the plane spanned by the vectors $(x_1 - x_0), (x_2 - x_0)$. Then one easily computes that the image $\theta(\Delta)$ lies in $E$, too. Therefore, if one considers a single triangle, it suffices to study a planar triangle.

2.2. Mathematical properties of the geometric element transformation. Let $X$ be the set of non-collinear triples $(x_0, x_1, x_2) \in (\mathbb{R}^2)^3$ which define planar triangles. The geometric triangle transformation $\theta : X \rightarrow X$ is obviously well-defined; i.e. it maps a triangle onto a triangle. Further, one easily observes that $\theta$ commutes with the isometry group action on the plane, that is, it does not matter if one first rotates, reflects or translates a triangle and then applies $\theta$ or conversely, first applies $\theta$. This property is desirable for triangle mesh transformations as the transformation should not depend on the position of the mesh in space. Additionally, $\theta$ is invariant under rescaling of a triangle. One could say that $\theta$ is compatible with the similarity group actions on a triangle, more exactly:

Lemma 0.1. Let $\Delta$ and $\Delta'$ be two similar triangles defined by triples $(x_0, x_1, x_2)$ and $(x_0', x_1', x_2') \in (\mathbb{R}^2)^3$, i.e. there exist $A \in \text{SO}(2, \mathbb{R})$, $\sigma \in \mathbb{R}^2$ and $a \in \mathbb{R}^+$ such that $a(Ax_i + \sigma) = x_i'$ for $i = 0, 1, 2$. Then the images $\theta(\Delta)$ and $\theta(\Delta')$ are also similar two each other, in particular, it holds that $a(Ax_{i,\text{new}} + \sigma) = x_{i,\text{new}}'$ for $i = 0, 1, 2$.

The proof of this lemma is a simple computation, using the fact that rotations and translations are isometries and that any norm is absolutely homogeneous. Further and more importantly, one can prove that $\theta$ regularizes any triangle:

Theorem 1. Let $\Delta \in X$ be given. Then $\theta^n(\Delta)$ for $n \rightarrow \infty$ converges to an equilateral triangle.

The proof of this theorem can be found in [10]. It is based on the fact that a triangle is equilateral if and only if the distances from the vertices to the centroid are all equal and that the sequence of fractions of these distances for $\theta^n(\Delta)$ converges to one.

Further, one could prove the following stronger result:

Theorem 2. Let $\Delta_{eq} \in X$ be an equilateral triangle and $\Lambda$ the set of triangles similar to it. Then $\Lambda$ is a global attractor for $\theta$. In particular, for any planar triangle $\Delta$ the sequence $\theta^n(\Delta)$ converges uniformly at exponential rate to one point in $\Lambda$ for $n \rightarrow \infty$.

The proof of this theorem in [10] relies on an analysis of the dynamics of the map $\theta$ which leads to the observation that the eigenvalues of the Jacobian matrix of $\theta$ at an equilateral triangle are solely responsible for the dynamical properties of the map. More precisely, it suffices to prove that the absolute values of all
eigenvalues are strictly smaller than one apart from four eigenvalues equal to one which correspond to the four-dimensional tangent space of $\Lambda$. Using Theorem 1 this allows then to conclude that $\Lambda$ is a global attractor.

These results could be generalized to a certain compact subset of planar triangle meshes asserting the effectivity of the triangle mesh transformation derived from $\theta$. One notices that the transformation is not orientation-preserving on the whole set of planar triangle meshes (see Figure 2), for example, if two distorted triangles share one edge, the orientation of one of the triangles could be reversed. For that reason, one defines a compact subset of meshes whose element distortion is bounded from below and proves that the mesh transformation $\Theta$ is orientation-preserving on this subset. The necessary computations to assure the effectivity of the triangle mesh transformation involve the analysis of huge jacobian matrices, and the limited knowledge of the exact nature of these matrices restricts the cases where we can really prove the global convergence of this mesh transformation. For that reason, we do not further explore these quite technical mathematical aspects here, but we would rather like to describe how to correlate this discrete transformation to a system of linear differential equations. This is the objective of the following section:

2.3. Relation to a system of damped oscillations. Looking for a mathematically simpler, but sufficiently adequate model to handle the transformation $\theta$, we describe the dynamics of $\theta$ by differential equations and try to bypass the computational difficulties in this way. The following observations together with Subsection 2.2 provide also an evidence for the numerical results in § 3.

2.3.1. Derivation and solution of a system of ODEs. One observes in Figure 2 that the quantities, responsible for the convergence properties of $\theta$, are the distances $R_i = ||x_i - c||_2$ of the vertices $x_i$, $i = 0, 1, 2$, to the centroid $c$. So, we consider these as time-dependent variables $R_i(t)$ and describe their dynamics by differential equations. First of all, the fixed points of $\theta$ are exactly the equilateral triangles, so we have to assure that $R_i(t) = 0$ for $i = 0, 1, 2$ if and only if the distances $R_0(t) = R_1(t) = R_2(t)$ are equal and therefore constant. Further, we observe that the distance $R_0(t)$ increases if $R_2(t)$ is greater than the average distance and decreases otherwise, so we can set

$$\dot{R}_0(t) = R_2(t) - \frac{1}{\sqrt{3}}(R_0(t) + R_1(t) + R_2(t)) = (2/3)R_2(t) - \frac{1}{\sqrt{3}}(R_0(t) + R_1(t)).$$

In this way we get for the distance vector $R(t) = (R_0(t), R_1(t), R_2(t))$ the following system of linear differential equations:

$$\begin{pmatrix} \dot{R}_0(t) \\ \dot{R}_1(t) \\ \dot{R}_2(t) \end{pmatrix} = \begin{pmatrix} -1/3 & -1/3 & 2/\sqrt{3} \\ 2/\sqrt{3} & -1/3 & -1/3 \\ -1/3 & 2/3 & -1/3 \end{pmatrix} \begin{pmatrix} R_0(t) \\ R_1(t) \\ R_2(t) \end{pmatrix} \quad \text{with} \quad \begin{pmatrix} R_0(0) \\ R_1(0) \\ R_2(0) \end{pmatrix} = \begin{pmatrix} R_0 \\ R_1 \\ R_2 \end{pmatrix}.$$
One easily computes that this system has as stationary solutions the constant vectors \((a, a, a) \in \mathbb{R}^3, a \in \mathbb{R}^+\) which correspond to the equilateral triangles.

**Remark 2.1.** We would like to stress that the presented system of ODEs is a model for the geometric transformation. It is not the case that the transformation is the discretization of the continuous system, e.g. derived from the Euler method.

Let us now shortly discuss the dynamics of this system of ODEs:

### 2.3.2. Dynamics of the continuous solutions.

Solving a system of linear differential equations is a standard technique: we compute the eigenvalues of the coefficient matrix as \(\lambda_0 = 0\) and a pair of complex conjugate eigenvalues \(\lambda_{1,2} = -1/2 \pm \sqrt{3}/2\) with corresponding eigenvectors \(v_0 = (1, 1, 1)\), and \(v_{1,2} = (-1/2 \pm \sqrt{3}/2, 1, -1/2 \mp \sqrt{3}/2)\). Denoting the initial values by \((R_0, R_1, R_2)\) we obtain as general solution for \(R_i(t), i \in \mathbb{Z}_3\),

\[
R_i(t) = \left(\frac{1}{\sqrt{3}}\right)(R_0, R_1, R_2) + \left(\frac{1}{\sqrt{3}}\right) \exp(-t/2)(c_i \cos(\sqrt{3}t/2) + s_i \sin(\sqrt{3}t/2)),
\]

using the abbreviations \(c_i = 2R_i - R_{i+1} - R_{i+2}\) and \(s_i = \sqrt{3}(R_{i+2} - R_{i+1})\).

As the distances between vertices and centroid for any triangle are positive, we get a half line \(\{(a, a, a) \mid a \in \mathbb{R}^+\}\) of constant solutions as equilibria which correspond to the equilateral triangles. Set \(\alpha = (1/\sqrt{3})(R_0 + R_1 + R_2)\) as the average distance of the initial triangle. As cosine and sinus are bounded by 1, we have for \(i = 0, 1, 2\)

\[
|R_i(t) - \alpha| \leq \left(\frac{1}{\sqrt{3}}\right) \exp(-t/2) \rightarrow 0 \quad \text{for } t \rightarrow +\infty.
\]

Consequently, every solution \(R(t)\) converges at exponential rate to the constant solution \((\alpha, \alpha, \alpha)\). The dynamics could be easily deduced from Figure 3. One could fill in the solution \(R(t)\), replacing \(R_0, R_1, R_2\), into the transformation \(\theta\) to convert it into a continuous process by \(x_i(t)\). This allows the observation that every vertex spirals slowly around the corresponding vertex of the limit triangle while approaching it. If one compares the limit triangle of this continuous process with the one obtained by \(\theta\) (e.g. in Figure 1), one observes that they coincide which affirms that our model is adequate.

### 2.3.3. Comparison between continuous solution and discrete transformation.

For further validation of our continuous model we compare the results by numerical computations: let \(\Delta\) be a planar triangle with distances \(R_0, R_1, R_2\) from its centroid to its vertices. Chosing an appropriate discretization for \(t\), we plot the continuous curves \(R(t)\) given by the solution of Equation (2) against the discrete iterates \(R^n\) for \(n \in \mathbb{N}\) which denote the distances of the \(n\)th iterate \(\theta^n(\Delta)\) of the initial triangle \(\Delta\). Figure 4 shows the accurateness of the continuous description for the discrete transformation and the rapid convergence of the transformation to a regular triangle.

### 2.3.4. Remarks on the relation to damped oscillations.

Besides from these nice dynamical properties, the system of linear differential equations (2) above provides a link to a physical model: looking at the solution (3) we observe that

\[
\dot{R}(t) = \left(\frac{1}{\sqrt{3}}\right) \left(\begin{array}{c}
2R_2 - R_1 - R_0 \\
2R_0 - R_1 - R_2 \\
2R_1 - R_2 - R_0
\end{array}\right) \exp(-t/2) + \left(\begin{array}{c}
\sqrt{3}(R_1 - R_0) \\
\sqrt{3}(R_2 - R_1) \\
\sqrt{3}(R_0 - R_2)
\end{array}\right) \sin(\sqrt{3}t/2) = \left(\begin{array}{c}
R_2(t) - \alpha \\
R_0(t) - \alpha \\
R_1(t) - \alpha
\end{array}\right).
\]
In particular, we get
\[
\dot{R}(0) = \begin{pmatrix}
R_2 - \alpha \\
R_0 - \alpha \\
R_1 - \alpha
\end{pmatrix} = \left(\frac{1}{3}\right) \begin{pmatrix}
2R_2 - R_0 - R_1 \\
2R_0 - R_1 - R_2 \\
2R_1 - R_0 - R_2
\end{pmatrix}.
\]

Accordingly, the system of linear differential equations could be equivalently written as three linear differential equations of second order where
\[
(R_0, R_1, R_2) = (R_i(0), R_i(0), R_i(0))
\]
are the initial values where \(i \in \mathbb{Z}_3\):
\[
\begin{align*}
\ddot{R}_i(t) + \dot{R}_i(t) + R_i(t) &= (\frac{1}{3})(R_i + R_{i+1} + R_{i+2}), \\
R_i(0) &= R_i, \quad \dot{R}_i(0) = \frac{2}{3}R_{i+2} - \frac{1}{3}R_{i+1} - \frac{1}{3}R_i.
\end{align*}
\]

Each of these equations can be seen as describing a damped oscillation. Consequently, the dynamics of each distance \(R_i(t), R_i(t)\) and \(R_i(t)\) can be understood as the dynamics of a damped oscillation which depend on the initial values of each other. Adopting the language of oscillations the dynamics of our system is governed by a spring constant equal to 1 and a damping constant equal to 1.

2.3.5. **Controlling the efficiency of the transformation.** The illustration of the transformation as a damped oscillation helps to understand how one can control the speed of the convergence for each triangle to achieve a better global smoothing effect. For this purpose let us introduce three parameters \(\alpha_i, i = 0, 1, 2\), which allow us to change proportionally the ratios \(r_i, i = 0, 1, 2\). So instead of mapping \(x_{i,new} = r_i x_i\) for \(i = 0, 1, 2\) we factorize by the parameter \(\alpha_i\) getting \(x_{i,new} = \alpha_i r_i x_i\) for \(i = 0, 1, 2\). If we keep the centroid in the origin during the transformation we obtain for \(i \in \mathbb{Z}_3\)
\[
\begin{align*}
x_{i,new} &= (\frac{2}{3})\alpha_1 r_i x_i - (\frac{1}{3})\alpha_1 r_{i+1} x_{i+1} - (\frac{1}{3})\alpha_1 r_{i+2} x_{i+2}.
\end{align*}
\]

As the equilateral triangle should be the fixed point of this transformation we impose that \(-\alpha_1/3 - \alpha_2/3 + 2\alpha_0/3 = 0\). Using this observation we change the differential equations for \(R(t)\) adequately and compute the solution depending on \(\alpha_0\) and \(\alpha_1\). This allows us to conclude how \(\alpha_0\) and \(\alpha_1\) have to be chosen to control the limit radius \(a\) where \(R(t)\) converges to. The greater \(\alpha_0\), the faster is the convergence, but at the same time the greater the limit radius \(a\). Sometimes, it is preferable to decelerate a bit the convergence of each triangle element to an equilateral triangle...
in order to improve the global convergence of the triangle mesh. This could be done by choosing $\alpha_0$ sufficiently small.

Inspired by the behavior of the continuous solution we adapt our discrete transformation and our implementation: By iterating over the parameter ranges $\alpha_0 = i/10, \alpha_1 = j/10$ for $i, j = 1, \ldots, 20$ we determine as optimal parameter $\alpha_0 = 0.3, \alpha_1 = 0.1$ which we used to smooth a randomly generated triangulation of the unit square, see Figure 5. This means that — for the test case — it is recommended to slow the convergence of the triangle transformation. For details to the implementation and used quality measure see § 3. Looking also at the comparative diagrams in Figure 5 one infers that the use of adapted parameters is a viable possibility to make the current approach more effective. Like an adaptive stepsize strategy one should choose the parameters according to the convergence behavior of the algorithm at every element in the preceding iteration step.

3. The mesh smoothing algorithm based on the geometric element transformation

3.1. Mesh transformation for triangle and tetrahedral meshes. As already described at the beginning, any triangle transformation could be used to define a triangle mesh transformation: firstly, apply the triangle transformation to every triangle element of the mesh, secondly, map the vertex onto the barycenter of its images under the triangle transformations. One could say that one combines the elementwise geometric triangle transformation with a Laplace transformation. Let us be more precise: let $M_\Delta$ be a triangle mesh defined by the set $V = (x_0, \ldots, x_{N-1})$, $x_i \in \mathbb{R}^2$ or $x_i \in \mathbb{R}^3$, of vertices, indexed counter-clockwisely, and the set $E = (\Delta_0, \Delta_1, \ldots, \Delta_{n-1})$ of triangle elements $\Delta_i = (i_0, i_1, i_2)$ with $i_j \in \{0, \ldots, N-1\}$. We define the triangle mesh algorithm in the following way:

**Step 1:** for $i = 0, \ldots, N-1$, find triangles $\Delta$ which contain the index $i$.

Denote the set of indices of adjacent triangles by $J(i)$.

**Step 2:** For $i = 0, \ldots, N-1$ update vertex $x_i$ as following: for $k \in J(i)$ iterate the triangle $\Delta_k$ formed by $(x_{k_0}, x_{k_1}, x_{k_2})$ with $\theta_k$ given by (1) to $\Delta_{k, new}$. Rescale $\Delta_{k, new}$ to keep the area constant. If $\Delta_{k, new}$ is flipped, that is, if its orientation is reversed, then reset $\Delta_{k, new} = \Delta_k$. Update the vertex $x_i$ by the arithmetic mean $\frac{1}{|J(i)|} \sum_{k \in J(i)} \theta_k(x_i)$.

**Step 3:** Apply boundary constraints.

![Figure 5. Influence of adaptive parameters: initial triangulation with mean aspect ratio of $q = 0.478$ on the left, in the middle after 5 iterations with $q = 0.649$ and on the right after 5 iterations using adaptive parameters $\alpha_0 = 0.3$ and $\alpha_1 = 0.1$ reaching $q = 0.662$.](image-url)
Written in this way, this algorithm can be directly implemented into any mathematical software or programming language.

The use of this transformation can be naturally extended to a volume mesh of tetrahedra by transforming the triangle faces of each tetrahedron: let $M_T$ be a mesh defined by the set $V = (x_0, \ldots, x_{N-1})$ of vertices $x_i \in \mathbb{R}^3$ and the set $E = (T_0, \ldots, T_{n-1})$ of tetrahedral elements $T_i = (i_0, i_1, i_2, i_3)$ with $i_j \in \{0, \ldots, N-1\}$. We implemented the smoothing algorithm as following:

**Step 1:** for $i = 0, \ldots, N-1$, find tetrahedra $T$ which contain $x_i$. Denote the set of indices of adjacent tetrahedra by $J(i)$.

**Step 2:** For $i = 0, \ldots, N-1$ update vertex $x_i$ as following: for $k \in J(i)$ iterate the tetrahedron $T_k$ to $T_{k,\text{new}}$ by

1. compute its four triangle faces $(x_{k0}, x_{k1}, x_{k2})$, $(x_{k1}, x_{k2}, x_{k3})$, $(x_{k0}, x_{k2}, x_{k3})$ and $(x_{k0}, x_{k1}, x_{k3})$ using the geometric element transformation (1) to obtain $T_{k,\text{new}}$.

2. Rescale $T_{k,\text{new}} = (x_{k0}, x_{k1}, x_{k2}, x_{k3})$ to keep its volume constant.

Then update $x_i$ by the arithmetic mean $\frac{1}{|J(i)|} \sum_{k \in J(i)} x_i$.

**Step 3:** Apply boundary constraints.

In the implementation we executed Step 2.1 three times in every iteration as this has proved to be more efficient.

### 3.2. Derived algorithm for hexahedral meshes.

Apart from triangular and tetrahedral meshes, hexahedral meshes are maybe the most important class of meshes for applications. This is our main motivation to generalize our algorithm to hexahedral meshes. We use the fact that every hexahedron defines an octahedron whose vertices are the barycenters of the six faces of the hexahedron, see Figure 6 as an illustration. Conversely, every octahedron determines a hexahedron by taking the barycenters of its eight faces. In this way, we compute to every hexahedron its corresponding octahedron. This could be treated as a closed triangle mesh. Let us now define more precisely our algorithm:

Let $M_H$ be a hexahedral mesh defined by the set $V = (x_0, \ldots, x_{N-1})$, $x_i \in \mathbb{R}^3$, of vertices, indexed counter-clockwisely, and the set $E = (H_0, \ldots, H_{n-1})$ of hexahedral elements $H_i = (i_0, \ldots, i_7)$ with $i_j \in \{0, \ldots, N-1\}$. The smoothing algorithm is then defined in the following way:

**Step 1:** for $i = 0, \ldots, N-1$, find hexahedra $H$ which contain $x_i$. Denote the set of indices of adjacent hexahedra by $J(i)$.

**Step 2:** For $i = 0, \ldots, N-1$ update vertex $x_i$ as following: for $k \in J(i)$ iterate the hexahedron $H_k$ to $H_{k,\text{new}}$ by

1. compute its dual octahedron $O_k$ whose vertices are defined by the barycenters of the faces of $H_k$.

![Figure 6. A regular hexahedron and its dual octahedron in red.](image)
(2) Consider its eight triangle faces as closed triangle mesh and iterate it using the geometric element transformation (1) to obtain $O_{k,\text{new}}$.

(3) Compute the new hexahedron $H_{k,\text{new}} = (\bar{x}_{k0}, \ldots, \bar{x}_{k7})$ by taking the barycenters of the triangle faces of $O_{k,\text{new}}$.

(4) Rescale $H_{k,\text{new}}$ to keep its volume constant.
Then update $x_i$ by the arithmetic mean $\frac{1}{\left| J(i) \right|} \sum_{k \in J(i)} \bar{x}_i$.

**Step 3:** Apply boundary constraints.

**Remark 2.2.**

(1) An equivalent application of our algorithm to a hexahedral mesh is the following: any hexahedron could be subdivided into four tetrahedra whose edges are given by the diagonals of the faces of the hexahedron. Apply then the algorithm to this mesh of four tetrahedra.

(2) Any platonic solid, these are tetrahedra, hexahedra, octahedra, dodecahedra and icosahedra, can be subdivided into regular tetrahedra. In this way, one could apply our algorithm to any mesh built by these polyhedra.

### 3.3. Numerical results and discussion on mesh quality improvement

In the following we shortly discuss the numerical results of our smoothing algorithm which we implemented in C++. We used the source code architecture of the mesh smoothing tool developed in a serial of articles (see [12] and citations within) which treats different approaches for an efficient implementation of geometric element transformation methods. All elements were updated and their nodes saved as intermediate nodes. Then the nodes were updated by the barycenter of the intermediate nodes. We have chosen two simple triangle, two tetrahedral and two hexahedral meshes as examples and display the mean quality improvement. This article does not focus on computational details, so this section serves more as an outlook that the geometric element transformation, described in this article, could be the base of a very efficient mesh smoothing algorithm which has comparable runtimes to smart Laplace algorithms, but obtains usually better quality results.

Apart from the SmartLaplace algorithm we use a global optimization method implemented in MESQUITE 2.3.0 (Mesh Quality Improvement Toolkit) to compare our obtained smoothing results with regard to element and mean quality improvement and runtime. As objective function for the global optimization we select the inverse mean ratio quality and as numerical optimization scheme the feasible Newton method (see user’s guide [5] for details). For all testing we use the same personal computer equipped with a quad-core-processor (Intel(R) Core(TM) i7 CPU 870 @293 GHz, 1197 MHz). The source code was compiled using g++ under Linux. The algorithms are in all cases stopped if the mean mesh quality improvement was less than $10^{-4}$ during the last iteration step. All sample meshes are valid meshes, that is, without inverted elements.

#### 3.3.1. Quality measures

We briefly introduce the quality measures which we use for quality assessment of the mesh smoothing algorithms.

**Triangle mesh.** As quality measure $q_\Delta$ for a single triangle element we use the ratio of minimal to maximal edge lengths of every triangle. The quality measure for a triangle mesh $V = (\Delta_0, \ldots, \Delta_{\left| V \right|-1})$ is then the mean of the quality measure $q_\Delta$ for every triangle $\Delta \in V$: $q_V = \frac{1}{\left| V \right|} \sum_{\Delta \in V} q_\Delta$.

**Tetrahedral mesh.** Let $T = (x_0, x_1, x_2, x_3)$ with $x_i \in \mathbb{R}^3$ be a tetrahedron. As quality measure $q_T$ for $T$ we use the mean ratio quality measure which is defined
as following (see [3]):

$$q_T(T) = \frac{3 \det(S)^{2/3}}{\text{trace}(S' S)}, \quad S = D(T) W,$$

where

$$D(T) = (x_1 - x_0, x_2 - x_0, x_3 - x_0), \quad W = \begin{pmatrix} 1 & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ 0 & \sqrt{3}/2 & \sqrt{3}/6 \\ 0 & 0 & \sqrt{2/3} \end{pmatrix}.$$ 

As quality measure for a tetrahedral mesh \( V = (T_0, \ldots, T_{|V|-1}) \) we used the mean quality measure of every element:

$$q_V = \frac{1}{|V|} \sum_{T \in V} q_T(T).$$

Hexahedral mesh. Let \( H = (x_0, \ldots, x_7) \) with \( x_i \in \mathbb{R}^3 \) be a hexahedron. As quality measure \( q_H \) for \( H \) we use the mean ratio quality measure which is defined using a subdivision of \( H \) into eight tetrahedra \( T_0, \ldots, T_7 \) given by

\( T_0 = (x_0, x_3, x_4, x_1), T_1 = (x_1, x_0, x_5, x_2), T_2 = (x_2, x_1, x_6, x_3), T_3 = (x_3, x_2, x_7, x_0), T_4 = (x_4, x_7, x_5, x_0), T_5 = (x_5, x_4, x_6, x_1), T_6 = (x_6, x_5, x_7, x_2) \) and \( T_7 = (x_7, x_6, x_4, x_3) \). The quality measure of \( H \) is then defined as the average of the eight values of the quality measure for the internal tetrahedra, with the difference that \( W \) is set to identity, so we get:

$$q_H(H) = \frac{1}{8} \sum_{k=1}^{8} 3 \det(S_k)^{2/3} / \text{trace}(S_k' S_k), \quad S_k := D(T_k),$$

where

$$D(T_k) = (x_{k,1} - p_{k,0}, x_{k,2} - x_{k,0}, x_{k,3} - x_{k,0}), \quad T_k = (x_{k,0}, x_{k,1}, x_{k,2}, x_{k,3}).$$

As quality measure for a tetrahedral mesh \( V = (T_0, \ldots, T_{|V|-1}) \) we use the mean quality measure of every element: \( q_V = \frac{1}{|V|} \sum_{T \in V} q_T(T) \).

3.3.2. Example 1: triangle mesh. First, we consider a randomly generated triangulation of the unit square as a sample triangle mesh, see Figure 5. The mesh consists of 450 triangle elements. In Figure 7 below we show how the number of elements with a certain quality measure develops over iterating the mesh and how the mean quality improves. Apart from this edge length ratio measure we examined three

![Figure 7](image_url)

Figure 7. Square quality element measure \( q_\Delta \) before and after the smoothing of the mesh in Figure 5 and improvement of mean mesh quality with respect to the iteration steps.

further quality measures for this simple mesh to assess their improvement under the algorithm: a widely used measure for the quality of a triangle is also the measure \( q_\alpha = \min(q_\alpha, q_i) \) with \( q_\alpha = \min(\min(\alpha_k)/\tau_s, 1) \) and \( q_i = \min(\tau_i/\max(\alpha_k), 1) \) where \( \alpha_k, k = 0, 1, 2 \), denote the interior angle of a triangle and \( \tau_s, \tau_i \) the minimum and maximum interior angles allowed. We present in Table 1 the improvement of the quality of our triangle mesh with respect to this quality measure \( q_\alpha \) with fixed \( \tau_s = \pi/6 \), \( \tau_i = 2\pi/3 \).
Apart from these easily calculated quality measures, there exist quality measures which are directly related to the error bounds for the application that uses the mesh, see [9] and [1] for a more detailed discussion and derivation of these measures. For this reason, we have chosen the quality measure \( q_{FE}(\Delta) = \frac{3 \cdot \text{area}(\Delta)}{l_0^2 + l_1^2 + l_2^2} \) where \( l_i = \|x_i - x_{i+1}\|_2 \) for \( i = 0, 1, 2 \mod 3 \) denote the edge lengths. The measure \( q_{FE} \) is a good indicator whether the maximal eigenvalues of the element stiffness matrices inside the finite element method stay small and therefore the condition number of these matrices will be reasonable. The other quality measure we considered for the triangle mesh is

\[
q_\nabla(\Delta) = \frac{\text{area}(\Delta)}{(l_0 \ast l_1 \ast l_2)^{2/3}}
\]

which provides us with the inverse of the error bound for the interpolation error of the gradient. Our results obtained for the three presented quality measures are shown in Table 1.

The next sample mesh displayed in Figure 8 is also a planar triangle mesh which was generated by triangulating a planar disk. It is instantaneously smoothed to a mesh of nearly overall equilateral triangles. We compare once more the mesh smoothed by the standard algorithm and using adapted parameter \( \alpha_0 = 0.3 \) and \( \alpha_1 = 0.1 \) (see §2.3.5) which seem to be a good possibility to improve the current approach looking at the diagrams in Figure 9. One remarkable consequence of the deceleration of the algorithm is its almost orientation-preserving character for the current sample mesh: no element gets inverted by the adapted mesh transformation in comparison to about 300 triangles (for every iteration step) which would get inverted and are therefore not transformed by the standard algorithm.

3.3.3. Example 2: tetrahedral mesh. Our tetrahedral sample mesh (Figure 10) is based on a model provided by the 3D meshes research database GAMMA maintained by INRIA which was meshed by 82958 tetrahedral elements. We apply as comparison a smart Laplace algorithm and Mesquite as described above to the same initial mesh. We have also tested the influence of adapted parameter over a broad parameter range from 0 to 2, but could not find an improvement as significant as for the triangle meshes above. A deceleration of the algorithm had also a sensible negative effect on the runtime. For these reasons we do not show the results for the varied algorithm with adapted parameter for the current sample mesh because either they do not noticeably differ or the quality improvement do not outweigh the negative consequences for the runtime. Our smoothing result shown in Figure 11 is nearly the same than the one obtained with Mesquite, but the runtime (see Table 2) is slightly faster. Looking at the velocity of smoothing it is evident that the quality improvement takes place much more instantaneous for geometric smoothing algorithms than for an optimization-based approach.

3.3.4. Example 3: hexahedral mesh. As hexahedral sample mesh (see Figure 12) we choose a wheel bearing model which was generated by a sweep approach applied to small sub-parts of the mesh (see [6]) and then meshed by 67055 hexahedral elements. The quality assessment displayed in Figure 14 and Table 3 shows
Figure 8. Planar triangle mesh of 38560 elements smoothed using adapted parameter $\alpha_0 = 0.3$ and $\alpha_1 = 0.1$ (see §2.3.5) with initial state on the right.

Figure 9. Histogram of mesh quality improvement and comparison of mesh quality improvement for the triangulated disk of Figure 8 using standard and adapted parameter.

|       | Original | GETMe | Mesquite | SmartLaplace |
|-------|----------|-------|----------|--------------|
| runtime | 6.34 s   | 53.04 s | 6.49 s   | 6.49 s       |
| mean ratio | 0.46     | 0.92  | 0.94     | 0.88         |

Table 2. Comparison of runtime for the hexahedral mesh in Figure 12.

that our algorithm delivers a result between the ones obtained by Mesquite and SmartLaplace (see Figure 13 for smoothing results) but in a runtime comparable to the Laplace algorithm.

|       | Original | GETMe | Mesquite | SmartLaplace |
|-------|----------|-------|----------|--------------|
| runtime | –        | 2.55 s | 3.06 s   | 0.6 s        |
| mean ratio | 0.49     | 0.74  | 0.75     | 0.70         |

Table 3. Comparison of runtime for the tetrahedral mesh of Figure 10.
FIGURE 10. Tetrahedral mesh with 82985 elements with initial mean quality $q_T = 0.49$ improving to $q_T = 0.74$.

FIGURE 11. Histogram of mesh quality improvement and comparison of mesh quality improvement for the tetrahedral mesh of Figure 10.
Figure 12. Cross section of wheel bearing model of 67055 hexahedral elements with initial mean quality $q_H = 0.46$ improving to $q_H = 0.92$.

Figure 13. Smoothing results for the hexahedral mesh of Figure 12 by Mesquite (on the left) and SmartLaplace (on the right).
Figure 14. Comparative histogram of the smoothing results by Mesquite and by SmartLaplace for the hexahedral mesh of Figure 12 on the left and comparison of mesh quality improvement of the global mesh.
4. Concluding Remarks

The presented geometric triangle transformation exhibits interesting mathematical properties and is, at a first glance, appropriate for a broad usage as base of a mesh smoothing algorithm. We did not explore further the possibilities to implement the algorithm in the best and most efficient way possible as this article focuses on mathematical considerations. A first step would be to implement a dynamical choice of the adaptive parameters depending on every single element to improve the performance of the smoothing algorithm. Although the tests with tetrahedral meshes have not yet been convincing, the much better results for planar triangle meshes let hope that one can find an intelligent way to control the velocity of the elementwise convergence of the algorithm in every step such that the overall result improves considerably.

Further, one should use the rotation-like transformation to align elements in the direction of flow creating anisotropic meshes, very important for applications in computational fluid dynamics.

For our experiments we chose valid meshes so that an untangling was not necessary. The question whether the current algorithm could be used to untangle meshes with inverted elements is surely worth to pursue as this would provide an untangling method which does not affect the mesh connectivity.

References

[1] M. Berzins, Mesh quality: A function of geometry, error estimates or both, in Proceedings of the 7th International Meshing Roundtable, 1998, pp. 229–238.
[2] L. Herrmann, Laplacian-isoparametric grid generation scheme, Journal of the Engineering Mechanics Division, 102 (1976), pp. 749–756.
[3] A. Jalali, M. Shariatradar, and C. Ollivier-Gooch, Accuracy analysis of unstructured finite volume discretization schemes for diffusive fluxes, Computers & Fluids, 101 (2014), pp. 220–232.
[4] A. Katz and V. Sankaran, Mesh quality effects on the accuracy of euler and navier-stokes solution on unstructured meshes, J. of Comp. Physics, 230 (2011), pp. 7670–7686.
[5] P. Knupp, L. Freitag-Duchin, and B. Tidwell, Mesquite mesh quality improvement toolkit user’s guide, https://software.sandia.gov/mesquite/, 2013.
[6] P. M. Knupp, Next-generation sweep tool: A method for generating all-hex meshes on two-and-one-half dimensional geometries, in Proceedings of the 7th International Meshing Roundtable, 1998, pp. 505–513.
[7] P. M. Knupp, Achieving finite element mesh quality via optimization of the jacobian matrix norm and associated quantities, part ia framework for surface mesh optimization, International Journal for Numerical Methods in Engineering, 48 (2000), pp. 401–420.
[8] P. M. Knupp, Algebraic mesh quality metrics, SIAM Journal on Scientific Computing, 23 (2001), pp. 193–218.
[9] J. R. Shewchuk, What is a Good Linear Element? Interpolation, Conditioning, and Quality Measures, in Proceedings of the 11th International Meshing Roundtable, 2002, pp. 115–126.
[10] D. Vartziotis and D. Bohnet, Convergence properties of a geometric mesh smoothing algorithm, Preprint, 2014.
[11] D. Vartziotis and B. Himpe1, Efficient mesh optimization using the gradient flow of the mean volume, SIAM J. Numer. Anal., 52 (2014), pp. 1050–1075.
[12] D. Vartziotis and J. Wipper, Fast smoothing of mixed volume meshes based on the effective geometric element transformation method, Comput. Methods Appl. Mech. Engrg., 201/204 (2012), pp. 65–81.
[13] A. E. Yilmaz and M. Kuzuoglu, A particle swarm optimization approach for hexahedral mesh smoothing, Int. J. Numer. Meth. Fluids, 60 (2009), pp. 55–78.
GEOMETRIC MESH SMOOTHING

NIKI Ltd. Digital Engineering, Research Center, 205 Ethnikis Antistasis Street, 45500 Katsika, Ioannina, Greece and, TWT GmbH Science & Innovation, Department for Mathematical Research & Services, Ernstthalenstr. 17, 70565 Stuttgart, Germany
E-mail address: dimitris.vartziotis@nikitec.gr

TWT GmbH Science & Innovation, Department for Mathematical Research & Services, Ernstthalenstr. 17, 70565 Stuttgart, Germany
E-mail address: doris.bohnet@twt-gmbh.de