615884

Application of Quaternions for Mesh Deformation*

Jamshid A. Samareh

NASA Langley Research Center, M.S. 159

Multidisciplinary Optimization Branch

Hampton, Virginia 23681 USA

j.a.samareh@larc.nasa.gov

Abstract

A new three-dimensional mesh deformation algorithm, based on quaternion algebra, is introduced. A brief overview of quaternion algebra is provided, along with some preliminary results for two-dimensional structured and unstructured viscous mesh deformation.

---

* Presented at the 8th International Conference on Numerical Grid Generation in Computational Field Simulations, June 2002.

Introduction

Mesh deformation is an important element in the analysis of moving bodies and shape optimization. The lack of robust and efficient mesh deformation tools is still a major barrier to routine applications of high-fidelity tools such as computational fluid dynamics (CFD) and computational structural mechanics (CSM) for multidisciplinary analysis and optimization. For example, CFD application for shape optimization requires a robust, automatic, and efficient tool to propagate the boundary deformation into the field mesh. For the gradient-based optimization, the efficiency is particularly crucial where in addition to the boundary deformation the sensitivity of the boundary coordinates must be propagated into the field mesh. Figure 1 shows an example of a boundary perturbation, where the boundary has been deformed, rotated, and translated.

The boundary deformation is defined as

$$\boldsymbol{\delta}_i = \mathbf{r}_i^d - \mathbf{r}_i^u \tag{1}$$

where $\mathbf{r}_i^u$ are the undeformed boundary coordinates, and $\mathbf{r}_i^d$ are the deformed boundary coordinates. There are two basic techniques to propagate the boundary perturbations into the field mesh: 1) mesh regeneration, and 2) mesh deformation. The next two subsections provide an overview of these techniques for structured and unstructured meshes.

Structured Mesh

Most structured grid regeneration and deformation techniques are based on transfinite interpolation (TFI). Gaitonde and Fiddes have provided a mesh regenerating technique based on TFI with exponential blending functions [1]. The choice of blending functions has a considerable influence on the quality and robustness of the field mesh. Soni has proposed a set of blending functions based on arclength [2]; such a set is extremely effective and robust for mesh regeneration and deformation. Jones and Samareh have presented an algorithm for general multiblock mesh regeneration and deformation based on Soni's blending functions [3].

Hartwich and Agrawal have used a variation of the TFI method [4]. They have introduced two new techniques: the use of the "slave-master" concept to semiautomate the process, and the use of a Gaussian distribution function to preserve the integrity of meshes in the presence of multiple body surfaces. Wong et al. have used Algebraic and Iterative Mesh 3D (AIM3D), which is based on a combination of algebraic and iterative methods [5]. Leatham and Chappell have used a Laplacian technique more commonly used for unstructured mesh deformation [6].

Unstructured Mesh

For unstructured meshes with large geometry changes, a new mesh may need to be regenerated at the beginning of each optimization cycle. Botkin has introduced a local remeshing procedure that operates only on the specific edges and faces associated with the design variable changes [7]. Similarly, Kodiyalam, Kumar, and Finnigan have used a mesh regeneration technique based on the assumption that the solid model topology stays fixed for small perturbations [8]. Solid model topology comprises the number of mesh-points, edges, and faces. Any change in the topology will cause the model regeneration to fail. To avoid such a failure, a set of constraints among design variables must be satisfied, in addition to constraints on their bounds.

During shape optimization, the boundary mesh may undergo many small deformations; it would be too costly to regenerate the mesh in response to these deformations. In addition, the new, regenerated mesh may not have the same number of mesh points and/or the same connectivity. Either of these situations will result in discontinuous sensitivity derivatives. Batina has presented a mesh deformation algorithm that did alleviate the need for mesh regeneration. Batina's approach models mesh edges with springs [9]. The spring stiffness k_{jk} for a given edge jk is taken to be inversely proportional to the element edge length. Then, the field mesh movement is computed through the static equilibrium equations:

$$\boldsymbol{\delta}^{n+1} = \frac{\sum_m k_{jk}\boldsymbol{\delta}^n}{\sum_m k_{jk}}, \text{ where } k_{jk} = \frac{1}{|\mathbf{r}_j - \mathbf{r}_k|} \tag{2}$$

The summation is over all the edges of the elements. The coefficient k_{jk} is relatively large for small cells. Therefore these small cells, which are usually near the boundary of the body, tend to undergo rigid body movement. This rigid body movement avoids rapid variations in deformation, thus eliminating the possibility of small cells having very large changes in volume. These large changes could lead to negative cell volumes.

Blom [10] has provided a detailed analysis for the spring method and draws an analogy between the spring method and an elliptic differential equation approach for structured mesh generation. Zhang and Belegundu have proposed an algorithm similar to the spring analogy that can handle large mesh deformation [11]. They have used the ratio of the cell Jacobian to the cell volume for the spring stiffness. Crumpton and Giles have found the spring analogy inadequate and ineffective for large mesh deformations [12] and proposed a formulation based on the heat conduction equation with the coefficient of thermal conductivity inversely proportional to cell volume. They attributed their success to the choice of cell volume used in the criteria for a valid mesh. In contrast, the

spring analogy uses only edges, which are not directly linked to the mesh validity.

Farhat et al. [13] have proposed a modification to the spring analogy algorithm to include additional torsional spring to control mesh skewness and folding. For two-dimensional applications, they demonstrated that the modified algorithm has advantages in terms of robustness, quality, and performance.

Tezduyar and Behr [14] have proposed an algorithm based on linear elasticity, which includes full stress tensor. Cavallo et al. [15] have applied this method to mesh deformation for aero/propulsive flowfield calculations. They noted that the method preserves the mesh quality, and it produces a better mesh than the spring analogy method. The linear elasticity approach requires solving the complete stress tensor. In contract, the spring analogy represents only the diagonal elements of the stress tensor. Cavallo et al. have concluded that the elasticity approach is considerably more expensive.

Role of boundary orientation in mesh deformation

The traditional deformation algorithms, such as interpolation and spring analogy, use boundary translation to deform the field mesh. However, the boundary deformation alters the boundary position as well as the boundary orientation (i.e., rotation angle) as shown in Fig. (2). The traditional mesh deformation

algorithms do not use this additional information on the changes in the boundary orientation. Morton, Melville, and Visbal [16] have proposed a TFI algorithm to interpolate the boundary deformation as well as the changes in the orientation through Euler angle. They concluded the inclusion of Euler angle preserves the mesh orthogonality for significant deformations. They successfully applied the algorithm to a two-dimensional structured CFD mesh.

Extension of the approach of Morton, Melville, and Visbal [16] to three-dimensional applications requires the direct interpolation of the changes in the orientation (three Euler angles). Historically, Euler angle representation is the most popular interpolation technique for orientation [17]. Euler angles ignore the interaction of multiple rotations about the separate axes which could lead to "gimbal lock" as described by Watt and Watt [17].

The three Euler rotations can be accomplished by a single rotation about a vector. This single rotation simplifies the interpolation process, but it has the inherent problem of non-smooth interpolation and the so-called "gimbal-lock." To avoid these problems, quaternions are used to represent the changes in the boundary orientation.

A brief introduction of quaternion algebra is presented in the next section. Then, a general three-dimensional mesh deformation algorithm based on quaternion algebra is presented.

What are quaternions?

Only brief review of quaternion algebra is provided here; readers are referred to the work of Altmann [18], Shoemake [19], and Philips, Hailey, and Gerbert [20] for more details. There is some controversy on who invented quaternion algebra. The articles by Altman [18] and Philips et al. [20] provide a very interesting history of quaternion algebra.

A quaternion is a generalized complex number (hypercomplex number) that is composed of one real and three imaginary numbers $(Q = q_0 + q_1 i + q_2 j + q_3 k)$, where $ii = jj = kk = -1$, $ij = -ji = k$, $jk = -kj = i$, $ki = -ik = j$. The following is a set quaternion properties that will be used later:

- Conjugate of a quaternion, $Q^* = q_0 - q_1 i - q_2 j - q_3 k$
- Magnitude of a quaternion, $\|Q\| = \sqrt{QQ^*} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}$
- Unit quaternion, $\|Q\| = 1$
- Associative, $(Q_1 Q_2) Q_3 = Q_1 (Q_2 Q_3)$
- Not commutative, $Q_1 Q_2 \neq Q_2 Q_1$

- Inverse of a quaternion, $Q^{-1} = Q^* / (QQ^*)$
- For unit quaternions, $Q^{-1} = Q^*$

A quaternion can be interpreted as a scalar together with a vector (direction),

$$Q = [s, \mathbf{v}], s = q_0, \mathbf{v} = (q_1, q_2, q_3)$$

In this notation, quaternion multiplication has the particularly simple form

$$Q_1 Q_2 = [s_1, \mathbf{v}_1][s_2, \mathbf{v}_2] = [s_1 s_2 - \mathbf{v}_1 \bullet \mathbf{v}_2, s_1 \mathbf{v}_2 + s_2 \mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2]$$

where $\bullet$ denotes the vector dot product, and $\times$ denotes the vector cross product.

Quaternions are ideal for modeling rotations. The last three components of a quaternion represent the axis around which the rotation occurs, and the first component represents the magnitude of the rotation. There are three steps involved in rotating a point, $\mathbf{p}$, about a unit vector, $\mathbf{u}$, by an angle, θ. First, a quaternion is constructed for the point as $P = [0, \mathbf{p}]$. Second, a quaternion is constructed for the rotation as

$$Q = [s, \mathbf{v}], s = \cos\frac{\theta}{2}, \mathbf{v} = \mathbf{u}\sin\frac{\theta}{2} \tag{3}$$

Third, the point is rotated as $P_{\text{rotated}} = QPQ^{-1}$. If Q is a unit quaternion, then we can use the conjugate of the quaternion to perform the rotation, $P_{\text{rotated}} = QPQ^*$. Multiple rotations can be simplified by using a single

quaternion. For example, if Q_1 and Q_2 are unit quaternions representing two rotations, the two rotations can be combined as

$$Q_2(Q_1 P Q_1^{-1})Q_2^{-1} = (Q_2 Q_1)P(Q_1^{-1}Q_2^{-1}) = \underbrace{(Q_2 Q_1)}\, P(\underbrace{Q_2 Q_1})^{-1}$$

Quaternion coordinates represent rotation as Cartesian coordinates represent translation as a single vector. This characteristic has been fully exploited in representing attitude of aircraft kinematics [20]. Quaternion coordinates are best for interpolation of orientation as used in computer animation. Shoemake has presented a robust and efficient application of quaternions for Bézier interpolation of orientation used in computer animation [19].

Quaternions and Mesh Deformation

This section presents a technique to model the boundary deformation by quaternion algebra. When these boundary quaternions are applied to the undeformed boundary mesh, they produce the deformed boundary mesh and orientation.

The deformation vectors, $\boldsymbol{\delta}$, represent the boundary translation, which is defined in the Euclidian space. In traditional mesh deformation algorithms, these vectors are used to propagate the deformation into the field mesh. In a similar manner, we will use the boundary quaternions to propagate the deformation into

the field mesh. The process of determining boundary quaternions is divided into three steps, as shown in Fig. (3).

In step 1, the mesh points for the undeformed boundary ($\mathbf{r}_i^u$), the deformed boundary ($\mathbf{r}_i^d$), and the neighboring points ($\mathbf{r}^u$ and $\mathbf{r}^d$) are translated to the origin.

$$\mathbf{r}^{u1} = \mathbf{r}^u - \mathbf{r}_i^u, \quad \mathbf{r}^{d1} = \mathbf{r}^d - \mathbf{r}_i^d \tag{4}$$

In the second step, $\mathbf{r}^{u1}$ is rotated so that the undeformed boundary normal vector aligns with the deformed boundary normal vector. This rotation is modeled with a quaternion. First, the normal vector of a plane shared by both deformed and undeformed normal vectors share (defined as $\mathbf{u} = \mathbf{n}_u \times \mathbf{n}_d$) and the angle α between two normal vectors is determined. Then, a quaternion is defined for the rotation as $Q_1 = [\cos \alpha/2, \mathbf{n}^u \times \mathbf{n}^d \sin \alpha/2]$. Points $\mathbf{r}^{u1}$ are rotated by quaternion to form $\mathbf{r}^{u2}$, such that

$$\mathbf{r}^{u2} = Q_1[0, \mathbf{r}^{u1}]Q_1^{-1} \tag{5}$$

In the third step, points $\mathbf{r}^{u2}$ are rotated about the deformed boundary normal vector to minimize the angle between corresponding neighboring points. The optimum rotation angle, θ, is defined as the average angle between

corresponding edges of $\mathbf{r}^{u2}$ and the edges of deformed boundary. Another quaternion can then be defined for this rotation, $Q_2 = [\cos \theta/2, \mathbf{n}^d \sin \theta/2]$.

These two quaternions are combined to form a single quaternion as $Q_i = Q_1 Q_2$. The total translation vector for the boundary can now be defined as $\Delta_i = \mathbf{r}_i^d - \mathbf{d}_i$, where $[\mathbf{d}_i, 0] = Q_i [0, \mathbf{r}_i^d] Q_i^{-1}$.

Quaternions and total translation vectors for all boundary mesh points have been computed. The translation vectors account for the translation, and quaternions account for the changes in the boundary orientation.

The translation vectors and quaternions are propagated into the field mesh by one of the traditional deformation algorithms such as TFI or the spring analogy. Then, the field mesh is updated based on the field values for the translation vectors and quaternions as

$$\mathbf{R}_{field}^d = \Delta_{field} + Q_{field} [0, \mathbf{R}_{field}^u] Q_{field}^{-1} \tag{6}$$

Results

The results are presented for structured and unstructured viscous mesh deformations. Figure (4) shows a viscous structured mesh with 257x65 mesh points. The undeformed mesh lines are orthogonal to the boundaries. The boundary mesh is deformed, rotated, and translated to simulate aeroelastic

deformation. Figure (4) shows comparisons of TFI (left side of figure) and quaternion approach (right side of figure). Unlike the traditional TFI, the quaternion approach can clearly preserve the boundary orthogonality. Because the boundary quaternions are based on the changes in the boundary mesh point positions as well as the orientations, the algorithm can guarantee that the mesh near the boundary has the same characteristics as the undeformed mesh. Figure (4) clearly demonstrates this important property.

Next, the quaternion approach is applied to an unstructured viscous mesh, where the flap has been rotated. The spring analogy was used to propagate the boundary quaternions to the field mesh. The results are shown in Fig. (5). Again, the use of quaternion has preserved the mesh characteristics.

Conclusions

A new three-dimensional mesh deformation algorithm based on quaternion algebra has been presented. These preliminary two-dimensional results indicate the traditional algorithms such as TFI and spring analogy can be easily augmented with the quaternions to preserve mesh quality near the viscous boundary. We plan to apply this method for three-dimensional structured and unstructured viscous meshes. We also plan to evaluate the quality of meshes deformed by quaternion approach by means of CFD applications.

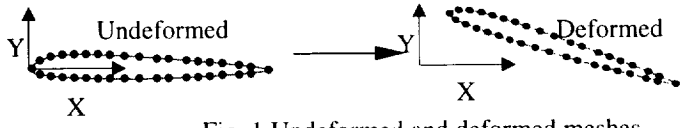

Fig. 1 Undeformed and deformed meshes

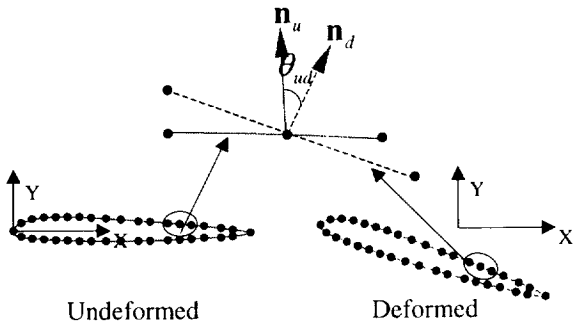

Fig. 2 Orientation changes caused by deformation

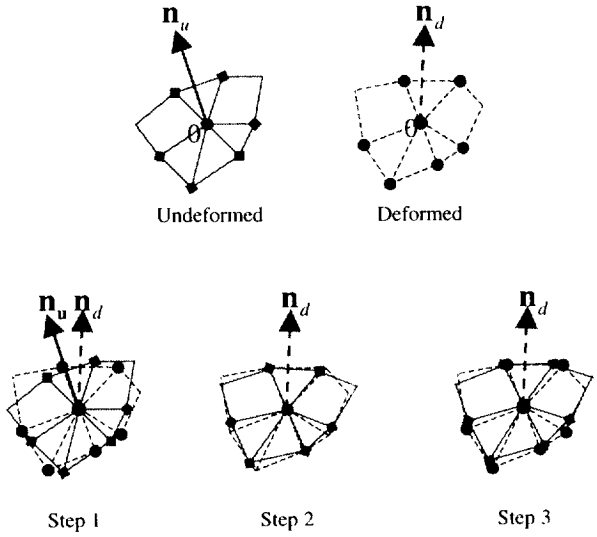

Fig. 3 Process of boundary quaternion construction

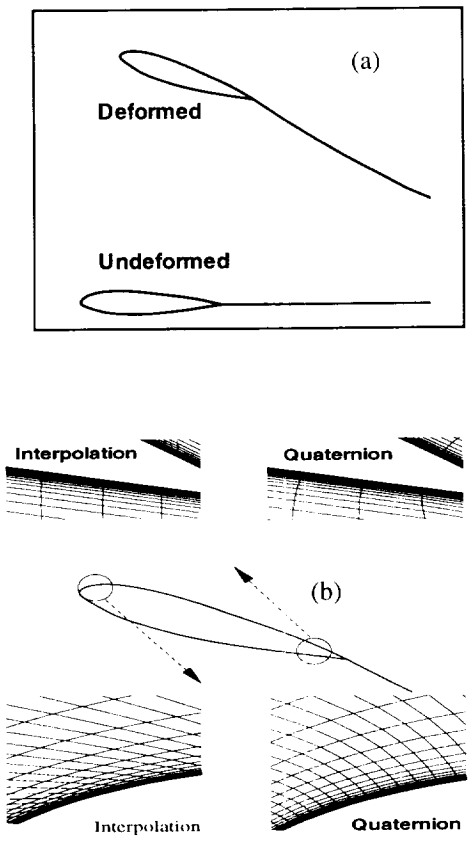

Fig. 4 Deformation comparison for structured mesh

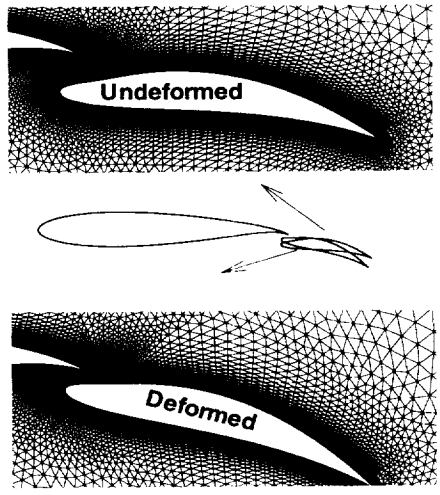

Fig. 5 Deformation comparison for unstructured mesh

[1] Gaitonde, A. L., and Fiddes, S. P., "Three-Dimensional Moving Mesh Method for the Calculation of Unsteady Transonic Flows," Aeronautical Journal, Vol. 99, No. 984, 1995, pp. 150–160.

[2] Soni, B. K., "Two- and Three-Dimensional Grid Generation Internal Flow Applications," AIAA Paper 85-1526, January 1985.

[3] Jones, W. T., and Samareh, J. A., "A Grid Generation System for Multidisciplinary Design Optimization," AIAA Paper 95-1689, June 1995. (Available online at http://techreports.larc.nasa.gov/ltrs/)

[4] Hartwich, P. M., and Agrawal, S., "Method for Perturbing Multiblock Patched Grids in Aeroelastic and Design Optimization Applications," AIAA Paper 97-2038, 1997.

[5] Wong, A. S. F., Tsai, H. M., Cai, J., Zhu, Y., Liu, F., "Unsteady Flow Calculations with a Multi-Block Moving Mesh Algorithm," AIAA-2000-1002, January 2000.

[6] Leatham, M., and Chappell, J. A., "On the Rapid Regeneration of Hybrid Grids Due to Design Driven Geometry Perturbation," Sixth International Conference on Numerical Grid Generation in Computational Field Simulation, Mississippi State University, 1998, pp. 533–542.

[7] Botkin, M. E., "Three-Dimensional Shape Optimization Using Fully Automatic Mesh Generation," AIAA Journal, Vol. 30, No. 5, 1992, pp. 1932–1934.

[8] Kodiyalam, S., Kumar, V., and Finnigan, P., "Constructive Solid Geometry Approach to Three-Dimensional Structural Shape Optimization," AIAA Journal, Vol. 30, No. 5, 1992, pp. 1408–1415.

[9] Batina, J. T., "Unsteady Euler Airfoil Solutions Using Unstructured Dynamic Meshes," AIAA Journal Vol. 28, No. 8, 1990, pp. 1381-1388.

[10] Blom, F. J., "Considerations on the Spring Analogy," International Journal for Numerical Methods in Fluids, Vol. 32, 2000, pp. 647-668.

[11] Zhang, S., and Belegundu, A. D., "A Systematic Approach for Generating Velocity Fields in Shape Optimization," Structural Optimization, Vol. 5, No. 1–2, 1993, pp. 84–94.

[12] Crumpton, P. I., and Giles, M. B., "Implicit Time-Accurate Solutions on Unstructured Dynamic Grids," International Journal for Numerical Methods in Fluids, Vol. 25, No. 11, 1997, pp. 1285–1300.

[13] Farhat, C., Degand, C., Koobus, B., Lesoinne, M., "Torsional Springs for Two-Dimensional Dynamics Unstructured Fluid Meshes," Computer Methods in Applied Mechanics and Engineering, Vol. 163, 1998, pp. 231-245.

[14] Tezduyar, T. E., Behr, M., "A New Strategy for Finite-Element Computations Involving Moving Boundaries and Interfaces—The Deforming-Spatial-Domain/Space-Time Procedure: I. The Concept and the Preliminary Numerical Tests," Computer Methods in Applied Mechanics and Engineering, Vol. 94, 1992, pp. 339-351.

[15] Cavallo, P. A., Hosangadi, A., Lee, T. A., Dash, S. M., "Dynamics Unstructured Grid Methodology with Application to Aero/Propulsive Flowfields," AIAA Paper 97-2310.

[16] Morton, S. A., Melville, R. B., Visbal, M. R., "Accuracy and Coupling Issues of Aeroelastic Navier-Stokes Solutions on Deforming Meshes," Journal of Aircraft, Vol. 35, No. 5, September-November 1998, pp. 798-805.

[17] Watt, A., Watt, M., "Advanced Animation and Rendering Techniques: Theory and Practice," Addison-Wesley Publishing Company, New York, 1992, Chap. 17.

[18] Altmann, S. L., "Rotations, Quaternions, and Double Groups," Clarendon Press, Oxford, 1986.

[19] Shoemake, K., "Animating Rotation with Quaternion Curves," SIGGRAPH 1985 Conference Proceedings, July 1985, pp. 245-254.

[20] Philips, W. F., Hailey, C. E., Gerbert, G. A., "Review of Attitude Representation Used for Air Kinematics," Journal Aircraft, Vol. 38, No. 4, July-August 2001, pp.718-737.