A Robust Unstructured Grid Movement Strategy using Three-Dimensional Torsional Springs

Clarence O. E. Burg[*]

Mississippi State University, Mississippi State, Mississippi, 39762, U.S.A.

The ability to move a grid during a simulation is of critical importance to many types of simulations, including aero-elasticity simulations for wings, gradient-based design optimization, dynamic movement of control surfaces for maneuvering, and surface tracking methods for free surface flows. For structured grids, transfinite interpolation is a robust method for propagating deformations on the surfaces into the volume grid, and its two-dimensional version is readily extendible to three-dimensions. For unstructured grids, two methods that have been used within two-dimensional grids are the use of linear springs and torsional springs to propagate the changes on the surface into the volume mesh. The use of linear springs provides an efficient method to move unstructured grids, but it is not very robust, in that moderate to large deformations will result in invalid meshes, where some of the elements have been inverted. Farhat developed the torsional spring method for two-dimensional grids, and has demonstrated that this method is much more robust, even for large deformations. Farhat and Murayama have attempted to extend the two-dimensional torsional springs method to three-dimensions, and their results are impressive. However, both of their methods reduces the three-dimensional tetrahedra into simpler components without analyzing the dynamics of the three-dimensional object, and as such, their methods do not fully represent a three-dimensional extension of the two-dimensional torsional springs method. In this paper, a three-dimensional extension of the torsional springs method is derived by analyzing the equations of volume and area for a tetrahedron and its faces and edges.

Nomenclature

k_{ij}	Linear Spring Stiffness
l_{ij}	Length of Edge Between Nodes i and j
$\Delta x_i, \Delta y_i$	Displacement at Node i
θ_i^{ijk}	Angle at Node i in Triangle ijk
C_{ijk}	Torsional Spring Stiffness
q^{ijk}	Displacements at Nodes i, j and k
R^{ijk}	Rotation Matrix for Triangle ijk
ξ	Any Spatial Dependency
$Area_{\triangle_{ijk}}$	Area of Triangle ijk
M^{ijk}	Moment at corner i in Triangle ijk
F^{ijk}	Force at corner i in Triangle ijk
T^{ijk}	Transformation Matrix at corner i in Triangle ijk
C_{linear}^{ij}	Linear Spring Stiffness in 3D
φ	Angle at Corner i in Volume $ijkl$
V^{ijkl}	Volume Spanned by Nodes $ijkl$
q_f	Fixed Displacements
q_m	Displacements for Movable Nodes
K_{ff}, K_{mm}	Square Matrices for Total Stiffnesses for Fixed and Moveable Nodes

[*]Assistant Research Professor, Computational Simulation and Design Center, 2 Research Blvd., Starkville, MS, 39759, Member AIAA.

I. Introduction

Moving computational grids are used within several areas of computational modeling including geometric design optimization, fluid-structure interaction and separation of rocket boosters or stores. For geometric design optimization, the shape of the object being optimized is changed, in order to improve the characteristics of the resulting flow, which requires flow evaluations on grids that conform to the new shape. An example of fluid-structure interaction is the change in shape and orientation of a wing under the stresses of flight, and a special case of this fluid-structure interaction is the induced flutter of the wing. Separation of objects in a computational grid include the separation of weapons ordinance or stores from an aircraft and the separation of booster rockets from a multi-stage rocket. In each of these cases, the computational grid changes due to changes in the boundary of the grid.

However, the grid movement requirements of various types of simulations differ. For instance, the simulation of two objects moving relative to one another but at a reasonable distance from each other, such as one ship passing another, requires the grid at a distance from the ships to move but not the grid near a ship. The simulation of a maneuvering missile that responds to a change in its control surface will require that the grid in a region near the control surface move while the grid attached to the control surface may remain fixed. But for the simulation of the water surface around a ship in motion requires that the grid attached to the ship move while conforming to the shape of the ship. And for geometric design optimization, or for simulations of vibrating surfaces, the grid attached to the surface must move with the surface. Several grid movement algorithms will work satisfactorily for objects that are translated or rotated within a grid, but many of these algorithms fail for motions that have higher frequency components such as exist for the latter cases.

Complicating many simulations, modeling viscous effects correctly is often crucial to capturing the flow features properly. Thus, near the boundary of the viscous objects, the grid is quite tightly packed. For boundary layer grids, the spacing off the wall can easily to be 10^{-6} times the size of the body. Thus, almost any changes in the location of the boundary will cause the boundary to pass many layers of nodes on the interior of the computational grid, unless these nodes move a distance similar to the change at the boundary.

As unstructured grid technology has improved, its use within computational modeling has increased, due to the ease of fitting an unstructured grid around almost any irregularly shaped object. Furthermore, the ability to adapt unstructured grids near regions of computational interest have added to its popularity. With the additional ability to generate highly stretched grids in the boundary layer, unstructured grids can be used within computational fluid dynamics to simulate the flow for a wide range of applications.

However, developing a robust grid movement scheme for unstructured grids, that does not produce negative volumes, that maintains the same nodal connectivity and that is computational efficient is an active area of research. Several different methods have been proposed, studied and implemented for moving unstructured grids including the linear spring analogy and the torsional spring analogy.

Batina[1] proposed the linear spring analogy, where a fictitious spring replaces each edge in the unstructured grid and the stiffness of the spring grows as the length of the edge decreases. This method has been widely used, due primarily to its ease of implementation and computational efficiency; however, this method quite frequently produces grids with negative volume elements. Several efforts have been made to improve the linear spring analogy, including those of Anderson, of Singh and of Murayama. Anderson[2] used the linear spring analogy and local reconnection of the grid wherever the deformed grid had elements with negative volumes. The resulting grids were valid but had degraded quality even after the local reconnection. Singh[3] and others have applied the linear spring analogy to problems of translating and rotating objects where only nodes outside a certain distance from the object in question are allowed to rotate while rigidly moving the points attached to and near the object in motion. For the class of problems that they were simulating, this approach worked well. Murayama[4] combined some of the aspects of the torsional spring method within the linear spring framework, where the stiffness associated with each edge included the linear spring stiffness and the sum of the torsional springs associated with the node. Thus, the governing equations were less cumbersome than the full torsional spring method, and they have shown some excellent results.

The third method, which was developed by Farhat[5] for two-dimensional unstructured grids, is similar to the linear spring analogy, but includes a torsional spring between each angle in the computational grid. This torsional spring method is guaranteed to generate grids with no negative volume elements. More recently, Farhat[6] has extended his method for three-dimensional meshes by slicing each element with two-dimensional planes and calculating the torsional spring contributions for each two-dimensional slice.

In the next section, both the linear spring and torsion spring methods are presented for two-dimensional unstructured grids, and several implementation issues are discussed. The two-dimensional torsional spring method is formulated differently from Farhat's presentation, because this formulation is clearly extendible to three-dimensional grids. The following section introduces the three-dimensional torsional spring method and discusses the application of this method to both tetrahedral and mixed element grids. In the final sections, two benchmark cases and several examples of three-dimensional grid deformation involving the solution of the free surface equation for notational ship hulls are presented.

II. Linear And Torsional Springs for Two-Dimensional Grids

A. Linear Spring Formulation

Batina[1] proposed a grid movement method for unstructured grids where each edge in the computational grid is replaced by a linear spring whose stiffness is inversely proportional to the length of the edge. Thus, for the edge connecting nodes i and j, the stiffness k_{ij} of the spring is

$$k_{ij} = \frac{1}{((x_i - x_j)^2 + (y_i - y_j)^2)^{p/2}} = \frac{1}{l_{ij}^p} \tag{1}$$

where (x_i, y_i) and (x_j, y_j) are the coordinates of nodes i and j and p is a predetermined coefficient, usually 1 or 2 and $l_{ij} = \sqrt{((x_i - x_j)^2 + (y_i - y_j)^2)}$. Given a set of nodal displacements or forces acting on the boundary of the computational grid, the following equations for the interior displacements are solved iteratively until all the forces are in equilibrium

$$\begin{aligned} \Delta x_j^{n+1} &= \frac{\sum_j k_{ij} \Delta x_j^n}{\sum_j k_{ij}} \\ \Delta y_j^{n+1} &= \frac{\sum_j k_{ij} \Delta y_j^n}{\sum_j k_{ij}} \end{aligned} \tag{2}$$

where j is summed over all edges connected to node i. This method is readily extendible to three-dimensional grids and is computationally efficient requiring only a few Jacobi iterations to achieve an acceptable level of accuracy.

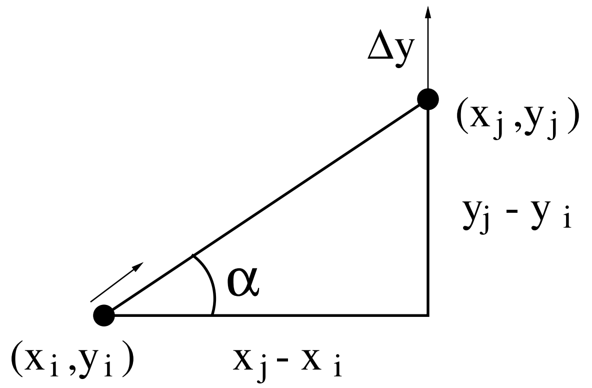

Figure 1. A Displacement in y-direction results in displacements in x and y directions.

Technically speaking, these equations do not simulate the behavior of a network of springs because there is no interaction between the x and y coordinates. A displacement in one coordinate will not influence the location in the other coordinate, as would be the case for a network of springs. An example of this coupled interplay is shown in Figure 1, where a displacement in the y-direction is applied to node j which results in a displacement in both directions for node i. To simulate the behavior of a network of springs, the following set of equations for each spring must be summed over all springs:

Forces in x-direction at node i:

$$k_{ij} \left[(\Delta x_i - \Delta x_j) \cos^2 \alpha + (\Delta y_i - \Delta y_j) \cos \alpha \sin \alpha \right] \tag{3}$$

Forces in y-direction at node i:

$$k_{ij} \left[(\Delta x_i - \Delta x_j) \cos \alpha \sin \alpha + (\Delta y_i - \Delta y_j) \sin^2 \alpha \right] \tag{4}$$

For either representation, however, this method often fails for complicated geometries and for large changes in the boundary, because negative areas are generated when nodes cross-over edges in the grid. The creation of negative areas is illustrated in Figure 2, where the top node is pushed downwards and the node in the middle is forced to pass the edge between the bottom two nodes.

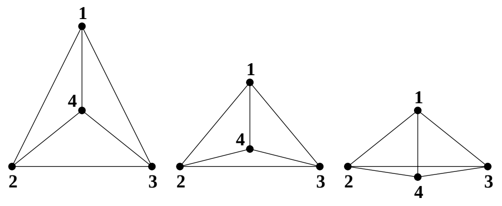

Figure 2. Negative Areas Produced by Linear Spring Method.

The reason for this failure is that the stiffness in the linear spring method prevents two nodes from colliding but does not prevent a node from crossing over an edge. As two nodes get closer together, the stiffness increases without bound preventing the collision; but there is no mechanism to prevent a node from crossing an edge.

B. Torsion Spring Formulation in 2D

To provide a more robust unstructured grid movement method, Farhat provided a mechanism to prevent a node from crossing an edge by using torsional springs around each node, as shown in Figure 3.

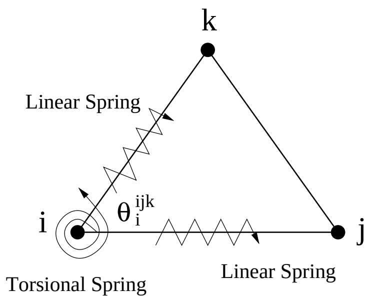

Figure 3. Placement of Linear and Torsion Springs.

The stiffness of the torsion spring is inversely proportional to the sine of the angle, so that the stiffness grows without bound as the angle decreases towards zero or increases towards 180^o, or

$$C_{ijk} = \frac{1}{\sin^2 \theta_i^{ijk}} \tag{5}$$

Thus, as a node moves towards an edge, the angle goes towards zero, and the stiffness of the torsion spring grows. The sine of the angle is squared to prevent a negative stiffness.

For the torsion spring method, the angle θ_i^{ijk} changes as the location of the three nodes in the angle change. The change in this angle $\Delta\theta_i^{ijk}$ with respect to changes in the locations of these nodes is related via

$$\Delta\theta_i^{ijk} = R^{ijk^T} q^{ijk} \tag{6}$$

where q^{ijk} is the displacements of the three nodes i, j, and k and R^{ijk} is rotation matrix. These matrices are actually vectors of the form

$$q^{ijk} = \begin{bmatrix} \Delta x_i \\ \Delta y_i \\ \Delta x_j \\ \Delta y_j \\ \Delta x_k \\ \Delta y_k \end{bmatrix}, R^{ijk} = \begin{bmatrix} \frac{\partial\theta_i}{\partial x_i} \\ \frac{\partial\theta_i}{\partial y_i} \\ \frac{\partial\theta_i}{\partial x_j} \\ \frac{\partial\theta_i}{\partial y_j} \\ \frac{\partial\theta_i}{\partial x_k} \\ \frac{\partial\theta_i}{\partial y_k} \end{bmatrix} \tag{7}$$

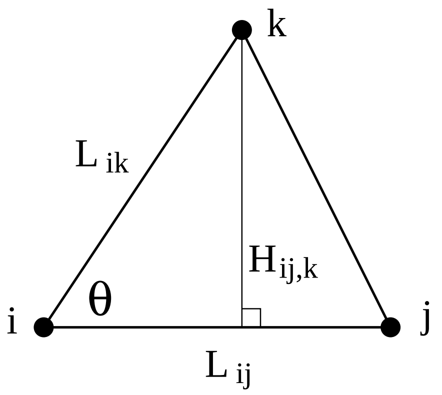

Figure 4. Notation for a typical angle in a Triangle

The area of a triangle, as shown in Figure 4, is

$$Area_{\triangle_{ijk}} = \frac{1}{2} L_{ij} H_{ij,k} \tag{8}$$

But the height of this triangle $H_{ij,k}$ is $\sin(\theta_i^{ijk}) L_{ik}$. Thus, the sine of the angle can be expressed as

$$\sin(\theta_i^{ijk}) = \frac{2 Area_{\triangle_{ijk}}}{L_{ij} L_{ik}} \tag{9}$$

The derivative of $\sin(\theta_i^{ijk})$ with respect to any term in q^{ijk} can be expressed via the chain rule as

$$\frac{\partial \sin(\theta)}{\partial \xi} = \frac{\partial \sin(\theta)}{\partial \theta} \frac{\partial \theta}{\partial \xi} = \cos(\theta) \frac{\partial \theta}{\partial \xi} \tag{10}$$

However, $\sin(\theta)$ is defined above, so that this derivative is

$$\frac{\partial \sin(\theta)}{\partial \xi} = \sin(\theta) \left(\frac{1}{Area_{\triangle_{ijk}}} \frac{\partial Area_{\triangle_{ijk}}}{\partial \xi} - \frac{1}{L_{ij}} \frac{\partial L_{ij}}{\partial \xi} - \frac{1}{L_{ik}} \frac{\partial L_{ik}}{\partial \xi} \right) \tag{11}$$

Thus, the entries in the rotation matrix R^{ijk} can be expressed as

$$\frac{\partial \theta}{\partial \xi} = \frac{\sin(\theta)}{\cos(\theta)} \left(\frac{1}{Area_{\triangle_{ijk}}} \frac{\partial Area_{\triangle_{ijk}}}{\partial \xi} - \frac{1}{L_{ij}} \frac{\partial L_{ij}}{\partial \xi} - \frac{1}{L_{ik}} \frac{\partial L_{ik}}{\partial \xi} \right) \tag{12}$$

This formulation reduces exactly to the formulation presented by Farhat but, as will be shown, more easily extends to three dimensions.

C. Linear Spring Formulation in 2D

Following the same formulation for the torsion spring, the change in the length of edge l_{ij} with respect to the displacements at the two endpoints can be expressed via a linear transformation

$$\Delta l_{ij} = R_{linear}^{ij} q_{ij} \tag{13}$$

where q_{ij} is the displacement at the two endpoints and R_{linear}^{ij} is the rotation matrix associated with the linear spring. Referring to Figure 1, the length of the edge is

$$l_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \tag{14}$$

and the change in the length with respect to a change in x_i is

$$\frac{\partial l_{ij}}{\partial x_i} = \frac{-(x_j - x_i)}{l_{ij}} = -\cos\alpha \tag{15}$$

Thus, the rotation matrix can be expressed as

$$R^{ij}_{linear} = \begin{bmatrix} -\cos\alpha \\ -\sin\alpha \\ \cos\alpha \\ \sin\alpha \end{bmatrix} \tag{16}$$

The stiffness matrix is simply $C^{ij}_{linear} = \frac{1}{l^2_{ij}}$. Converting these equations into forces, we get

$$\begin{aligned} F^{ij}_{linear} &= \left[{R^{ij}_{linear}}^T C^{ij}_{linear} R^{ij}_{linear} \right] q^{ij} \\ &= K^{ij}_{linear} q^{ij} \end{aligned} \tag{17}$$

where

$$K^{ij}_{linear} = \frac{1}{l^2_{ij}} \begin{bmatrix} \cos^2\alpha & \cos\alpha\sin\alpha & -\cos^2\alpha & -\cos\alpha\sin\alpha \\ \cos\alpha\sin\alpha & \sin^2\alpha & -\cos\alpha\sin\alpha & -\sin^2\alpha \\ -\cos^2\alpha & -\cos\alpha\sin\alpha & \cos^2\alpha & \cos\alpha\sin\alpha \\ -\cos\alpha\sin\alpha & -\sin^2\alpha & \cos\alpha\sin\alpha & \sin^2\alpha \end{bmatrix} \tag{18}$$

Expanding this matrix, the forces given in equations (3) and (4) are obtained.

D. Equilibrium of Forces

Following the derivation of Farhat, the moment M^{ijk} produced by the change in the angle θ can be expressed by the scalar equation

$$M^{ijk} = C^{ijk} \Delta\theta^{ijk} \tag{19}$$

where C^{ijk} is the stiffness of the spring. These moments can be converted into forces F^{ijk} for which the equilibrium conditions must be determined by a linear transformation matrix T^{ijk} determined from

$$F^{ijk} = T^{ijk} M^{ijk} \tag{20}$$

and equating the work done by the forces with the work done by the moments, or

$${F^{ijk}}^T q^{ijk} = {M^{ijk}}^T \Delta\theta^{ijk} \tag{21}$$

Plugging in equation (6) and using equation (20),

$$\left(T^{ijk} M^{ijk}\right)^T q^{ijk} = M^{ijk} {R^{ijk}}^T q^{ijk} \tag{22}$$

or

$$M^{ijk} {T^{ijk}}^T q^{ijk} = M^{ijk} {R^{ijk}}^T q^{ijk} \tag{23}$$

Thus, the transformation matrix is

$$T^{ijk} = R^{ijk} \tag{24}$$

and the equation for the forces is

$$\begin{aligned} F^{ijk} &= \left[R^{ijk} C^{ijk} {R^{ijk}}^T \right] q^{ijk} \\ &= K^{ijk} q^{ijk} \end{aligned} \tag{25}$$

This derivation is valid for any angle in the two-dimensional mesh whether it is part of triangle or a higher order element such as a quadrilateral; Farhat's derivation assumes that each element is a triangle and thus considers the three angles simultaneously. Thus, in the derivation presented herein, R^{ijk} is a $[6 \times 1]$ matrix and C^{ijk} is a $[1 \times 1]$ matrix; whereas in Farhat's derivation R^{ijk} is a $[6 \times 3]$ matrix and C^{ijk} is a $[3 \times 3]$ matrix.

III. Linear and Torsional Springs for Three-Dimensional Grids

A. Linear Spring Formulation in 3D

As was the case for two-dimensional linear springs, three-dimensional linear spring analogy replaces each edge with a spring whose stiffness is inversely proportional to the length of the edge, raised to some power p. To derive the rotation matrix, consider the edge e_{ij} between points i and j in Figure 5.

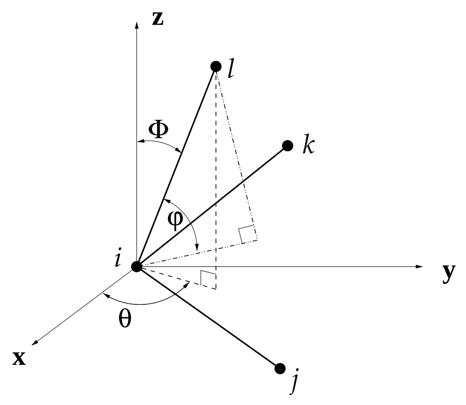

Figure 5. Diagram of Edge and Corner in 3D.

The length of this edge is

$$l_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2} \tag{26}$$

and the change in the length based on a change in x_i is

$$\frac{\partial l_{ij}}{\partial x_i} = \frac{-(x_j - x_i)}{l_{ij}} = -\cos\theta\sin\phi \tag{27}$$

The other derivatives in the rotation matrix are similar to this one. Thus, the relationship between the forces acting on an edge and change in location of the endpoints can be expressed as

$$F^{ij}_{linear} = K^{ij}_{linear} q^{ij} = \left[{R^{ij}_{linear}}^T C^{ij}_{linear} R^{ij}_{linear} \right] q^{ij} \tag{28}$$

where

$$R^{ij}_{linear} = \begin{bmatrix} -cos\theta sin\phi \\ -sin\theta sin\phi \\ -cos\phi \\ cos\theta sin\phi \\ sin\theta sin\phi \\ cos\phi \end{bmatrix} \tag{29}$$

and $C^{ij}_{linear} = \frac{1}{l^2_{ij}}$.

B. Torsional Springs in 3D

In 3D, the torsion spring is applied to each corner of an element, whether it is a tetrahedron or a higher order element. The corner depicted in Figure 5 is labeled c^{ijkl}_i and consists of 4 nodes $\vec{x}_i$, $\vec{x}_j$, $\vec{x}_k$ and $\vec{x}_l$. In this figure, nodes $\vec{x}_j$ and $\vec{x_k}$ appear to lie in the xy-plane, which is not necessarily true. Constructing 3 vectors $\vec{v}_{ij}$, $\vec{v}_{ik}$, $\vec{v}_{il}$, originating at node $\vec{x}_i$ and passing through the other three nodes, the volume of the tetrahedron V^{ijkl} formed by these four nodes can be expressed as

$$\begin{aligned} V^{ijkl} &= \frac{1}{6}\vec{v}_{il} \cdot (\vec{v}_{ij} \times \vec{v}_{ik}) \\ &= \frac{1}{3}A^{ijk}H^l_{ijk} \\ &= \frac{1}{3}A^{ijk} l_{il} \sin\varphi \end{aligned} \tag{30}$$

where A^{ijk} is the area of the triangular face associated with nodes i, j and k. Thus, the equation for $\sin\varphi$ can be expressed as

$$\sin\varphi = \frac{3V^{ijkl}}{A^{ijk} l_{il}} \tag{31}$$

and the equation for $\cos\varphi$ is

$$\cos\varphi = \frac{\sqrt{{A^{ijk}}^2 {l_{il}}^2 - 9{V^{ijkl}}^2}}{A^{ijk} l_{il}} \tag{32}$$

The derivative of $\sin\varphi$ with respect to a variable ξ is then

$$\frac{\partial \sin\varphi}{\partial \xi} = \cos\varphi \frac{\partial\varphi}{\partial\xi} \tag{33}$$

and

$$\begin{aligned}\frac{\partial \sin\varphi}{\partial \xi} = \frac{\partial}{\partial\xi}\left(\frac{3V^{ijkl}}{A^{ijk} l_{il}}\right) &= \frac{3}{A^{ijk} l_{il}}\frac{\partial V^{ijkl}}{\partial\xi} - \frac{3V^{ijkl}}{{A^{ijk}}^2 l_{il}}\frac{\partial A^{ijk}}{\partial\xi} - \frac{3V^{ijkl}}{A^{ijk} {l_{il}}^2}\frac{\partial l^{il}}{\partial\xi} \\ &= \frac{3V^{ijkl}}{A^{ijk} l_{il}}\left(\frac{1}{V^{ijkl}}\frac{\partial V^{ijkl}}{\partial\xi} - \frac{1}{A^{ijk}}\frac{\partial A^{ijk}}{\partial\xi} - \frac{1}{l_{il}}\frac{\partial l^{il}}{\partial\xi}\right) \\ &= \sin\varphi\left(\frac{1}{V^{ijkl}}\frac{\partial V^{ijkl}}{\partial\xi} - \frac{1}{A^{ijk}}\frac{\partial A^{ijk}}{\partial\xi} - \frac{1}{l_{il}}\frac{\partial l^{il}}{\partial\xi}\right)\end{aligned} \tag{34}$$

so

$$\frac{\partial\varphi}{\partial\xi} = \frac{sin\varphi}{cos\varphi}\left(\frac{1}{V^{ijkl}}\frac{\partial V^{ijkl}}{\partial\xi} - \frac{1}{A^{ijk}}\frac{\partial A^{ijk}}{\partial\xi} - \frac{1}{l_{il}}\frac{\partial l^{il}}{\partial\xi}\right) \tag{35}$$

The variation of φ with respect to x, y and z for the four nodes that make up the corner are needed. This variation forms the vector $R^{ijkl}_{torsion}$, and the stiffness matrix $K^{ijkl}_{torsion}$ is formed from the vector $R^{ijkl}_{torstion}$ along with the stiffness coefficient $C^{ijkl}_{torsion} = \frac{1}{sin^2\varphi}$. Each corner in a tetrahedra consists of three different angles as determined from the choice of the face and the opposite edge, so that the contribution from each angle in each corner must be considered. Furthermore, for higher order elements such as prisms and pyramids, each corner of these elements must be considered individually.

C. Solution Method

For both the two-dimensional and the three-dimensional cases, after summing up the forces at each node, the force equation to be solved has the form

$$F = Kq = 0 \tag{36}$$

where the displacements at certain boundary locations are fixed, or

$$q = \bar{q} \text{ on } \Gamma \tag{37}$$

After reindexing the nodes in the grid so that the fixed nodes are located at the front of the list, the displacement vector q can be expressed as

$$q = \begin{bmatrix} q_f \\ q_m \end{bmatrix} \tag{38}$$

where q_f are the fixed displacements and q_m are the displacements for the movable nodes. Similarly the matrix K can be expressed as

$$K = \begin{bmatrix} K_{ff} & K_{fm} \\ K_{mf} & K_{mm} \end{bmatrix} \tag{39}$$

where K_{ff} and K_{mm} are square matrices and $K_{fm} = K^T_{fm}$. As a result, the equation to be solved is

$$K_{mm} q_m = -K_{mf} q_f \tag{40}$$

These equations are solved using a Gauss-Seidel iterative method.

IV. Examples

The following examples show the compression of a benchmark problem of a set of tetrahedra within a larger tetrahedra, as an extension of the two-dimensional benchmark presented by Farhat, the translation of a cube within a cube, similar to that performed by Murayama, and three different simulations of the free surface flow problem, requiring the motion of the grid. These free surface flow problems include the flow around a NACA 0012 hydrofoil, around a simple hull form called the Series 60 $C_b = 0.6$ hull and around a more realistic hullform, representative of naval destroyers.

A. Benchmark Cases

The first example is a three-dimensional benchmark case which is an extension of the two-dimensional benchmark case used by Farhat[5]. The undeformed mesh consists of 9 nodes and 18 tetrahedra and is shown in Figure 6 and 7. The node at the top of the mesh moves downward compressing the tetrahedra. Figure 6 shows the view from one of the three sides of the mesh, and Figure 7 shows the mesh from the top.

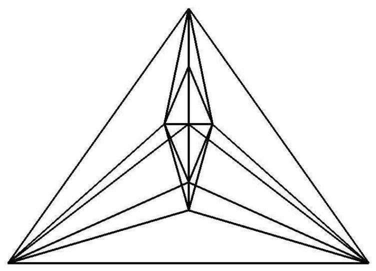

Figure 6. Benchmark Case from the Side.

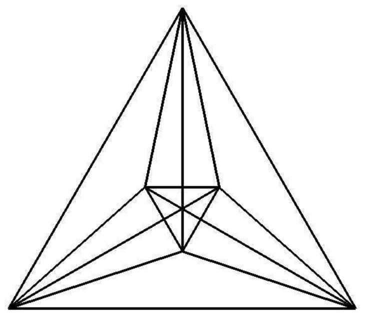

Figure 7. Benchmark Case from the Top.

The top node is forced downward compressing nodes. The entire mesh remains valid when using the torsional springs as is shown in Figure 8, 9 and 10, where the mesh is compressed to 25%, 50% and 75% of its original size. The linear spring analogy fails when the grid is compressed by more than 30%, at which point the lowest node in the interior of the mesh passes through the bottom face of the mesh.

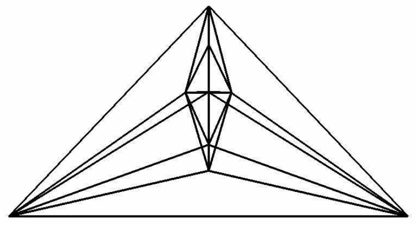

Figure 8. Benchmark Case Compressed by 25%.

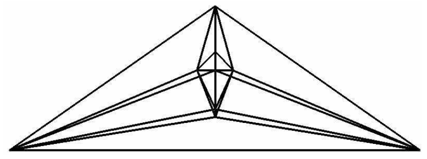

Figure 9. Benchmark Case Compressed by 50%.

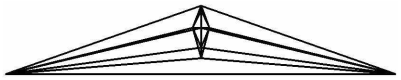

Figure 10. Benchmark Case Compressed by 75%.

In Figures 11 and 12, the original and deformed grid for a cube of length 1 within a cube of length 3 are shown. The inner cube is pushed upwards by a distance of 0.6. The resultant mesh is still valid (i.e., the volumes of each element are positive), without the need to reconnect the nodes or regenerate the grid.

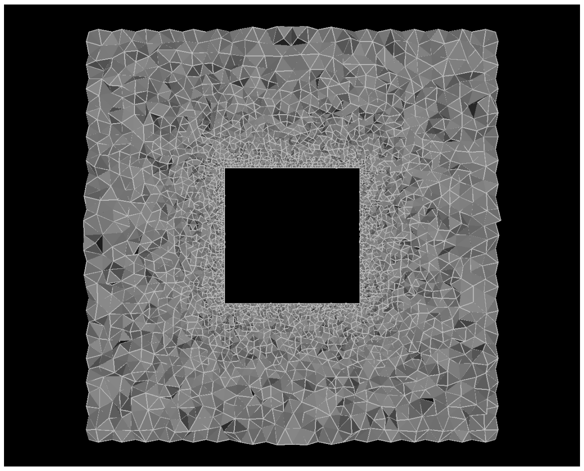

Figure 11. Original Cube within a Cube.

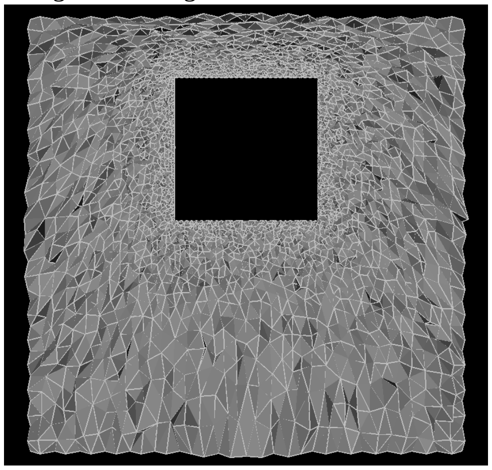

Figure 12. Deformed Cube within a Cube.

B. Applications to Free Surface Flows

The first application deals with the flow of water past a two-dimensional submerged hydrofoil, which creates water waves. In order to simulate the flow of water, the interface between the water and the air must be allowed to move. For surface tracking methods, the grid must also be moved to match the free surface. The flow past the NACA 0012 hydrofoil has been calculated experimentally, and several researchers have run simulations for this geometry to validate their free surface flow solvers. Hence, this case is an excellent choice as an initial test case for free surface simulations. Burg[7] provides a full discussion of the free surface tracking algorithm, which uses the torsional spring method to move the mesh.

Shown in Figures 13 and 14 are the original and deformed unstructured grids around the two-dimensional hydrofoil. The quality of the deformed grid is roughly the same as the original grid, despite the deformation. This simulation was performed on a two-dimensional grid that was extruded into three-dimensions, and the grid movement was performed on the three-dimensional grid. As can be seen from the figures, the deformed grid has similar grid quality as the original grid.

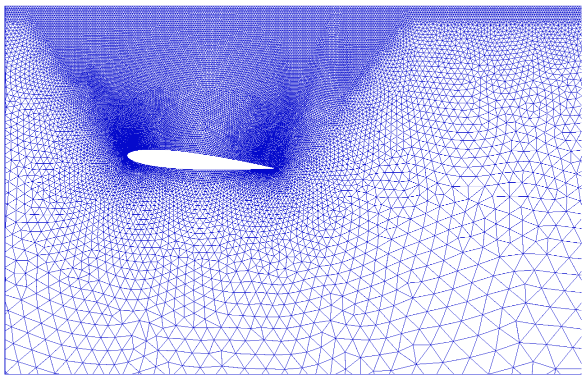
Figure 13. Original unstructured grid about NACA 0012 hydrofoil.

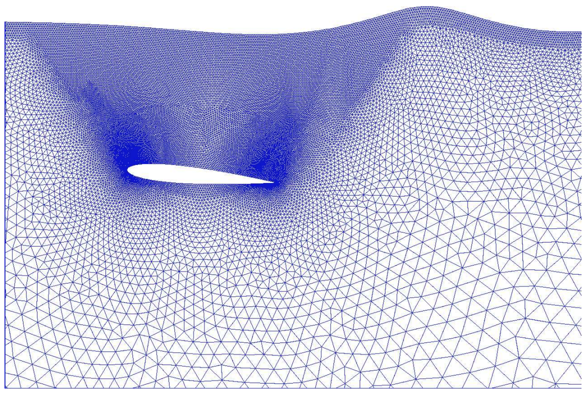
Figure 14. Deformed unstructured grid about NACA 0012 hydrofoil.

The second test case deals with the Series 60 $C_b = 0.6$ hull form. This hull form is often used by researchers as a validation case since the geometry is relatively simple and the flow features are not complicated. There is also a significant amount of experimental data available for this hull form, for comparison purposes. The main reason that this case is shown within this paper is that the grid as the stern of the ship is extremely distorted, highlighting the ability of the grid movement algorithm to handle complicated mesh deformations.

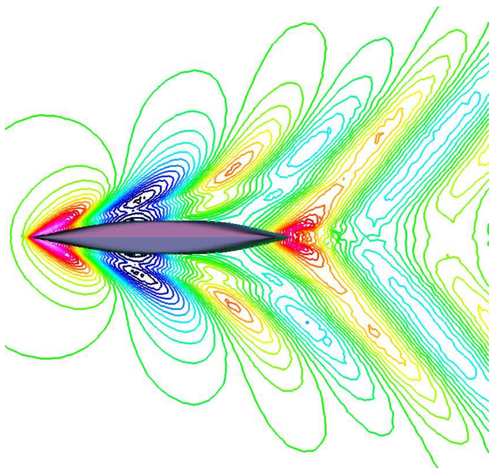
Figure 15. Contours of Free Surface Elevations for Series 60 Hull form.

In Figure 15, the free surface contours are shown for the Series 60 hull form, where the red and white values are the highest elevations. The flow is from left to right, and at the stern of the ship, the flow rises greatly, as it does at the bow. At the bow of the ship, the geometry of the ship rises vertically and does not present much difficulty for the grid movement algorithm. At the stern of the Series 60, the geometry leaves the water at a sharp angle, so that if the water rises by 1 unit vertically, a node of the surface of the ship moves by approximately 2 units horizontally. Thus, the grid is greatly stretched in the stern region. The elements along the centerline for the original grid and for the deformed grid are shown in Figures 16 and 17. In Figure 17, the grid is clearly stretched upward and to the right, but the grid is still a valid grid with no negative volumes.

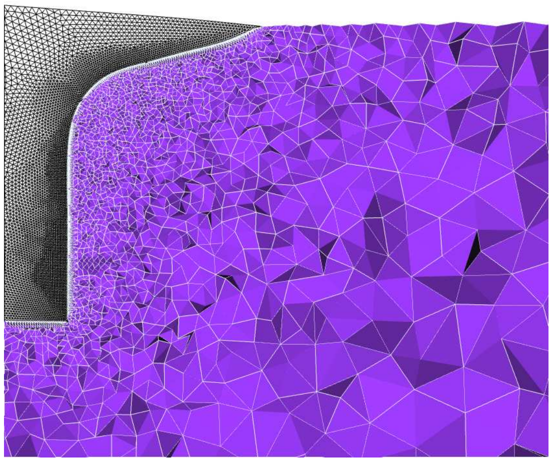
Figure 16. Original Series 60 Grid.

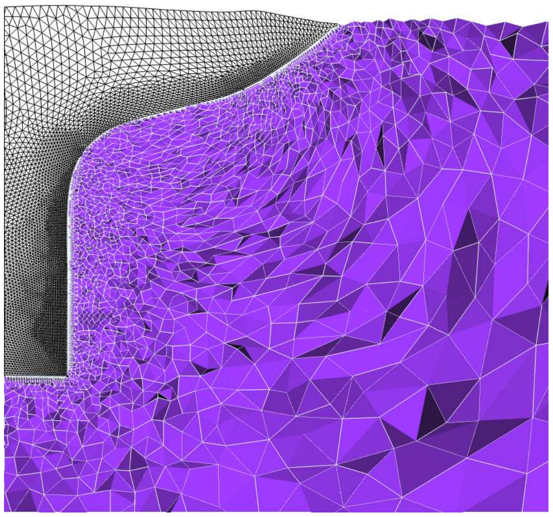
Figure 17. Deformed Series 60 Grid.

The final example deals with the David Taylor Model Basin Model 5415 hull form, which is a prototypical hullform for naval destroyer class ships. It has many of the features of actual ships such as a protruding bow, a bulbous bow sonar dome, a tapered stern region where the propellers, shafts, struts and rudder would be placed, and a transom stern. The transom stern is of particular importance for the free surface calculations. The flow is the stern region is highly complicated and is strongly dependent on the Reynolds number, which affects the free surface and hence the forces acting on the vessel. The flow in the stern makes gridding the region complicated as well as simulating the flow with a free surface flow solver where the grid must be moved to match the free surface.

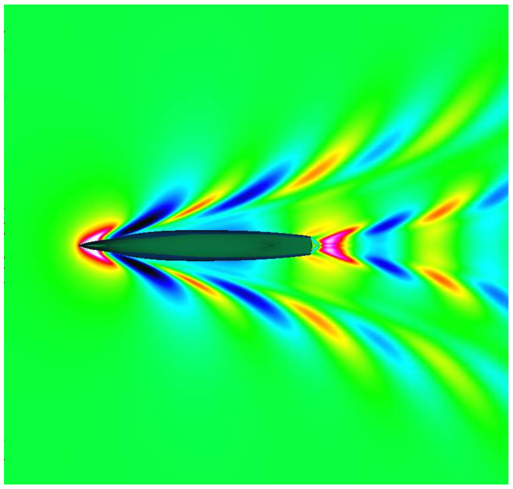
Figure 18. Free Surface Elevations for the DTMB Model 5415.

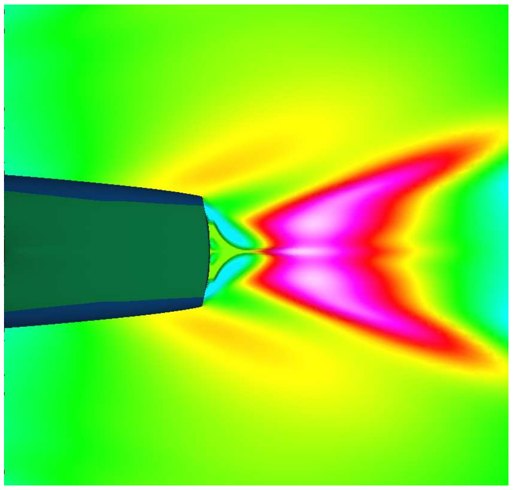
Figure 19. Top View of Free Surface Elevations in Stern Region.

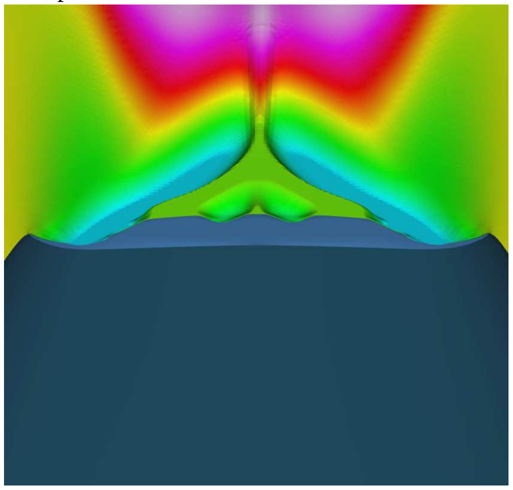
Figure 20. View of Free Surface Elevations from Stern of Ship.

The free surface for the DTMB Model 5415 is shown in Figures 18, 19 and 20, showing the overview of the flow patterns as the flow travels from left to right. Again, the higher elevations are in red and white while the lower elevations are in blue and black. The familiar Kelvin wave pattern originating at the bow can be seen, and the "rooster tail" at the stern is also apparent. The close-up in the stern region in Figure 19 shows the complicated free surface structures in the stern region, and in Figure 20, the three-dimensional effects of the free surface can be seen in the stern looking away from the ship, where the ship's hull is shown in gray. In particular, the flow drops dramatically behind the corner of the stern and the port and starboard sides, and then rises again at the centerline. This flow behavior greatly changes the shape of the grid in this region.

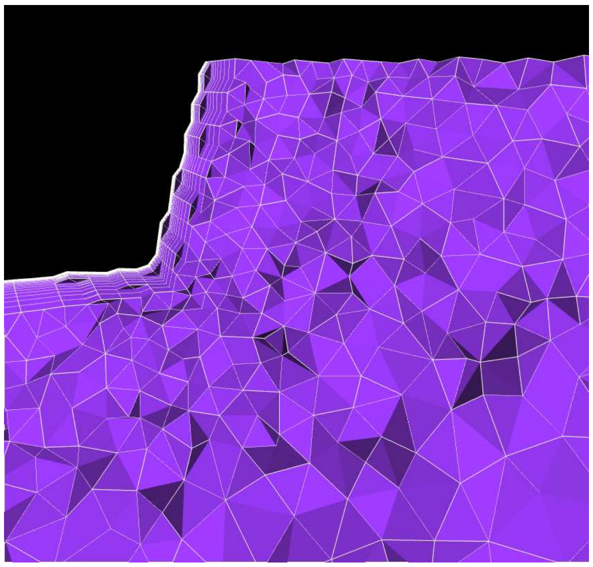
Figure 21. Grid for Original Model 5415.

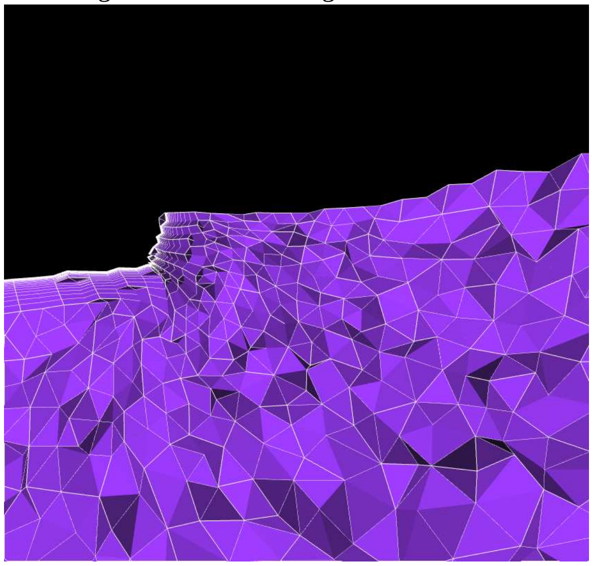
Figure 22. Grid for Deformed Model 5415.

A comparison of the grids along one cutting plane of the grid is shown in Figures 21 and 22. The cutting plane is parallel to the centerline and is shifted so that it cuts through the depressed region behind the intersection of the stern and port side of the ship. The grid is compressed at the stern by approximately 60%, and the deformation in the stern region is quite complicated being both compressed and expanded in neighboring areas.

Both the Series 60 hull form and the DTMB Model 5415 series hull form have grids that include prisms and pyramids near the viscous surfaces. The points nearest the hull are moved rigidly with the deformations on the viscous surfaces, as the free surface rises and falls. However, outside of a certain distance away from the viscous surfaces, the nodes move in response to the forces acting through the torsional and linear springs within the mesh. From the success of these two simulations, this torsional spring method clearly can be applied to elements other than tetrahedra.

V. Conclusion

A three-dimensional version of the torsional spring method has been presented that considers the three-dimensional contributions of each element without reducing the elements to two-dimensions. In order to derive this formulation, the two-dimensional linear and torsional springs formulations were recast so that the extension to three-dimensions would be obvious. This formulation is extendible to other elements besides the basic tetrahedron, since it only considers the corners of each element.

This algorithm is demonstrated on two benchmark cases, dealing with a three-dimensional extension of Farhat's simplest benchmark case and with a cube within a cube. In both cases, the grid is compressed significantly farther than could be accomplished with linear springs. Then the grid movement algorithm is demonstrated using three free surface flow simulations, including flow past a submerged hydrofoil, flow around the Series 60 C_b=60 hull form which has a tapered stern region, and flow around the DTMB Model 5415 series hull, which has a transom stern. In each case, the deformed grid remains valid even after moderate to severe distortions of the grid.

Acknowledgments

This work was supported by the Office of Naval Research under the direction of L. Patrick Purtell in conjunction with efforts to simulate the free surface flow around surface vessels. The support is gratefully acknowledged.

References

[1] Batina, J. T., *Unsteady Euler Airfoil Solutions Using Unstructured Dynamic Meshes*, AIAA Paper 89-0115, 27th AIAA Aerospace Sciences Meeting, January, 1989.

[2] Anderson, W. K., *Aerodynamic Design Optimization on Unstructured Grids with a Continuous Adjoint Formulation*, AIAA Paper 97-0643, 35th AIAA Aerospace Sciences Meeting, January 1997.

[3] Singh, K. P., Newman, J. C., and Baysal, O., *Dynamic Unstructured Method for Flows Past Multiple Objects in Relative Motion*, AIAA Journal, Vol. 33, No. 4, pp. 641-649, April, 1995.

[4] Murayama, M., Nakahashi, K., and Matsushima, K., *Unstructured Dynamic Mesh For Large Movement and Deformation*, AIAA Paper 2002-0122, 40th AIAA Aerospace Sciences Meeting, January, 2002.

[5] Farhat, C., Degand, C., Koobus, B., and Lesoinne, M., *Torsional Springs for Two-Dimensional Dynamic Unstructured Fluid Meshes*, Comput. Methods Appl. Mech. Engrg 163, pp. 231-245, 1998.

[6] Degand, C., Farhat, C., *A Three-Dimensional Torsional Spring Analogy Method for Unstructured Dynamic Meshes*, Computers and Structures, 80, pp. 305-316, 2002.

[7] Burg, C. O. E., Sreenivas, K., Hyams, D. G., and Mitchell, B., *Unstructured Nonlinear Free Surface Flow Solutions: Validation and Verification*, AIAA Paper 2002-2977, 32nd AIAA Fluid Dynamics Conference, St. Louis, June, 2002.