INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2000; **32**: 647–668

Considerations on the spring analogy

Frederic J. Blom*

Institut de Machines Hydrauliques et Mécanique des Fluides, Ecole Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland

SUMMARY

This paper presents an investigation on the spring analogy. The spring analogy serves for deformation in a moving boundary problem. First, two different kinds of springs are discussed: the vertex springs and the segment springs. The vertex spring analogy is originally used for smoothing a mesh after mesh generation or refinement. The segment spring analogy is used for deformation of the mesh in a moving boundary problem. The difference between the two methods lies in the equilibrium length of the springs. By means of an analogy to molecular theory, the two theories are generalized into a single theory that covers both. The usual choice of the stiffness of the spring is clarified by the mathematical analysis of a representative one-dimensional configuration. The analysis shows that node collision is prevented when the stiffness is chosen as the inverse of the segment length. The observed similarity between elliptic grid generation and the spring analogy is also investigated. This investigation shows that both methods update the grid point position by a weighted average of the surrounding points in an iterative manner. The weighting functions enforce regularity of the mesh. Based on these considerations, several improvements on the spring analogy are developed. The principle of Saint Venant is circumvented by a boundary correction. The prevention of inversion of triangular elements is improved by semi-torsional springs. The numerical results show the superiority of the segment spring analogy over the vertex one for a small rotation of an NACA0012 mesh. The boundary correction allows for large deformation of the mesh, where the standard spring analogy fails. The final test is performed on a Navier–Stokes mesh. This mesh contains high aspect ratio mesh cells near the boundary. Large deformation of this mesh shows that the semi-torsional spring improvement is imperative to retain the validity of this mesh. Copyright © 2000 John Wiley & Sons, Ltd.

KEY WORDS: elliptic grid generation; mesh deformation; moving boundaries; unstructured meshes

1. INTRODUCTION

In modern computational fluid dynamics the moving boundary problem plays an important role. It arises, for example, in free surface flows, store separation problems, forced vibration problems and fluid–structure interaction. A crucial part of the computation is the deformation

* Correspondence to: De Nieuwkamp 35, 7447 BH Hellendoorn, The Netherlands.

Received May 1998
Revised November 1998

of the mesh during the time integration of the fluid. A common technique to deform a mesh is the spring analogy. The first objective of this paper is to present an original analysis of the spring analogy in order to enhance the insight into the methods. The second objective is to improve the spring analogy based on the analysis.

The spring analogy consists of replacing the mesh by fictitious springs. For unstructured triangular grids, the spring analogy is introduced by Batina [1], who used the method for a forced vibration problem of an airfoil. Many researchers have adopted the spring analogy to solve moving boundary problems. To give a few examples, the spring analogy is used for free surface problems by Slikkeveer *et al.* [2], store separation problems by Hassan *et al.* [3], forced vibration and fluid–structure interaction problems by Blom and Leyland [4], and aeroelastic calculations by Farhat *et al.* [5] and Piperno [6]. The spring analogy is also suitable for structured meshes as is, for example, shown by Nakahashi and Deiwert [7], who used it for mesh adaptation.

Another possibility of coping with the moving boundary problem is to regenerate the mesh several times in the course of the simulation. This method is applied by Grüber and Carstens [8] on a cascade in forced vibration. They used structured meshes generated by elliptic grid generation. Their mesh is interpolated in time between the meshes at maximum amplitude. Schulze [9] applies the same technique on an aeroelastic motion of an airfoil. The interpolation technique is impossible in aeroelasticity since the motion of the structure is a part of the problem. Therefore, the mesh has to be regenerated after every time step. This seems rather cumbersome from the computational point of view. However, it will be shown in this paper that the computational techniques for elliptic grid generation and the spring analogy are in fact very similar.

The spring analogy is applied independently from the moving boundary problem on unstructured mesh smoothing. There, the spring analogy is used to move the nodes inside an unstructured triangular mesh to obtain equilateral elements. Richter and Leyland [10] applied the spring analogy after adaptation cycles to restore the smoothness in the mesh. Weatherhill *et al.* [11] applied the method to smooth the mesh after unstructured grid generation.

The spring analogy that is used for moving boundary problems is different from the analogy used for mesh smoothing. Both methods and their differences are presented in Section 2. These methods are then analysed in Section 3. Based on these analyses, some improvements are developed in Section 4. The numerical results obtained by the methods are shown in Section 5. Finally, the conclusions are presented in Section 6.

2. NUMERICAL METHODS

In this section two different spring analogies are presented. The terms vertex and segment springs are introduced here in order to distinguish between the different methods. The vertex spring analogy is discussed in Section 2.1, with the segment spring analogy reviewed in Section 2.2.

2.1. Vertex springs

When the segments are considered as springs, they have to possess an equilibrium length. For the vertex method, this equilibrium length is zero. The springs are taken as linear and therefore Hook's law determines the force at every node i exerted by the nodes j, which are connected to node i,

$$\vec{F}_i = \sum_{j=1}^{v_i} \alpha_{ij}(\vec{x}_j - \vec{x}_i) \tag{1}$$

where α_{ij} is the stiffness of the spring between node i and j and v_i is the number of neighbours of node i. For the system to be in equilibrium, the force at every node i has to be zero. After regrouping, the iterative equation to be solved yields

$$\vec{x}_i^{\,k+1} = \frac{\sum_{j=1}^{v_i} \alpha_{ij}\vec{x}_j^{\,k}}{\sum_{j=1}^{v_i} \alpha_{ij}} \tag{2}$$

This equation is solved for every internal node i in the mesh. Equation (2) converges to static equilibrium of the spring system. Equilibrium is not imperative for the mesh deformation since the spring model is only used as a tool to retain mesh regularity. Therefore, the number of iterations can remain reasonably small.

The boundary conditions are of the Dirichlet type. The positions of the boundary nodes are fixed during the calculation. The iterations are only performed on the internal nodes $\vec{x}_i$. When a boundary is moved or deformed, the position of the boundary nodes are strongly imposed by the Dirichlet boundary conditions.

The physical interpretation of Equation (2) is depicted in Figure 1. The elastic problem is solved for every node at each iteration. This means that, at every iteration, the new nodal position $\vec{x}_i$ is calculated as a weighted average of the surrounding nodes. The weighting factor is given by the spring stiffness α_{ij}. Mathematically, this implies performing Jacobi iterations on a linear system $[A]\{x\} = \{b\}$. In this case, the matrix $[A]$ is formed by the spring stiffness α_{ij}. The vector $\{x\}$ contains the mesh positions. The vector $\{b\}$ contains the non-homogeneous terms, which are implicitly formed by the Dirichlet boundary conditions. By solving the partial

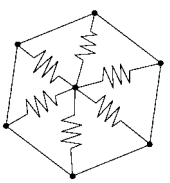

Figure 1. Physical interpretation of the iterative method in the spring analogy.

problems in (2), one avoids factorization of the large banded matrix $[A]$. Since the equilibrium position of the springs is zero, every spring is under tension. Therefore, the mesh can be deformed by this spring analogy even when the boundaries are stationary.

The stiffness in the vertex spring analogy is taken constant. Since the numerical value of this constant has no influence, it is chosen as unity

$$\alpha_{ij} = 1, \quad \forall i, j \tag{3}$$

The position of the mesh points is calculated using Equations (2) and (3) as the metric mean of the surrounding nodes.

2.2. Segment springs

The segment spring analogy is proposed by Batina [1] in order to deform a mesh around a pitching airfoil. In this method, the equilibrium lengths of the springs are equal to the initial lengths of the segments. Hook's Law is applied to the displacement of the nodes. The force is written as

$$\vec{F}_i = \sum_{j=1}^{v_i} \alpha_{ij}(\vec{\delta}_j - \vec{\delta}_i) \tag{4}$$

where $\vec{\delta}_i$ is the displacement of node i. At static equilibrium of the system, the force at every node i has to be zero. The iterative equation to be solved reads

$$\vec{\delta}_i^{\,k+1} = \frac{\sum_{j=1}^{v_i} \alpha_{ij}\vec{\delta}_j^{\,k}}{\sum_{j=1}^{v_i} \alpha_{ij}} \tag{5}$$

Here, the Dirichlet boundary conditions are given by the known displacement of the boundaries. As proposed by Batina [1], the spring stiffness is taken as proportional to the inverse of the segment length

$$\alpha_{ij} = \frac{1}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}} \tag{6}$$

This choice for the stiffness is analysed in Section 3.2. After the iteration of (5), the nodes are displaced by adding the final displacement

$$\vec{x}_i^{\,\text{new}} = \vec{x}_i^{\,\text{old}} + \vec{\delta}_i^{\,k,\text{final}} \tag{7}$$

The memory requirements for the segment method are higher than these for the vertex method, as the displacement $\vec{\delta}$ has to be kept in memory.

3. ANALYSIS OF THE METHODS

The methods presented in the previous section are analysed here. First, a general theory that covers the segment and the vertex methods is presented in Section 3.1. Then, the stiffness in the segment method is analysed in Section 3.2. Finally, the analogy with elliptic grid generation is shown is Section 3.3.

3.1. Generalization of the methods

The only difference between vertex and segment springs is their equilibrium length. In this section, the formulation of the spring analogy is generalized to a theory that covers both methods. In order to show this, an analogy to the theory of molecular dynamics is used. This theory is very similar to that of unstructured grid generation/deformation when the molecules are regarded upon as mesh points, as pointed out by Borhani [private communication, 1998]. A good introduction to the theory of molecular dynamics is given in the textbook by Fosdick *et al.* [12].

In molecular dynamics, the force field among the molecules is assumed to be conservative. Therefore, there exists a potential function around every molecule. The force exerted by the molecules is calculated by the gradient of the total potential function. Since the problem is linear, the total potential is calculated as the sum of all partial potential functions. When the kinetic energy of the molecules is low, the dynamics can be represented by Hook's Law [12,13]. The radial potential energy function is then proportional to the square of the displacement from the equilibrium position. The force exerted on molecule i by the other molecules is then given by

$$\vec{F}_i = \sum_{j=1}^{N} \alpha_{ij}(\vec{x}_j - \vec{x}_i - \vec{d}_{ij}) \tag{8}$$

where $\vec{d}_{ij}$ is the equilibrium vector between node i and j. N denotes the total number of molecules modelled. This equation is very similar to (1). To show the analogy, the molecules are replaced by mesh points and the connectivity of the mesh is also taken into account. The influence of the molecules is localized by replacing the number of molecules N by the number of neighbours v_i. The force is then calculated by

$$\vec{F}_i = \sum_{j=1}^{v_i} \alpha_{ij}(\vec{x}_j - \vec{x}_i - \vec{d}_{ij}) \tag{9}$$

This force is almost the same as in (1). It is directly seen that the vertex spring analogy has zero equilibrium length since $\vec{d}_{ij}$ is zero in (1).

To show the equivalence of (9) and the segment spring force (4), the equilibrium vector $\vec{d}_{ij}$ is written as

$$\vec{d}_{ij} = \vec{x}_j^{\,\mathrm{old}} - \vec{x}_i^{\,\mathrm{old}} \tag{10}$$

where $\vec{x}_j^{\text{old}}$ is the initial position of node x_j. Then (9) is rewritten as

$$\vec{F}_i = \sum_{j=1}^{v_i} \alpha_{ij}((\vec{x}_j^{\text{new}} - \vec{x}_j^{\text{old}}) - (\vec{x}_i^{\text{new}} - \vec{x}_i^{\text{old}})) \tag{11}$$

With Equation (7), this force is the same as in (4).

The force given by (9) is a generalization of the vertex and segment methods. The equilibrium vector $\vec{d}_{ij}$ is an arbitrary constant in this model.

3.2. Analysis of the stiffness in the segment method

In this section the stiffness of the springs (6) in the segment spring analogy is analysed. The choice made in (6), which is proposed by Batina [1], is used by many researchers. The same researchers who adopted the spring analogy of Batina, e.g. References [4–6], also used this stiffness. The stiffness in Equation (6) is logical since mesh points that are closer to each other exhibit a stronger force. Since node collision is more probable in dense regions, Equation (6) diminishes the probability of node collision.

To analyse the stiffness, a representative system of linear springs on a line is considered. This configuration is depicted in Figure 2. When the system consists of n springs, the force equilibrium of the nodes gives

$$\begin{bmatrix} k_1+k_2 & -k_2 & & & & 0 \\ -k_2 & k_2+k_3 & -k_3 & & & \\ & -k_3 & k_3+k_4 & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & k_{n-1}+k_n & -k_{n-1} \\ 0 & & & & -k_{n-1} & k_n \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{n-1} \\ u_n \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{Bmatrix} \tag{12}$$

The derivation is performed on a system with general k_i and Δx_i. The spring stiffness k_i for which the system conserves the order of the nodes is searched for. From Equation (12), the equilibrium of the node i is found,

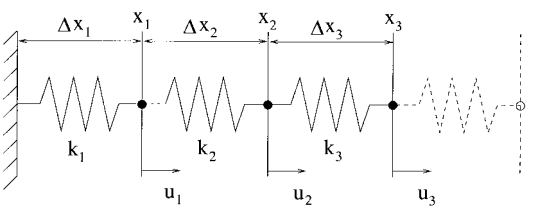

Figure 2. Configuration of linear springs.

$$-k_i u_{i-1} + (k_i + k_{i+1})u_i = k_{i+1}u_{i+1} \tag{13}$$

The displacement of node $i+1$ is given by the next theorem.

Theorem 1
$\forall i \in \mathbb{N}$, $i \geq 1$, u_{i+1} is given by

$$u_{i+1} = \left(\frac{\sum_{j=1}^{i+1} \frac{1}{k_j}}{\sum_{j=1}^{i} \frac{1}{k_j}} \right) u_i \tag{14}$$

Proof
The proof is given by induction. For $i = 1$, the relation is found on the first line of the system (12), which states

$$(k_1 + k_2)u_1 = k_2 u_2 \Rightarrow \left(\frac{\frac{1}{k_1} + \frac{1}{k_2}}{\frac{1}{k_1}} \right) u_1 = u_2 \tag{15}$$

Next, Equation (13) is used to solve u_{i+2},

$$-k_{i+1}u_i + (k_{i+1} + k_{i+2})u_{i+1} = k_{i+2}u_{i+2} \tag{16}$$

This is divided by k_{i+2} and Equation (14) is used to express u_i,

$$\left(-\frac{k_{i+1}}{k_{i+2}} \left(\frac{\sum_{j=1}^{i} \frac{1}{k_j}}{\sum_{j=1}^{i+1} \frac{1}{k_j}} \right) + \frac{k_{i+1}}{k_{i+2}} + 1 \right) u_{i+1} = u_{i+2} \tag{17}$$

which gives, after some algebra,

$$\left(\frac{\sum_{j=1}^{i+2} \frac{1}{k_j}}{\sum_{j=1}^{i+1} \frac{1}{k_j}} \right) u_{i+1} = u_{i+2} \tag{18}$$

This concludes the proof of Theorem 1. □

The displacement of node x_n is given by $u_n = p$. Then the new co-ordinates of x_n and x_{n-1} are calculated by

$$x_n^{\text{new}} = x_n^{\text{old}} + p,$$

$$x_{n-1}^{\text{new}} = x_{n-1}^{\text{old}} + p \frac{\sum_{i=1}^{n-1} \frac{1}{k_i}}{\sum_{i=1}^{n} \frac{1}{k_i}} \tag{19}$$

Subtraction of these two terms gives Δx_n^{new},

$$\Delta x_n^{\text{new}} = \Delta x_n^{\text{old}} + p \left(1 - \frac{\sum_{i=1}^{n-1} \frac{1}{k_i}}{\sum_{i=1}^{n} \frac{1}{k_i}} \right) \tag{20}$$

The displacement p is now expressed by

$$p = \alpha \sum_{i=1}^{n} \Delta x_i \tag{21}$$

Then, Equation (20) gives

$$\Delta x_n^{\text{new}} = \Delta x_n^{\text{old}} + \alpha \sum_{i=1}^{n} \Delta x_i \left(\frac{\frac{1}{k_n}}{\sum_{i=1}^{n} \frac{1}{k_i}} \right) \tag{22}$$

The spring stiffness is now chosen as

$$k_i = \frac{1}{\Delta x_i} \tag{23}$$

Then Δx_n^{new} is given by

$$\Delta x_n^{\text{new}} = \Delta x_n^{\text{old}}(1 + \alpha) \tag{24}$$

So, $\Delta x_n^{\text{new}} > 0$ if $\alpha > -1$. This derivation proves that the usual choice of the stiffness made in (23) prevents the nodes from colliding when they are placed on a line and move along this line. Equation (23) is equivalent to Equation (6) since the stiffness is the inverse of the segment length.

3.3. Analogy with elliptic grid generation

The vertex spring analogy exhibits a similar behaviour to elliptic grid generation for structured grids. The triangular mesh contracts near convex boundaries and expands near concave

boundaries. This behaviour is also observed and analysed in elliptic grid generation by Carstens [14].

In order to show this behaviour here, an unstructured and a structured mesh around the NACA0012 airfoil are subjected to the vertex spring analogy and elliptic grid generation respectively. The original unstructured mesh is shown in Figure 3. The mesh consists of 1352 nodes and 2628 elements. This mesh is used as the initial mesh for the vertex spring analogy. The result after 3000 iterations is shown in Figure 4. The original structured mesh is depicted in Figure 5. The mesh contains 40 nodes in a circumferential direction and 30 nodes in a radial direction. This mesh is used as an initial solution for the elliptic grid generation. The mesh after 3000 iterations is shown in Figure 6. The contraction of the mesh near the convex boundary is clearly seen for the unstructured as well as for the structured mesh. This observation of similarity demands further investigation to get insight into these phenomena.

In order to investigate the observed relationship, elliptic grid generation is reviewed here. In elliptic grid generation, the physical co-ordinates (x, y) are mapped onto the computational co-ordinates (ξ, η). This mapping is shown in Figures 7 and 8 for a 19×19 O-grid around a circle.

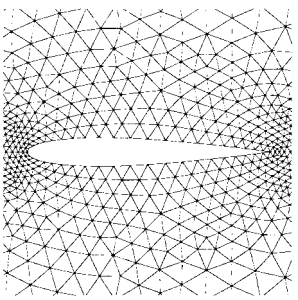

Figure 3. Original unstructured mesh.

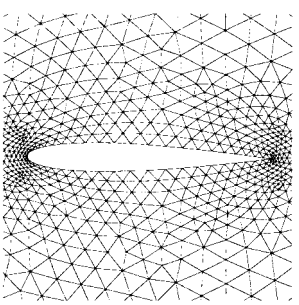

Figure 4. Deformed mesh by vertex springs.

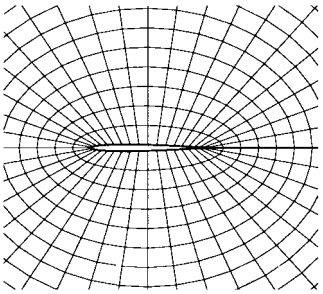

Figure 5. Original structured mesh.

In order to obtain a smooth physical mesh, the mapping is subjected to the following elliptic partial differential equation:

$$\xi_{xx} + \xi_{yy} = P(\xi, \eta),$$

$$\eta_{xx} + \eta_{yy} = Q(\xi, \eta) \tag{25}$$

From the computational viewpoint, the inverse problem is more interesting. The inverse problem is derived by transforming partial derivatives to (x, y) into partial derivatives to (ξ, η) by means of the chain rule. This gives

$$\alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} = -J^2(x_\xi P(\xi, \eta) + x_\eta Q(\xi, \eta)),$$

$$\alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} = -J^2(y_\xi P(\xi, \eta) + y_\eta Q(\xi, \eta)) \tag{26}$$

where

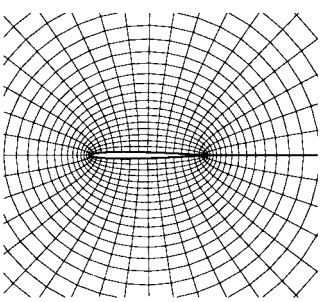

Figure 6. Elliptic grid generated mesh.

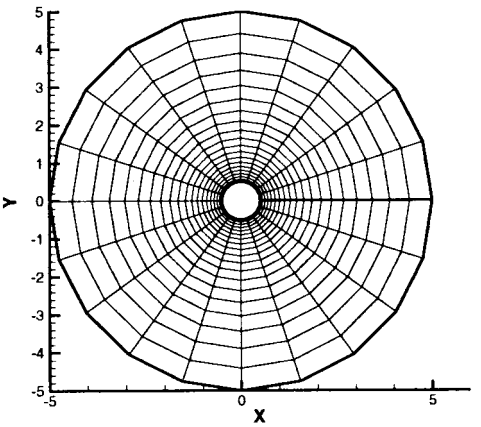

Figure 7. (x, y) mesh.

$$\alpha = x_\eta^2 + y_\eta^2, \qquad \beta = x_\xi x_\eta + y_\xi y_\eta, \qquad \gamma = x_\xi^2 + y_\xi^2, \qquad J = x_\xi y_\eta - x_\eta y_\xi \tag{27}$$

The functions P and Q are control functions that determine the angle of the grid lines and their spacing respectively. The Q function controls the spacing of $\eta = \text{constant}$ lines near a boundary. When $Q > 0$, the $\eta = \text{constant}$ lines are repulsed from a boundary, and when $Q < 0$, $\eta = \text{constant}$ lines are attracted to the boundary. The P function controls the inclination of the $\xi = \text{constant}$ lines to the boundary. When $P > 0$, the $\xi = \text{constant}$ lines are rotated to the right, and when $P < 0$, the $\xi = \text{constant}$ lines are rotated to the left. This is shown mathematically by Carstens [14].

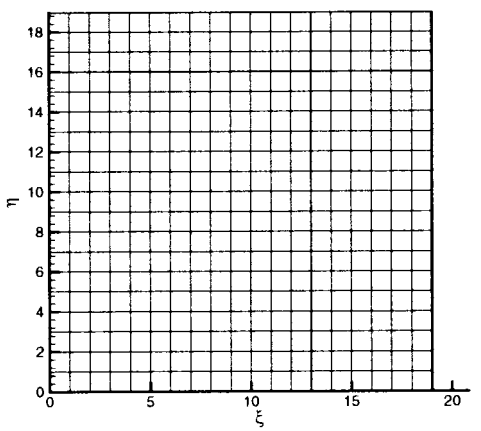

Figure 8. (ξ, η) mesh.

Equation (26) requires boundary conditions. The boundary conditions are of the Dirichlet type since the external and body boundaries are given data, like in the spring analogy. At the left- and right-hand side of the computational co-ordinates, periodicity boundary conditions are given to obtain an O-grid (see Figure 7). The Dirichlet boundary conditions are given by

$$\left.\begin{aligned} x(\xi, \eta_{\min}) &= x_1(\xi) \\ y(\xi, \eta_{\min}) &= y_1(\xi) \\ & \\ x(\xi, \eta_{\max}) &= x_2(\xi) \\ y(\xi, \eta_{\max}) &= y_2(\xi) \end{aligned}\right\} \xi_{\min} \le \xi \le \xi_{\max} \tag{28}$$

The periodic boundary conditions are given by

$$x(\xi_{\max}, \eta) = x(\xi_{\min}, \eta), \qquad y(\xi_{\max}, \eta) = y(\xi_{\min}, \eta) \tag{29}$$

Next, the equations need to be solved numerically. This is done by a finite difference technique. The grid system in computational space (ξ, η) is formed by a set of equidistant grid points. The number of points in the ξ-direction is M and the number of points in the η-direction is N. These points are separated by unity distance. Then the system of grid points is given by

$$\xi_i = i, \quad i = 1, M; \qquad \eta_j = j, \quad j = 1, N$$

$$\Delta\xi = \Delta\eta = 1 \tag{30}$$

Then the partial derivatives of the physical co-ordinates to the computational co-ordinates are calculated as central differences

$$x_\xi = \frac{1}{2}(x_{i+1,j} - x_{i-1,j})$$

$$x_\eta = \frac{1}{2}(x_{i,j+1} - x_{i,j-1})$$

$$x_{\xi\xi} = x_{i+1,j} - 2x_{i,j} + x_{i-1,j}$$

$$x_{\eta\eta} = x_{i,j+1} - 2x_{i,j} + x_{i,j-1}$$

$$x_{\xi\eta} = \frac{1}{4}(x_{i+1,j+1} - x_{i+1,j-1} - x_{i-1,j+1} + x_{i-1,j-1}) \tag{31}$$

These expressions are then substituted in Equation (26), which gives

$$2\alpha(x_{i+1,j}-2x_{i,j}+x_{i-1,j})-\beta(x_{i+1,j+1}-x_{i+1,j-1}-x_{i-1,j+1}+x_{i-1,j-1})$$
$$+2\gamma(x_{i,j+1}-2x_{i,j}+x_{i,j-1})=-J^2[(x_{i+1,j}-x_{i-1,j})P_{i,j}+(x_{i,j+1}-x_{i,j-1})Q_{i,j}] \tag{32}$$

where

$$\alpha=\frac{1}{4}[(x_{i,j+1}-x_{i,j-1})^2+(y_{i,j+1}-y_{i,j-1})^2]$$

$$\beta=\frac{1}{4}(x_{i+1,j}-x_{i-1,j})(x_{i,j+1}-x_{i,j-1})+\frac{1}{4}(y_{i+1,j}-y_{i-1,j})(y_{i,j+1}-y_{i,j-1})$$

$$\gamma=\frac{1}{4}[(x_{i+1,j}-x_{i-1,j})^2+(y_{i+1,j}-y_{i-1,j})^2]$$

$$J=\frac{1}{4}(x_{i+1,j}-x_{i-1,j})(y_{i,j+1}-y_{i,j-1})-\frac{1}{4}(x_{i,j+1}-x_{i,j-1})(y_{i+1,j}-y_{i-1,j}) \tag{33}$$

For Equations (31) and (32), analogous expressions are found in the y co-ordinate.

The discrete equations are now solved by means of the Jacobi iterative method. An algebraic mesh is usually chosen as an initial solution to the problem. Then, the next x iterate is found by the following update:

$$\bar{x}_{i,j}^{k+1}=\frac{\alpha^k(x_{i+1,j}^k+x_{i-1,j}^k)+\gamma^k(x_{i,j+1}^k+x_{i,j-1}^k)}{2(\alpha^k+\gamma^k)}$$
$$-\beta^k(x_{i+1,j+1}^k-x_{i+1,j-1}^k-x_{i-1,j+1}^k+x_{i-1,j-1}^k)$$
$$+(J^k)^2[(x_{i+1,j}^k-x_{i-1,j}^k)P_{i,j}+(x_{i,j+1}^k-x_{i,j-1}^k)Q_{i,j}] \tag{34}$$

followed by

$$x_{i,j}^{k+1}=\omega\bar{x}_{i,j}^{k+1}+(1-\omega)x_{i,j}^k \tag{35}$$

where ω is a relaxation parameter. Since the equation is non-linear (the factors α, β, γ and J are a function of x), the relaxation parameter has to be chosen as $0<\omega\leq 1$.

From Equation (34) it is seen that the next iterate $x_{i,j}^{k+1}$ is a function of $x_{i-1,j-1}^k$, $x_{i-1,j+1}^k$, $x_{i+1,j-1}^k$ and $x_{i+1,j+1}^k$. These values are all neighbours of $x_{i,j}$, so Equation (34) can be rewritten as follows:

$$\bar{x}_{i,j}^{k+1}=\frac{\sum_{p=1}^{v_i}\alpha_{ip}x_p^k}{\sum_{p=1}^{v_i}\alpha_{ip}}+\sum_{p=1}^{\Upsilon_i}\beta_{ip}x_p^k \tag{36}$$

In this equation, the α^k and γ^k terms are accounted for by the first summation over v_I, which is the number of direct neighbours of $x_{i,j}$ (upper, lower, left- and right-hand). The β_k and $(J^k)^2$ terms are placed in the second sum over Υ_i, which is the extended number of neighbours. This number also contains the diagonal neighbours. Equation (36) is very similar to Equation (2) for the vertex spring analogy. In elliptic grid generation the next iterate is also found by means of a weighted mean of the surrounding nodes.

The physical meaning of the α, β, γ and J weighting functions is as follows: α stands for the length in j-direction square; γ stands for the length in i-direction square (see Figure 9); β stands for a measure of the deviation of orthogonality of the lines $x_{i,j+1} - x_{i,j-1}$ and $x_{i+1,j} - x_{i-1,j}$ (see Figure 10). β is equal to zero when the $\xi =$ constant and $\eta =$ constant lines are orthogonal. Finally, J stands for the Jacobian, which is the area of the element in the physical co-ordinates. The influence of the control functions P and Q has already been mentioned. Their influence is weighted by the Jacobian squared.

Next, only one iteration is considered and no relaxation is applied ($\omega = 1$). The mesh is forced to be as equilateral as possible by the following corrections: the new position of $x_{i,j}$ is a mean of the node above and below, weighted by the squared horizontal distance between the node on the left and right; the position is also weighted by the mean of the node on the left and right weighted by the squared vertical distance between the node above and below; then the orthogonality of the mesh is enforced by the β term, which adds a weighted mean of the diagonal neighbours. The weighting functions force the mesh to be very regular. This results

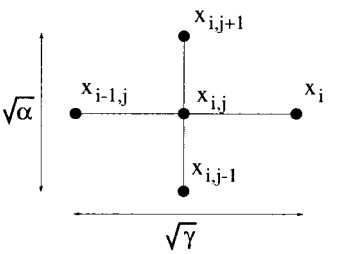

Figure 9. Definition of α and γ.

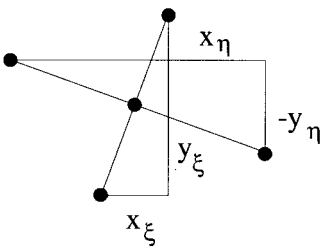

Figure 10. $\beta = x_\xi x_\eta + y_\xi y_\eta$.

in very smooth orthogonal meshes, as is shown for the NACA0012 airfoil in Figure 6 and for a circle in Figure 7.

This analysis shows the capability of the weighting functions. They determine the regularity of the mesh. As the spring analogy also contains weighting functions, these allow for improvements of the method. The similarity is originally observed for the vertex spring method. However, the analysis in Section 3.1 shows that the vertex and segment method belong to the same covering theory. Therefore, the analogy holds for the complete spring analogy. The difference between elliptic grid generation and the spring analogy is the structure of the data, which is less regular for unstructured meshes. In the next section, the weighting functions of the spring analogy are used for improvements.

As mentioned in Section 1, regeneration of the mesh is also possible for a moving boundary problem. In aeroelastic and free surface problems, where the moving boundary is part of the solution, it seems rather cumbersome to regenerate a mesh after every time step. However, Equation (36) shows that elliptic mesh generation is nothing other than a manipulation of the existing mesh. When this mesh already satisfies the elliptic equation, the number of iterations in (36) can be rather small since the initial solution is close to the converged solution. Therefore, 'elliptic mesh manipulation' seems to be a more appropriate term than 'elliptic mesh generation', since one has to start with an existing mesh to obtain the final one.

4. IMPROVEMENTS OF THE METHODS

An analysis on a one-dimensional spring system in Section 3.2 provided a suitable choice of the spring stiffness in order to avoid node collisions. This analysis only considered springs along a line. In this section some improvements of the algorithm on two-dimensional triangular unstructured meshes are proposed. In Section 4.1, a boundary improvement is developed. A semi-torsional improvement is discussed in Section 4.2.

4.1. Boundary improvement

The regularity of the mesh after transformation is determined by the weighting functions, as observed in elliptic grid generation. Therefore, the spring stiffness is allowed to change by the introduction of two parameters in the definition of the stiffness,

$$\alpha_{ij} = \phi((x_i - x_j)^2 + (y_i - y_j)^2)^{\psi} \tag{37}$$

In this way, the parameters ϕ and ψ allow for modification of the spring stiffness. In the original concept of Batina [1], the parameter values are $\phi = 1$ and $\psi = -0.5$.

The spring analogy is shown to be very similar to elliptic grid generation in the previous section. The resulting equation from the spring analogy is also elliptic, as was shown by Palmerio [15]. Consequently, the principle of Saint Venant for elliptic equations holds for the deformation of the mesh by the spring analogy. This principle states that local perturbations of the solution only have local impact. The purpose of the spring analogy is to regularize a mesh after boundary deformation. When the mesh is regularized by the spring analogy, this

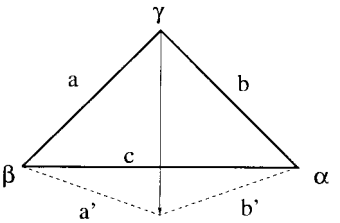

Figure 11. Snap-through of cell.

regularization only takes place close to the deformed boundary. Therefore, the spring analogy can only handle relatively small deformations since the mesh regularizes only locally. In order to circumvent the principle of Saint Venant, the stiffness near the boundary is increased so that the deformation is spread out further into the mesh. To achieve this local stiffening of the system, the factor ϕ in Equation (37) is increased for a number of element layers adjacent to the boundary.

4.2. Semi-torsional improvement

A significant problem in deforming unstructured triangular grids is cell inversion. The one-dimensional consideration presented in Section 3.2 only showed the prevention of one-dimensional cell inversion. Triangular grids possess the additional difficulty of cell inversions where the summit of the triangle passes through the opposite face (see Figure 11). This movement will be referred to as 'snap-through'. This snap-through requires relatively little energy since the springs on segments a and b rotate. As rotational motion does not store any energy in the springs, the snap-through of a mesh cell occurs relatively easy.

In order to prevent snap-through of mesh cells, Farhat *et al.* [16] proposed torsional springs that are placed in the corner between adjacent edges. These springs are superposed on lineal springs. These lineal springs are the same as those treated in this paper. The improved method performs very good, even when the motion is large and the mesh is fine.

It is well recognized that the angles play an important role in deforming the mesh. Therefore, the spring analogy is altered by incorporating the angle information in the spring stiffness here. In order to achieve a semi-torsional spring by a lineal spring system, the spring stiffness of a segment is divided by the angle of the edge facing the segment. Following Figure 11, this means that the stiffness of segment c is divided by the angle made between segments a and b. As the sum of all angles in a triangle is π, the stiffness will hardly change if the triangle is equilateral (since then $\alpha = \beta = \gamma = \pi/3 \approx 1$). The practical implementation of this semi-torsional spring system is obtained by dividing the multiplication factor ϕ in Equation (37) by the angle facing the segment. In this way a small angle inside an element faces a spring with a high stiffness. Consequently, the force required for snap-through is increased.

The iterative equation to be solved (5) remains linear. The only extra effort for this improvement lies in the calculation of the stiffness α_{ij} (37). In order to improve the performance of the semi-torsional spring method, the spring stiffness can be recalculated at

intermediate iteration levels. In this way, the mesh cells that are squeezed during the deformation are even more protected from snap-through. One has to be careful in applying this strategy since a non-linear system of equations is solved (α_{ij} becomes a function of the displacement). The non-linear system possesses different stability properties that can lead to divergence of the iterative solution.

5. NUMERICAL RESULTS

The numerical example used here is an unstructured mesh around the NACA0012 airfoil. This standard airfoil geometry has a reasonable sharp trailing edge, which is very suitable for showing the performance of the methods. The vertex and segment spring analogies are used to deform the mesh.

The mesh which is used for these simulations contains 3693 nodes and 7266 triangular elements. First, the airfoil is rotated 5° around the quarter chord. The mesh is then deformed by the vertex spring analogy. Three thousand iterations are sufficient for convergence of the method. The result is shown in Figure 12. It is seen that the final mesh is very regular and no cell inversion occurs. The deformation of the mesh is also performed by the segment spring analogy. Here also, 3000 iterations are performed. The mesh after applying the segment spring analogy is shown in Figure 13. This method also performs very efficiently for this case.

The drawback of the vertex method is the contraction of the mesh near a convex boundary. This property causes the mesh to be very fine near the leading edge of the airfoil (see Figure 14). The mesh that is deformed by the segment spring analogy does not possess this property and the size of the mesh cells near the leading edge remains about the same (Figure 15). Therefore, the segment spring analogy is more suitable for regularization of the mesh after boundary deformation. Hence, the segment method will be used from here on.

As the deformation of the mesh increases it becomes more difficult to restore the regularity of the mesh. In order to show the boundary improvement on the segment spring analogy, the NACA0012 airfoil is rotated 45° around the quarter chord. First, the mesh is restored with the standard spring analogy. In this case, the stiffness of the springs is calculated by Equation (6).

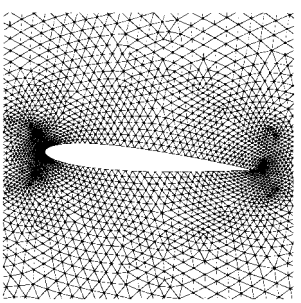

Figure 12. 5° Rotation of NACA0012, vertex spring method.

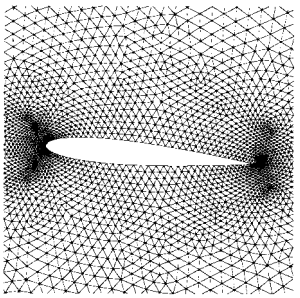

Figure 13. 5° Rotation of NACA0012, segment spring method.

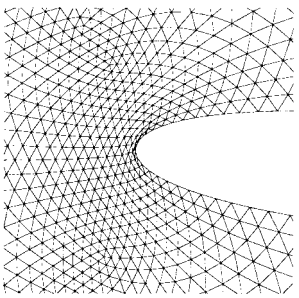

Figure 14. 5° Rotation of NACA0012, vertex spring method, zoom at leading edge.

The number of iterations on system (5) required for convergence is 3000. The resulting mesh is depicted in Figure 16. Only the trailing edge is shown since there the largest deformations take place. It is seen that the standard segment spring analogy is inadequate for such large deformations. Then the mesh is restored by the segment spring method with boundary

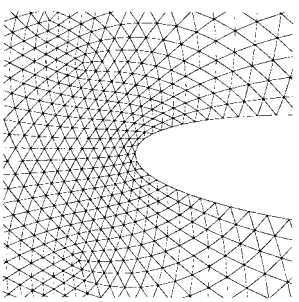

Figure 15. 5° Rotation of NACA0012, segment spring method, zoom at leading edge.

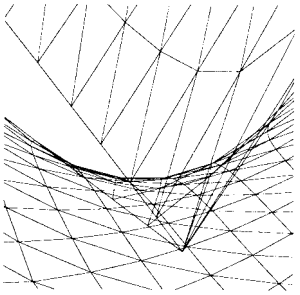

Figure 16. 45° Rotation of NACA0012, standard segment spring method.

improvement. In the interior of the mesh, the parameters ϕ and ψ are chosen so as $\phi = 1$ and $\psi = -1$. The springs are stiffened at one layer adjacent to the boundary. This has been shown to be sufficient. In this layer the parameters ϕ and ψ are chosen so as $\phi = 5$ and $\psi = -1$. The number of iterations taken is again 3000. The result of this simulation is shown in Figure 17. This figure shows that the boundary improvement is imperative in this case in order to restore the regularity of the mesh. Finally, the semi-torsional improvement is also applied to this problem. The values of the parameters are the same as in the previous calculation. The iterative equation (5) is applied 3000 times. After every 1000 iterations, the stiffness α_{ij} (37) is recalculated. The mesh after application of this method is shown in Figure 18. The semi-torsional improvement exhibits the best performance since the squeezing of mesh cells is significantly reduced compared with the previous methods.

The final test is performed on a Navier–Stokes mesh around the NACA0012 airfoil. A Navier–Stokes mesh contains high aspect ratio mesh cells near the airfoil in order to capture the boundary layer. Therefore, this is an ultimate test case for the improvements on the spring analogy since most problems occur close to the boundary and on stretched meshes. The mesh contains 4278 nodes and 8350 elements. The segment spring method with boundary improvement is compared with the semi-torsional improved segment spring method on this case. The airfoil is rotated 25.7° ($\pi/7$ rad) around the quarter chord. The parameters and number of

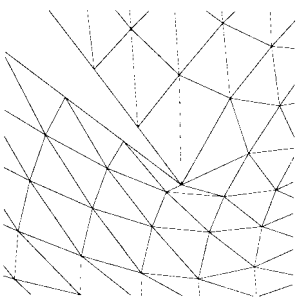

Figure 17. 45° Rotation of NACA0012, boundary-improved segment spring method.

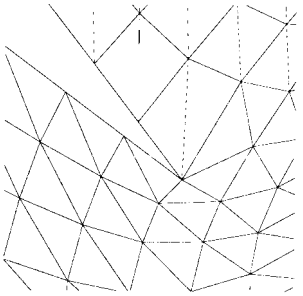

Figure 18. 45° Rotation of NACA0012, semi-torsional segment spring method.

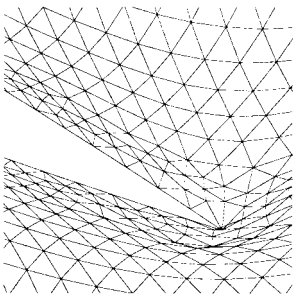

Figure 19. $\pi/7$ rad rotation of NACA0012, boundary-improved segment spring method.

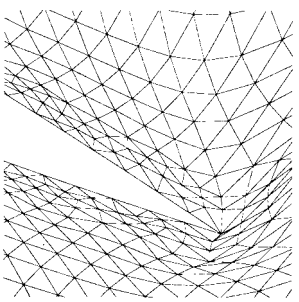

Figure 20. $\pi/7$ rad rotation of NACA0012, semi-torsional segment spring method.

iterations are taken the same as in the previous case. The result for the boundary-improved method is shown in Figure 19. As the deformation is the largest at the trailing edge, only a zoom at this region is shown. It is seen that even with this improved method, cell inversion can not be avoided. The mesh after application of the semi-torsional segment method is depicted

in Figure 20. This figure shows that in this case, the semi-torsional improvement is imperative in order to retain the validity of the mesh.

6. CONCLUSIONS

A thorough analysis of the spring analogy is presented here. The purpose of the spring analogy is to restore the regularity of the mesh after boundary deformation. The segments inside a mesh are replaced by fictitious springs to move the nodes. Two different spring analogies are distinguished. The first method (vertex) considers the equilibrium length of the spring to be zero. In the second method (segment), the equilibrium lengths of the springs are equal to their initial lengths. These two theories are generalized into a single theory that covers the vertex and the segment methods. This theory contains the equilibrium length as an additional input. The usual choice of the spring stiffness is shown to prevent node collision in a one-dimensional representative problem. Then, the observed similarity between the spring analogy and elliptic grid generation is investigated. It is shown that the elliptic grid generation technique possesses very similar features to the spring analogy. Based on the considerations two improvements on the spring analogy are developed. The principle of Saint Venant for elliptic equations is circumvented by a boundary correction. This boundary correction alters the spring stiffness near the moving boundary in order to spread out the deformation into the mesh. Then a semi-torsional spring is developed, which restrains the so-called snap-through movement. The methods are then applied to the rotation of an NACA0012 airfoil. Both methods perform well on a small rotation of the airfoil. The segment spring analogy is superior to the vertex one since it does not contract the mesh near convex boundaries. When the airfoil is rotated more substantially, the standard segment method fails whereas the boundary-corrected method still performs well. A large rotation of a Navier–Stokes mesh shows that the semi-torsional method is imperative to retain validity of the mesh. It is concluded that the analysis gives a good insight into the properties of the spring analogy, which is applied in many research studies involving moving boundaries. The improvements that are developed allow for large motion of fine meshes. These improvements are obtained by only small modifications of the algorithm.

ACKNOWLEDGMENTS

REFERENCES

1. Batina JT. Unsteady Euler airfoil solutions using unstructured dynamic meshes. *AIAA Journal* 1990; **28**(8): 1381–1388.
2. Slikkeveer PJ, van Loohuizen EP, O'Brien SBG. An implicit surface tension algorithm for Picard solvers of surface-tension-dominated free and moving boundary problems. *International Journal for Numerical Methods in Fluids* 1996; **22**: 851–865.
3. Hassan O, Probert EJ, Morgan K. Unstructured mesh procedures for the simulation of three-dimensional transient compressible inviscid flows with moving boundary components. *International Journal for Numerical Methods in Fluids* 1998; **27**: 41–55.
4. Blom FJ, Leyland P. *Analysis of fluid–structure interaction on moving airfoils by means of an improved ALE method*. AIAA Paper 97-1770, 1997.

5. Farhat C, Lesoinne M, Maman N. Mixed explicit/implicit time integration of coupled aeroelastic problems: three-field formulation, geometric conservation and distributed solution. *International Journal for Numerical Methods in Fluids* 1995; **21**: 807–835.
6. Piperno S. Explicit/implicit fluid/structure staggered procedures with a structural prediction and fluid subcycling for 2D inviscid aeroelastic simulations. *International Journal for Numerical Methods in Fluids* 1997; **25**: 1207–1226.
7. Nakahashi K, Deiwert GS. Three-dimensional adaptive grid method. *AIAA Journal* 1986; **24**(6): 948–954.
8. Grüber B, Carstens V. Computation of the unsteady transonic flow in harmonically oscillating turbine cascades taking into account viscous effects. *ASME Journal of Turbomachinery* 1998; **120**(1): 104–111.
9. Schulze S. *Transonic aeroelastic simulation of a flexible wing section*. AGARD Structures and Materials Panel Workshop on Numerical Unsteady Aerodynamics and Aeroelastic Simulation, AGARD R-822, Aalborg, Denmark, 13–17 October 1997; 10:1–10:20.
10. Richter R, Leyland P. Entropy correcting schemes and non-hierarchical auto-adaptive dynamic finite element type meshes: applications to unsteady aerodynamics. *International Journal for Numerical Methods in Fluids* 1995; **20**(6): 853–868.
11. Weatherhill NP, Gaither KP, Gaither JA. Building unstructured grids for computational fluid dynamics. *Computational Fluid Dynamics Journal* 1995; **4**(1): 1–28.
12. Fosdick LD, Jessup ER, Schauble CJC, Domik G. *An Introduction to High-Performance Scientific Computing*. MIT Press: Cambridge, MA, 1996.
13. Sears FW, Zemansky MW, Young HD. *College Physics*, 5th edn. Addison-Wesley: Reading, MA, 1980.
14. Carstens V. *Two-dimensional elliptic grid generation for airfoils and cascades*. Technical Report DFVLR-FB 88-52, DLR, 1988.
15. Palmerio B. An attraction–repulsion mesh adaption model for flow solution on unstructured grids. *Computers and Fluids* 1994; **23**(3): 487–506.
16. Farhat C, Degand C, Koobus B, Lesoinne M. Torsional springs for two-dimensional dynamic unstructured fluid meshes. *Computer Methods in Applied Mechanics and Engineering* 1998; **163**: 231–245.

