COMMUNICATIONS IN NUMERICAL METHODS IN ENGINEERING, Vol. 12, 599–608 (1996)

# IMPROVED ALE MESH VELOCITIES FOR MOVING BODIES

RAINALD LÖHNER AND CHI YANG
*GMU/CSI, George Mason University, Fairfax, VA 22030-4444, USA*

SUMMARY

A Laplacian smoothing of the mesh velocities with variable diffusivity based on the distance from moving bodies is introduced. This variable diffusivity enforces a more uniform mesh velocity in the region close to the moving bodies. Given that in most applications these are regions where small elements are located, the new procedure decreases element distortion considerably, reducing the need for local or global remeshing, and in some cases avoiding it altogether.

KEY WORDS finite elements; moving grids; moving bodies; mesh velocity; ALE

## 1. INTRODUCTION

For any arbitrary Lagrangean–Eulerian (ALE) unstructured-grid field solver that considers bodies or surfaces in relative motion to one another, a recurring question has been how to specify the mesh velocity of the field points.[1–10] In mathematical terms: given the velocity $\mathbf{w}$ on the moving surfaces:

$$\mathbf{w}|_{\Gamma_0} = \mathbf{w}_0, \tag{1}$$

and, at a certain distance from these moving surfaces, as well as all the remaining surfaces, a vanishing mesh velocity

$$\mathbf{w}|_{\Gamma_1} = 0, \tag{2}$$

find the spatial distribution of $\mathbf{w}$ such that element distortion is minimized. If this mesh velocity distribution is not smooth, distorted elements will appear quickly, forcing many local or global remeshings, with the ensuing loss of accuracy and increase in CPU requirements. Three families of methods have been used to specify the mesh velocity:

(a) Analytic user-prescribed functions,
(b) Smoothing of coordinates, and
(c) Smoothing of velocities.

In the first case, the mesh velocity is prescribed to be an analytic function of the distance from the surface. Efficient distance-from-body search algorithms are nowadays common,[11–14] albeit scalar. Given the distance from moving surfaces $\delta$, and the point on the surface closest to it $\mathbf{x}|\Gamma$, the mesh velocity at any field point is given by

$$\mathbf{w} = \mathbf{w}(\mathbf{x}|_\Gamma) f(\delta). \tag{3}$$

CCC 1069–8299/96/100599–10
© 1996 by John Wiley & Sons, Ltd.
*Received 24 October 1995*
*Revised 11 March 1996*

The function $f(\delta)$ assumes the value of unity for $\delta = 0$, and decays to zero as $\delta$ increases. This makes the procedure somewhat restrictive for general use, particularly if several moving bodies are present in the flowfield. On the other hand, the procedure is extremely fast if the initial distance $\delta$ can be employed for all times.[6,8]

In the second case, the edges of the unstructured grid are treated as springs that are relaxed in time to achieve equilibrium. In this way, a uniform element distribution is maintained. Starting from the prescribed boundary velocities, a new set of boundary coordinates is obtained at the new time step:

$$\mathbf{x}^{n+1}|_\Gamma = \mathbf{x}^n|_\Gamma + \Delta t\, \mathbf{w}|_\Gamma. \tag{4}$$

Based on these new values for the coordinates of the boundary points, the mesh is smoothed. Although more sophisticated mesh smoothing techniques have been proposed,[10] by far the most common way to smooth this new mesh is via spring analogy relaxation.[3,7] The force exerted by each spring (edge) is a function of its length and acts along its direction. Therefore, the sum of the forces exerted by all springs surrounding a point can be written as

$$\mathbf{f}_i = \sum_{j=1}^{ns_i} c(|\mathbf{x}_j - \mathbf{x}_i|)(\mathbf{x}_j - \mathbf{x}_i), \tag{5}$$

where $c$ denotes the spring function, $\mathbf{x}_i$ the coordinates of the point, and the sum extends over all the points surrounding the point. At the surface of the computational domain, no movement of points is allowed, i.e. $\Delta \mathbf{x} = 0$. The new values for the coordinates are obtained iteratively via a relaxation or conjugate gradient scheme.[3,7] Once the new coordinates have been evaluated, the mesh velocity is computed from

$$\mathbf{w} = \frac{1}{\Delta t}(\mathbf{x}^{n+1} - \mathbf{x}^n). \tag{6}$$

Most of the potential problems that may occur for this type of mesh velocity smoothing are due to initial grids that have not been smoothed. For such cases, the velocity of the moving boundaries is superposed to a ficticious mesh smoothing velocity which may be quite large during the initial stages of a run. Moreover, for spring analogy smoothers there is no guarantee that negative elements will not appear.

In the third case, the mesh velocity is smoothed directly, based on the exterior boundary conditions given by (1), (2). The aim, as stated before, is to obtain a mesh velocity field $\mathbf{w}$ in such a way that element distortion is minimized. Consider for the moment the 1-D situation sketched in Figure 1. At the left end of the domain, the mesh velocity is prescribed to be $w_0$. At

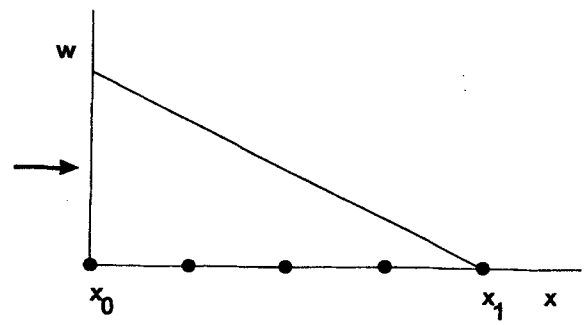


Figure 1. Mesh velocity smoothing in 1-D

the right end, the mesh velocity vanishes. If the mesh velocity decreases *linearly*, i.e.

$$\frac{\partial w}{\partial x} = g_v, \tag{7}$$

then the elements will maintain their initial size ratios. This is because for any two elements the change in size $\delta h$ during one time step is given by

$$\delta h = (w_2 - w_1)\Delta t = \Delta w\, \Delta t. \tag{8}$$

This implies that for the size ratio of any two elements $i, j$ we obtain

$$\left.\frac{h_i}{h_j}\right|^{n+1} = \frac{h_i|^n + \Delta w_i \Delta t}{h_j|^n + \Delta w_j \Delta t} = \frac{h_i|^n + g_v h_i|^n \Delta t}{h_j|^n + g_v h_j|^n \Delta t} = \left.\frac{h_i}{h_j}\right|^n, \tag{7}$$

i.e. all elements in the regions where mesh velocity is present will be deformed in roughly the same way. Solutions with constant gradients are reminiscent of Laplacian operators, and indeed, for the general case, the mesh velocity may be obtained by solving

$$\nabla \mathbf{k} \nabla \mathbf{w} = 0, \tag{10}$$

with the Dirichlet boundary conditions given by (1), (2). This system is discretized using finite element procedures. The resulting system of equations can be solved in a variety of ways, e.g. via relaxation as

$$C^{ii}\, \Delta \mathbf{w}^i = -\Delta t\, K^{ij}(\mathbf{w}^i - \mathbf{w}^j), \tag{11}$$

where

$$C^{ii} = \sum_{i \neq j} |K^{ij}|, \tag{12}$$

and the optimal $\Delta t$-sequence is given by

$$\Delta t^i = \frac{1}{1 + \cos\left[\dfrac{\pi(i-1)}{n}\right]}, \qquad i = 1, n. \tag{13}$$

If the diffusion coefficient appearing in (10) is set to $k=1$, a true Laplacian velocity smoothing is obtained. This yields the most 'uniform deformation' of elements, and therefore minimizes the number of remeshings or remappings required. Alternatively, for element-based codes, one may approximate the Laplacian coefficients $K^{ij}$ in (11) by

$$\nabla^2 \mathbf{w} \approx -(\mathbf{M}_l - \mathbf{M}_c)\mathbf{w}, \tag{14}$$

where $\mathbf{M}_l$, $\mathbf{M}_c$ denote, respectively, the lumped and consistent mass matrices.This approximation is considerably faster for element-based codes (for edge-based codes there is no difference in speed between the true Laplacian and this expression), but it is equivalent to a diffusion coefficient $k = h^2$. This implies that the gradient of the mesh velocity field will be larger for smaller elements. These will therefore distort at a faster rate than the larger elements. Obviously, for uniform grids this is not a problem, but in many cases the smallest elements are close to the surfaces that move, prompting many remeshings.

Based on the previous arguments, one may also consider a diffusion coefficient of the form $k = h^{-p}$, $p > 0$. In this case, the gradient of the mesh velocity field will be larger for the larger

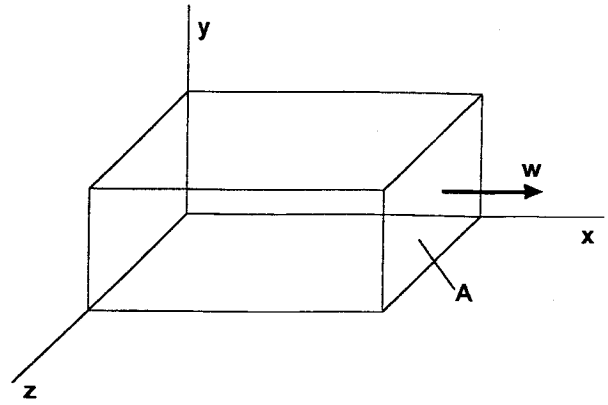


Figure 2. 1-D movement of face A

elements. The larger elements will therefore distort at a faster rate than the smaller ones – a desirable feature for many applications.

To see more clearly the difference between mesh velocity and coordinate smoothers, consider the simple box shown in Figure 2. Suppose that face A is being moved in the $x$-direction. For the case of coordinate smoothing, there will, in all likelihood, appear mesh velocities in the $y$- and $z$-directions. This is because, as the mesh moves, the smoothing technique will result in displacements of points in the $y$- and $z$-directions, and hence velocities in the $y$- and $z$-directions. On the other hand, for the case of mesh velocity smoothing, only displacements in the $x$-direction will appear. This is because the Dirichlet boundary conditions given by (1), (2) do not allow any mesh velocity other than in the $x$-direction to appear. We consider this an advantage of mesh velocity smoothers, and have therefore pursued them from the outset.

## 2. VARIABLE DIFFUSIVITY LAPLACIAN SMOOTHING

In most practical applications, the relevant flow phenomena and associated gradients of density, velocity and pressure are on or close to the bodies immersed in the fluid. Hence, the smallest elements are typically encountered close to the bodies. A straightforward Laplacian smoothing of the mesh velocities will tend to distort the elements in these critical regions. Thus, the small elements in the most critical regions tend to be the most deformed, leading to a loss in accuracy and possible reinterpolation errors due to the high rate of remeshings required. In an attempt to mitigate this shortcoming, we propose a diffusion coefficient $k$ that is based on the distance $\delta$ from the moving bodies. In general, $k$ should be a function of $\delta$ as sketched in Figure 3. For small $\delta$, $k$ should be large, leading to a small gradient of $\mathbf{w}$, i.e. nearly constant mesh velocity close to the moving bodies. For large $\delta$, $k$ should tend to unity in order to ensure the most uniform deformation of the (larger) elements that are away from the bodies.

### *2.1. Distance evaluation*

The calculation of the distance $\delta$ can be carried out in a variety of ways. Scalar, optimal search procedures have been employed within grid generation and turbulence modelling,[11–14] but for purposes of parallelization we prefer the Laplacian-based distance evaluation detailed

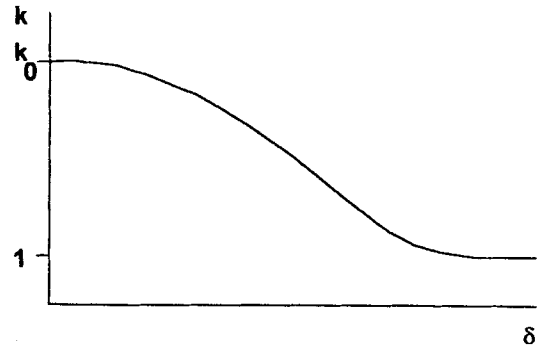


Figure 3. General shape for desired diffusivity $k$

here. Consider the 1-D Poisson problem:

$$\delta_{,xx} = -s; \qquad \delta(0) = 0; \qquad \delta_{,x}|_{x_1} = 0. \tag{15}$$

The exact solution is given by

$$\delta = sx\left(x_1 - \frac{x}{2}\right), \tag{16}$$

implying

$$x_1 = \sqrt{\frac{2\delta_1}{s}}; \qquad \delta_{,x}|_0 = \sqrt{2s\delta_1}. \tag{17}$$

This means that in order to obtain a unit gradient at $x=0$, one should choose $s=1/2\delta_1$, which in turn leads to $x_1 = 2\delta_1$. Another way to interpret the results is that the 'rigidization' distance $x_1$ is related to the maximum value of $\delta = \delta_1$ and the source strength $s$.

This simplified analysis is not valid for 2-D and 3-D solutions of the general Poisson problem

$$\nabla^2\delta = -s; \qquad \delta|_{\Gamma_0} = 0; \qquad \delta_{,n}|_{\Gamma_1} = 0, \tag{18}$$

where for radial symmetry the solutions contain $\ln(r)$ and $1/r$ terms. On the other hand, we do not require the exact distance from moving bodies, but only a distance function that will give the proper behaviour for $k$. We therefore use (18) to determine the distance function $\delta$. The Poisson problem is solved using finite element procedures and an iterative procedure similar to that given by (11)–(13) above. This fits naturally into existing CFD codes, where edge-based Laplacian operator modules exist for artificial or viscous dissipation terms. The Neumann condition in (18) is enforced by not allowing $\delta$ to exceed a certain value. After each iterative pass during the solution of (18), we impose

$$\delta \leqslant \delta_1, \tag{19}$$

which in effect produces the desired Neumann boundary condition at $\Gamma$.

### 2.2. *Diffusivity as a function of distance*

As stated before, for small $\delta$, $k$ should be large, leading to a very small gradient of $\mathbf{w}$, i.e. nearly constant mesh velocity close to the moving bodies. For large $\delta$, $k$ should tend to unity in

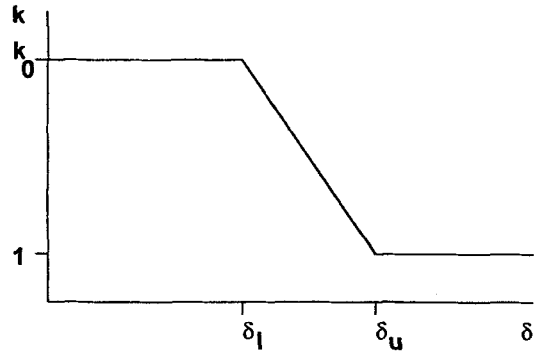


Figure 4. Diffusivity as a function of distance

order to ensure the most uniform deformation of the (larger) elements that are away from the bodies. For the diffusion $k$, we use the following constant-linear-constant function (see Figure 4):

$$k = k_0 + (1 - k_0)\max\left(0, \min\left(1, \frac{\delta - \delta_l}{\delta_u - \delta_l}\right)\right). \tag{20}$$

The choice of the cut-off distances is, in principle, arbitrary. We have found $\delta_l = x_1/4$, $\delta_u = x_1/2$ to be a good choice.

## 3. EXAMPLES

The procedure outlined above has been used extensively within an unstructured-grid, edge-based ALE CFD code.[9] The usefulness of changing $k$ according to the distance as given by (20) is demonstrated on a moving wing as well as a hypersonic store release case computed recently.

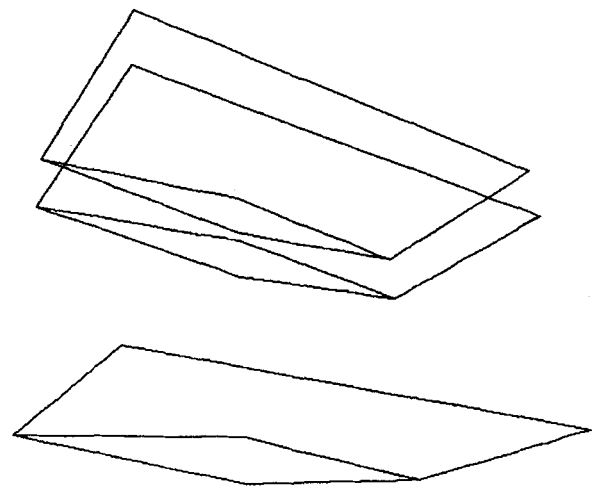

Figure 5. Outline of wind after 0, 30 and 100 time steps

### 3.1. Wing

A moving wing is first considered using both Laplacian ($k_0 = 1$, $x_1 = 0$) and modified Laplacian ($k_0 = 100$, $x_1 = 2$) velocity smoothing. Figure 5 shows the outline of the wing after 0, 30 and 100 time steps. The surface grids and mesh velocities obtained using the two methods after 30 time steps are shown in Figures 6 and 7. As a result of the larger gradient of the mesh velocity field, the meshes in the vicinity of the wing start becoming distorted in the case of Laplacian velocity smoothing, and the first negative element appears after 34 time steps. On the other hand, for the modified Laplacian velocity smoothing the meshes in the vicinity of the wing remain undistorted, and no remeshing is required even after 100 time

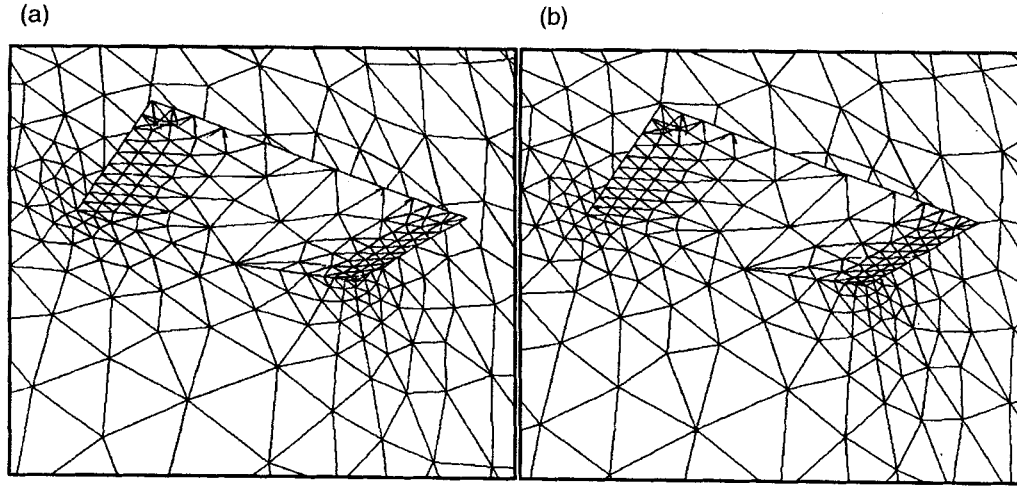


Figure 6. Surface mesh after 30 time steps: (a) Laplacian velocity smoothing; (b) modified Laplacian velocity smoothing

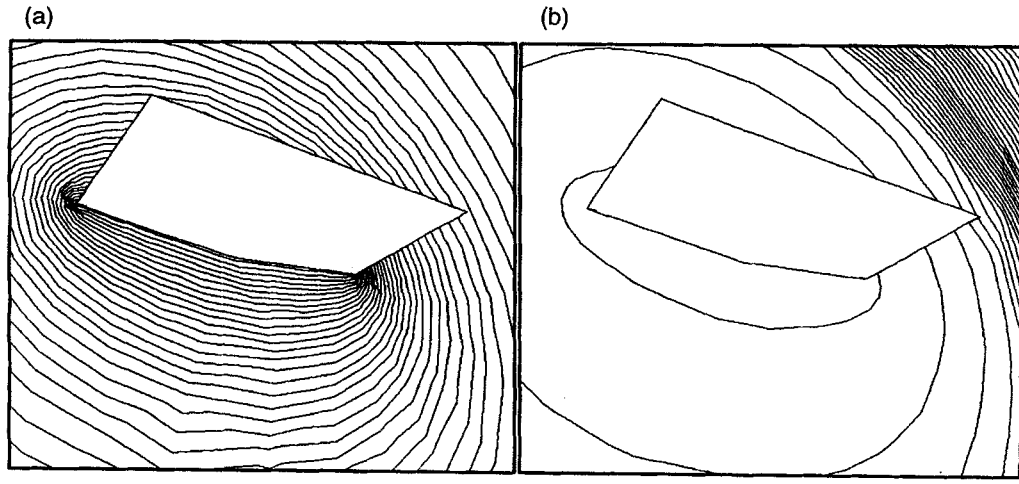


Figure 7. Surface mesh velocity after 30 time steps: (a) Laplacian velocity smoothing ($\Delta \mid v \mid = 0{\cdot}05$); (b) modified Laplacian velocity smoothing ($\Delta \mid v \mid = 0{\cdot}05$)

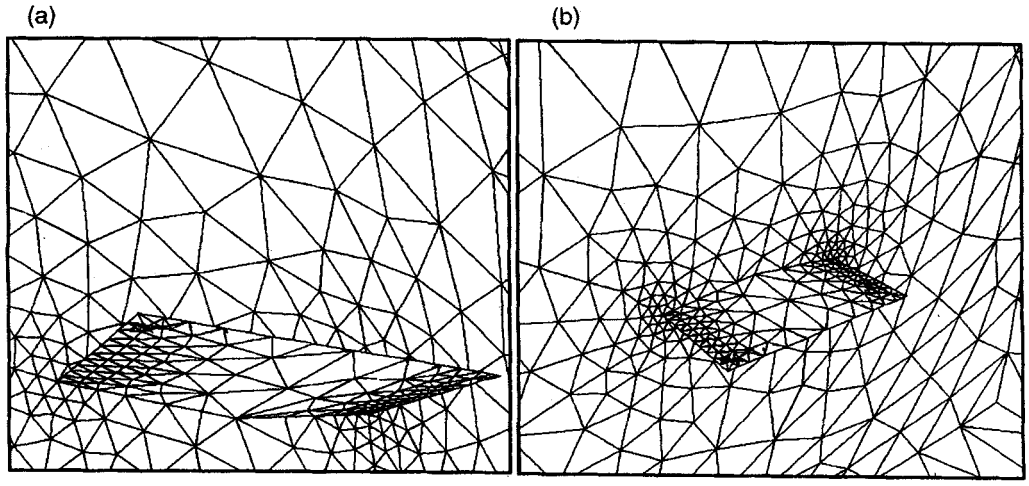


Figure 8. Surface mesh after 100 time steps (modified Laplacian velocity smoothing)

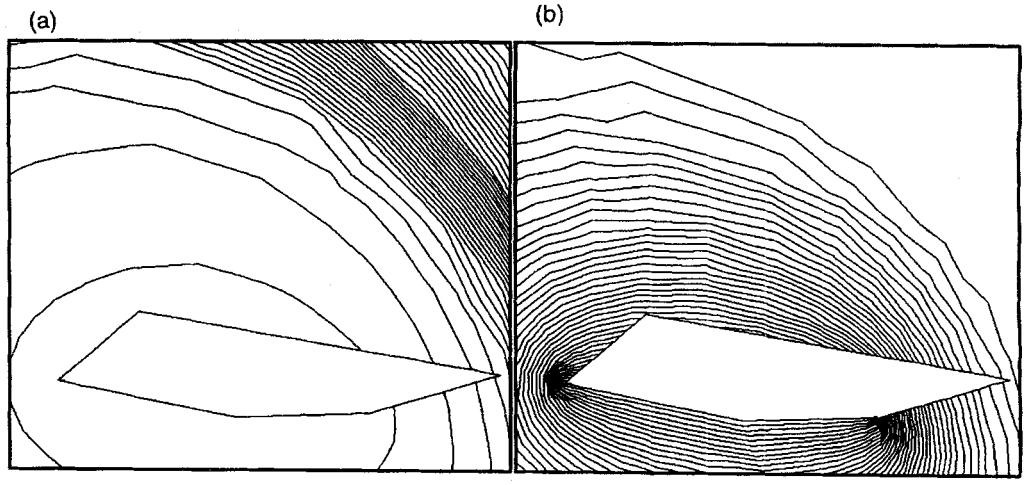


Figure 9. (a) Surface mesh velocity after 100 time steps ($\Delta \mid v \mid = 0{\cdot}05$); (b) distance function ($\Delta\delta = 0{\cdot}05$); (modified Laplacian velocity smoothing)

steps. The surface grid and mesh velocity, as well as the distance function after 100 time steps are shown in Figures 8 and 9, respectively.

### *3.2. Hypersonic store release*

As a second, more realistic case, we consider a hypersonic store release. From a given state, we followed the solution for 100 time steps, setting in the first case $k_0 = 1$, $x_1 = 0$, and in the second case $k_0 = 50$, $x_1 = 0{\cdot}08$. The surface grids and mesh velocities obtained after 100 time steps are displayed in Figures 10 and 11. As one can see, the new variable $k$ mesh velocity smoothing leads to a much less deformed grid close to the moving missile. For this case, the

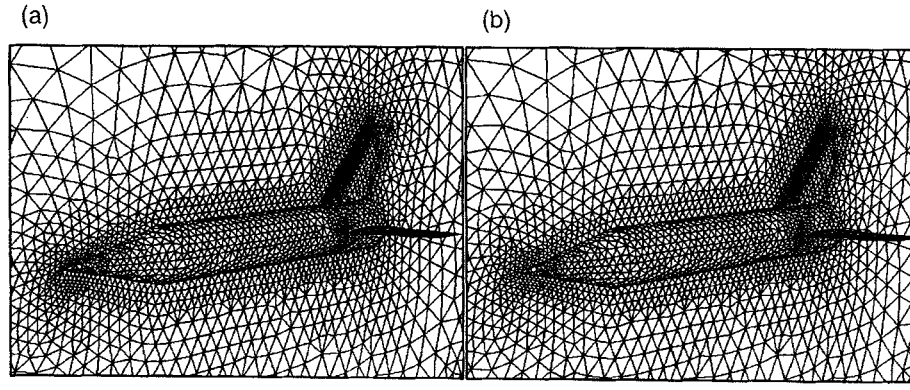


Figure 10. Surface mesh after 100 time steps: (a) Laplacian velocity smoothing; (b) modified Laplacian velocity smoothing

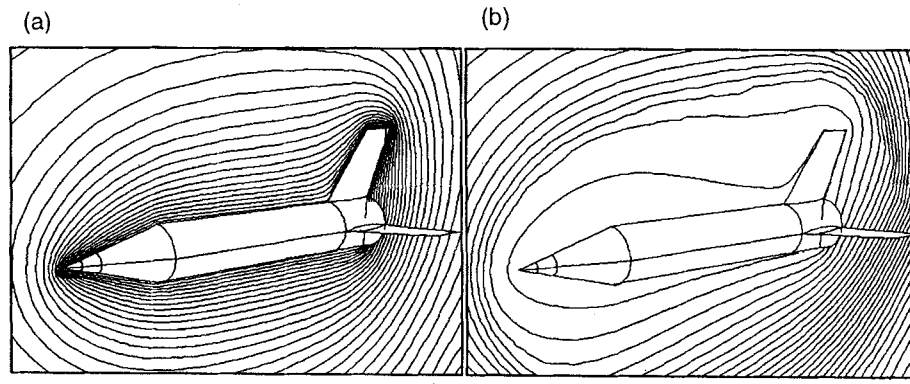


Figure 11. Surface mesh velocity after 100 time steps: (a) Laplacian velocity smoothing; (b) modified Laplacian velocity smoothing

number of local remeshings required dropped by a factor of 1:4, leading to considerable CPU savings in a multiprocessor environment.

## 4. CONCLUSIONS

A Laplacian smoothing of the mesh velocities with variable diffusivity based on the distance from moving bodies has been found effective for unstructured-grid ALE solvers. The variable diffusivity enforces a more uniform mesh velocity in the region close to the moving bodies. Given that in most applications these are regions where small elements are located, the new procedure decreases element distortion considerably, reducing the need for local or global remeshing, and in some cases avoiding it altogether.

As the mesh movement is linked to a time-stepping algorithm for the fluid part, and the body movement occurs at a slow pace compared to the other wavespeeds in the coupled fluid/solid system, normally no more than five steps are required to smooth the velocity field sufficiently, i.e. $3 \leqslant n \leqslant 5$ in (11). The overhead incurred by this type of smoothing is very small compared to the overall costs for any ALE-type methodology for Euler or Navier–Stokes flow solvers.

ACKNOWLEDGEMENTS

This work was partially funded by AFOSR under contract F496209410119, with Dr. Leonidas Sakell as the technical monitor.

REFERENCES

1. L. Formaggia, J. Peraire and K. Morgan, 'Simulation of a store separation using the finite element method', *Appl. Math. Model.*, **12**, 175–181 (1988).
2. R. Löhner, 'An adaptive finite element solver for transient problems with moving bodies', *Comput. Struct.*, **30**, 303–317 (1988).
3. J. T. Batina, 'Unsteady Euler airfoil solutions using unstructured dynamic meshes', *AIAA J.*, **28**(8), 1381–1388 (1990).
4. R. Löhner, 'Three-dimensional fluid-structure interaction using a finite element solver and adaptive remeshing', *Comput. Syst. Eng.*, **1**(2–4), 257–272 (1990).
5. J. D. Baum and R. Löhner, 'Numerical simulation of pilot/seat ejection from an F-16', *AIAA-93-0783*, (1993).
6. A. H. Boschitsch and T. R. Quackenbush, 'High accuracy computations of fluid-structure interaction in transonic cascades', *AIAA-93-0485*, (1993).
7. R. D. Rausch, J. T. Batina and H. T. Y. Yang, 'Three-dimensional time-marching aeroelastic analyses using an unstructured-grid Euler method', *AIAA J.*, **31**(9), 1626–1633 (1993).
8. G. A. Davis and O. O. Bendiksen, 'Unsteady transonic two-dimensional Euler solutions using finite elements', *AIAA J.*, **31**, 1051–1059 (1993).
9. J. D. Baum, H. Luo and R. Löhner, 'A new ALE adaptive unstructured methodology for the simulation of moving bodies', *AIAA-94-0414*, (1994).
10. V. Venkatakrishnan and D. J. Mavriplis, 'Implicit method for the computation of unsteady flows on unstructured grids', *AIAA-95-1705-CP*, (1995).
11. P. Rostand, 'Algebraic turbulence models for the computation of 2-D high speed flows using unstructured grids', *ICASE Rep. 88-63*, (1988).
12. R. Löhner, 'Some useful data structures for the generation of unstructured grids', *Commun. Appl. Numer. Methods*, **4**, 123–135 (1988).
13. J. Bonet and J. Peraire, 'An alternate digital tree algorithm for geometric searching and intersection problems', *Int. J. Numer. Methods Eng.*, **31**, 1–17 (1991).
14. D. Martin and R. Löhner, 'An implicit linelet-based solver for incompressible flows', *AIAA-92-0668*, (1992).