INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2004; **60**:641–660 (DOI: 10.1002/nme.978)

Adaptive embedded unstructured grid methods

Rainald Löhner[1,*,†], Joseph D. Baum[2], Eric Mestreau[2], Dmitri Sharov[2], Charles Charman[3] and Daniele Pelessone[4]

[1]*School of Computational Sciences, MS 4C7, George Mason University, Fairfax, VA 22030-4444, U.S.A.*
[2]*Advanced Technology Group, Science Applications Int. Corp., McLean, VA, U.S.A.*
[3]*General Atomics, San Diego, CA, U.S.A.*
[4]*ES3, Solana Beach, CA, U.S.A.*

SUMMARY

A simple embedded domain method for node-based unstructured grid solvers is presented. The key modification of the original, edge-based solver is to remove all geometry-parameters (essentially the normals) belonging to edges cut by embedded surface faces. Several techniques to improve the treatment of boundary points close to the immersed surfaces are explored. Alternatively, higher-order boundary conditions are achieved by duplicating crossed edges and their endpoints. Adaptive mesh refinement based on proximity to or the curvature of the embedded CSD surfaces is used to enhance the accuracy of the solution. User-defined or automatic deactivation for the regions inside immersed solid bodies is employed to avoid unnecessary work. Several examples are included that show the viability of this approach for inviscid and viscous, compressible and incompressible, steady and unsteady flows, as well as coupled fluid–structure problems. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: embedded grids; CFD; finite element methods

1. INTRODUCTION

The numerical solution of partial differential equations (PDEs) is usually accomplished by performing a spatial and temporal discretization with subsequent solution of a large algebraic system of equations [1]. The spatial discretization is commonly performed via polyhedra, also called (finite) volumes or elements. The final assembly of these polyhedra yields the so-called

*Correspondence to: Rainald Löhner, School of Computational Sciences, MS 4C7, George Mason University, 4400 University Drive, Fairfax, VA 22030-4444, U.S.A.
†E-mail: rlohner@gmu.edu

Contract/grant sponsor: DTRA
Contract/grant sponsor: AFOSR

Received 10 April 2003
Revised 20 June 2003
Accepted 18 August 2003

mesh. The transition from an arbitrary surface description to a proper mesh still represents a difficult task [2, 3]. Considering the rapid advance of computer power, together with the perceived maturity of field solvers, an automatic transition from arbitrary surface description to mesh becomes mandatory.

Two types of grids are most commonly used: body-conforming and embedded. For body-conforming grids the external mesh faces match up with the surface (body surfaces, external surfaces, etc.) of the domain. This is not the case for the embedded approach (also known as ficticious domain, immersed boundary or Cartesian method), where the surface is placed inside a large mesh (typically a regular parallelepiped), with special treatment of the elements close to the surfaces.

Considering the general case of moving or deforming surfaces with topology change, both approaches have complementary strengths and weaknesses:

(a) *Body-conforming moving meshes*: the PDEs describing the flow need to be cast in an arbitrary Lagrangean–Eulerian (ALE) frame of reference, the mesh is moved in such a way as to minimize distortion, if required the topology is reconstructed, the mesh is regenerated and the solution reinterpolated. All of these steps have been optimized over the course of the last decade, and this approach has been used extensively [4–7]. The body-conforming solution strategy exhibits the following shortcomings: the topology reconstruction can sometimes fail for singular surface points; there is no way to remove subgrid features from surfaces, leading to small elements due to geometry; reliable parallel performance beyond 16 processors has proven elusive for most general-purpose grid generators; the interpolation required between grids invariably leads to some loss of information; and there is an extra cost associated with the recalculation of geometry, wall-distances and mesh velocities as the mesh deforms.

(b) *Embedded fixed meshes*: the mesh is not body-conforming and does not move. Hence, the PDEs describing the flow can be left in the simpler Eulerian frame of reference. At every timestep, the edges crossed by CSD faces are identified and proper boundary conditions are applied in their vicinity. While used extensively [8–18] this solution strategy also exhibits some shortcomings: the boundary, which has the most profound influence on the ensuing physics, is also the place where the worst elements are found; at the same time, near the boundary, the embedding boundary conditions need to be applied, reducing the local order of approximation for the PDE; no stretched elements can be introduced to resolve boundary layers; adaptivity is essential for most cases; and there is an extra cost associated with the recalculation of geometry (when adapting) and the crossed edge information.

The development of the present embedded, adaptive fixed mesh capability was prompted by the inability of Computational Structural Dynamics (CSD) codes to ensure strict no-penetration during contact. Several blast–ship interaction simulations revealed that the amount of twisted metal was so considerable that any enforcement of strict no-penetration (required for consisted topology reconstruction) became impossible. Hence, at present the embedded approach represents the only viable solution.

It may appear somewhat contradictory to use an unstructured (tetrahedral) solver in conjunction with surface embedding. Most of the work carried out to date was in conjunction with Cartesian solvers [8–10, 12–17, 19], the argument being that flux evaluations could be optimized due to co-ordinate alignment. However, the achievable gains of such co-ordinate alignment may be limited due to the following mitigating factors:

(a) For most of the high resolution schemes the cost of limiting and the approximate Riemann solver far outweigh the cost of the few scalar products required for arbitrary edge orientation;
(b) The fact that any of these schemes (Cartesian, unstructured) requires mesh adaptation in order to be successful immediately implies the use of indirect addressing; given current trends in microchip design, indirect addressing, present in both types of solvers, may outweigh all other factors;
(c) Three specialized (x, y, z) edge-loops versus one general edge-loop, and the associated data reorganization implies an increase in software maintenance costs.

For a tetrahedral based solver, surface embedding represents just another addition in a toolbox of mesh handling techniques (mesh movement, overlapping grids, remeshing, h-refinement, etc.).

The remainder of the paper is organized as follows: Section 2 describes in general terms the treatment of embedded surfaces. Section 3 details the techniques used to mask edges crossed by surface faces, as well as the points close to it. The attention then turns to the changes required in the flow code to treat embedded surfaces (Sections 4–7). Adaptive refinement is considered in Section 8, the transfer of loads and fluxes in Section 9, and the treatment of gaps or cracks in Section 10. Enhancements for visualization of results are mentioned in Section 11, and numerical examples are presented in Section 12. Finally, some conclusions and an outlook for future developments are given in Section 13.

In what follows, we denote by CSD faces the surface of the computational domain that is embedded. We implicitly assume that this information is given by a triangulation, which typically is obtained from a CAD package via STL files, remote sensing data or from a CSD code in coupled fluid–structure applications.

2. TREATMENT OF EMBEDDED SURFACES

Two basic approaches have been proposed to modify field solvers in order to accommodate embedded surfaces: force- and kinematics-based. The first type applies an *equivalent balancing force* to the flowfield in order to achieve the kinematic boundary required at the embedded surface [18, 20]. The second approach, followed here, is to apply *kinematic boundary conditions* at the nodes close to the embedded surface. Depending on the required order of accuracy and simplicity, a first or second-order (higher-order) scheme may be chosen to apply the kinematic boundary conditions. Figure 1 illustrates the basic difference between these approaches.

A first-order scheme can be achieved by:

- Eliminating the edges crossing the embedded surface;
- Forming boundary coefficients to achieve flux balance;
- Applying boundary conditions for the end-points of the crossed edges based on the normals of the embedded surface.

A second-order scheme can be achieved by:

- Duplicating the edges crossing the embedded surface;
- Duplicating the end-points of crossed edges;

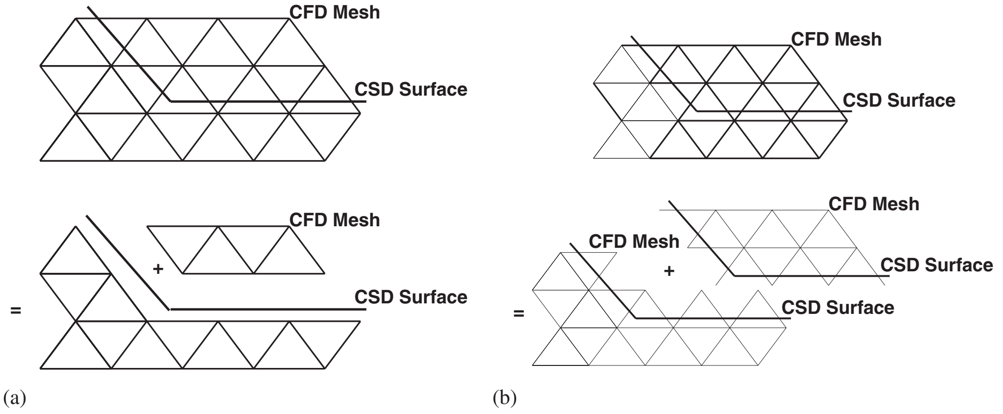


Figure 1. (a) First order treatment of embedded surfaces; and (b) second order treatment of embedded surfaces.

- Applying boundary conditions for the end-points of the crossed edges based on the normals of the embedded surface.

We note that in either case CFD edges crossed by CSD faces are modified/duplicated. Given that an edge/face crossing is essentially the same in 2-D and 3-D, these schemes are rather general.

3. DETERMINATION OF CROSSED EDGES

Given the CSD triangulation and the CFD mesh, the first step is to find the CFD edges cut by CSD faces. This is performed by building first an octree of the CSD faces. Then, a (parallel) loop is performed over the edges. For each edge, the bounding box of the edge is built. From the octree, all the faces in the region of the bounding box are found. This is followed by an in-depth test to determine which faces cross the given edge. The crossing face closest to each of the edge end-nodes is stored. This allows to resolve cases of thin gaps or cusps. Once the faces crossing edges are found, the closest face to the end-points of crossed edges is also stored. This allows to apply boundary conditions for the points close to the embedded surface. For transient problems, the procedure described above can be improved considerably. The key assumption is that the CSD triangulation will not move over more than 1–2 elements during a time step. If the topology of the CSD triangulation has not changed, the crossed-edge information from the previous time step can be re-checked. The points of edges no longer crossed by a face crossing them in the previous time step are marked, and the neighbouring edges are checked for crossing. If the topology of the CSD triangulation has changed, the crossed-edge information from the previous time step is no longer valid. However, the points close to cut edges in the previous time step can be used to mark 1–2 layers of edges. Only these edges are then re-checked for crossing.

4. FIRST ORDER TREATMENT

The first order scheme is the simplest to implement. Given the CSD triangulation and the CFD mesh, the CFD edges cut by CSD faces are found and deactivated. Considering an arbitrary field point i, the time-advancement of the unknowns $\mathbf{u}^i$ for an explicit edge-based time integration scheme [1] is given by

$$M^i \Delta \mathbf{u}^i = \Delta t \sum_{ij\Omega} C^{ij}(F_i + F_j) \tag{1}$$

Here C, F, M denote, respectively, the edge-coefficients, fluxes and mass-matrix. For any edge ij crossed by a CSD face, the coefficients C^{ij} are set to zero. This implies that for a uniform state $\mathbf{u} = const.$ the balance of fluxes for interior points with cut edges will not vanish. This is remedied by defining a new boundary point to impose total/normal velocities, as well as adding a 'boundary contribution', resulting in

$$M^i \Delta \mathbf{u}^i = \Delta t \left[\sum_{ij\Omega} C^{ij}(F_i + F_j) + C^i_\Gamma F_i \right] \tag{2}$$

The point-coefficients C^i_Γ are obtained from the condition that $\Delta \mathbf{u} = 0$ for $\mathbf{u} = const.$ Given that gradients (e.g. for limiting) are also constructed using a loop of the form given by Equation (1) as:

$$M^i \mathbf{g}^i = \sum_{ij\Omega} C^{ij}(u_i + u_j) \tag{3}$$

it would be desirable to build the C^i_Γ coefficients in such a way that the constant gradient of a linear function u can be obtained exactly. However, this is not possible, as the number of coefficients is too small. Therefore, the gradients at the boundary are either set to zero or extrapolated from the interior of the domain.

The mass-matrix M^i of points surrounded by cut edges must be modified to reflect the reduced volume due to cut elements. Again, the simplest possible modification of M^i is used. In a pass over the edges, the smallest 'cut edge fraction' ξ for all the edges surrounding a point is found. The modified mass-matrix is then given by

$$M^i_* = \frac{1 + \xi_{\min}}{2} M^i \tag{4}$$

Note that the value of the modified mass-matrix can never fall below half its original value, implying that timestep sizes will always be acceptable.

4.1. Boundary conditions

For the new boundary points belonging to cut edges the proper PDE boundary conditions are required. In the case of flow solvers, these are either an imposed velocity or an imposed normal velocity. For limiting and higher-order schemes, one may also have to impose boundary conditions on the gradients. The required surface normal and boundary velocity are obtained directly from the closest CSD face to each of the new boundary points.

These low-order boundary conditions may be improved by extrapolating the velocity from the surface with field information. The location where the flow velocity is equal to the surface

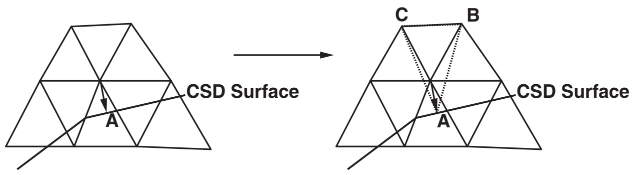


Figure 2. Extrapolation of velocity.

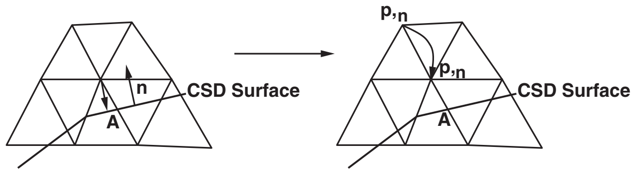


Figure 3. Extrapolation of normal pressure gradient.

velocity is the surface itself, and not the closest boundary point. As shown in Figure 2, for each boundary point the closest point on the CSD face is found. Then, two (three) neighbouring field (i.e. non-boundary) points are found and a triangular (tetrahedral) element that contains the boundary point is formed. The velocity imposed at the field point is then found by interpolation. In this way, the boundary velocity 'lags' the field velocities by one time step.

The normal gradients at the boundary points can be improved by considering the 'most aligned' field (i.e. non-boundary) point to the line formed by the boundary point and the closest point on the CSD face (see Figure 3).

5. HIGHER ORDER TREATMENT

As stated before, a higher-order treatment of embedded surfaces may be achieved by using ghost points or mirrored points to compute the contribution of the crossed edges to the overall solution. This approach presents the advantage of not requiring the modification of the mass matrix as all edges (even the crossed ones) are taken into consideration. It also does not require an extensive modification of the various solvers. On the other hand, it requires more memory due to duplication of crossed edges and points, as well as (scalar) CPU time for renumbering/reordering arrays. Particularly for moving body problems, this may represent a considerable CPU burden.

5.1. Boundary conditions

By duplicating the edges, the points are treated in the same way as in the original (non-embedded) case. The boundary conditions are imposed indirectly by mirroring and interpolating the unknowns as required. Figure 4 depicts the contribution due to the edges surrounding point i. A CSD boundary crosses the CFD domain. In this particular situation point j, which lies on the opposite side of the CSD face, will have to use the flow values of its mirror image j' based on the crossed CSD face.

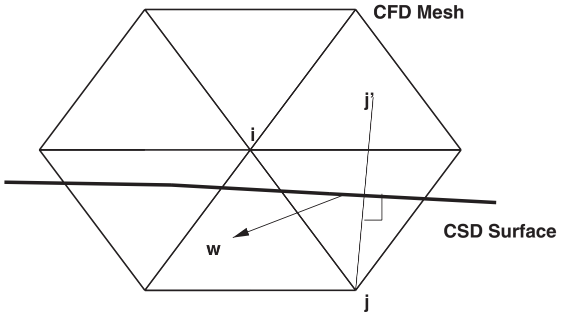


Figure 4. Higher order boundary conditions.

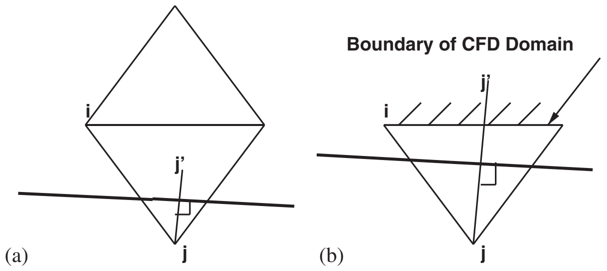


Figure 5. Problem cases.

The flow values of the mirrored point are then interpolated from the element the point resides in using the following formulation:

$$\begin{aligned} \rho_m &= \rho_i, \quad p_m = p_i \\ \mathbf{v}_m &= \mathbf{v}_i - 2[(\mathbf{v}_i - \mathbf{w}_{\text{csd}}) \cdot \mathbf{n}]\mathbf{n} \end{aligned} \tag{5}$$

where $\mathbf{w}_{\text{csd}}$ is the average velocity of the crossed CSD face, ρ the density, $\mathbf{v}$ the flow velocity and p the pressure. Proper handling of the interpolation is also required as the element used for the interpolation might either be crossed (Figure 5(a)) or not exist (Figure 5(b)).

A more accurate formulation of the mirrored pressure and density can also be used taking into account the local radius of curvature of the CSD wetted surface:

$$\begin{aligned} p_m &= p_i - \rho_i \frac{[\mathbf{v}_i - (\mathbf{v}_i - \mathbf{w}_{\text{csd}}) \cdot \mathbf{n}]^2}{R_i} \Delta \\ \rho_m &= \rho_i \left(\frac{p_m}{p_i} \right)^{1/\gamma} \end{aligned} \tag{6}$$

where R_i is the radius of curvature and Δ the distance between the point and its mirror image. This second formulation is more complex and requires the computation of the 2 radii (3D) of curvature at each CSD point. The radius of curvature plays an important role for large elements but this influence can be diminished by the use of automatic h-refinement.

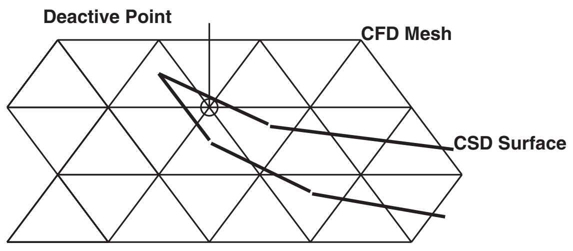


Figure 6. Deactive point.

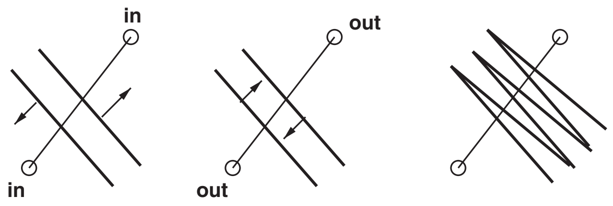


Figure 7. Edges with multiple crossing faces.

6. DEACTIVATION OF INTERIOR REGIONS

For highly distorted CSD surfaces, or for CSD surfaces with thin reentrant corners, all edges surrounding a given point may be crossed by CSD faces (see Figure 6). The best way to treat such points is to simply deactivate them [1, chapter 16].

This deactivation concept can be extended further in order to avoid unnecessary work for regions inside solid objects. Two approaches were pursued in this direction: seed points and automatic deactivation.

(a) *Seed points*: In this case, the user specifies a point inside an object. The closest CFD field point to this so-called seed point is then obtained. Starting from this point, additional points are added using an advancing front (nearest neighbour layer) algorithm, and flagged as inactive. The procedure stops once points that are attached to crossed edges have been reached.

(b) *Automatic deactivation*: For complex geometries with moving surfaces, the manual specification of seed points becomes impractical. An automatic way of determining which regions correspond to the flowfield one is trying to compute and which regions correspond to solid objects immersed in it is then required. The algorithm employed starts from the edges crossed by embedded surfaces. For the end-points of these edges an in/outside determination is attempted. This is non-trivial, particularly for thin or folded surfaces (Figure 7). A more reliable way to determine whether a point is in/outside the flowfield is obtained by storing, for the crossed edges, the faces closest to the end-points of the edge. Once this in/outside determination has been done for the end-points of crossed edges, the remaining points are marked using an advancing front algorithm. It is important to remark that in this case both the inside (active) and outside (deactive) points are marked at the same time. In the case of a conflict, preference is always given to mark the points as inside the flow domain (active). Once the points have been marked as active/inactive, the element and edge-groups required for vectorization are inspected

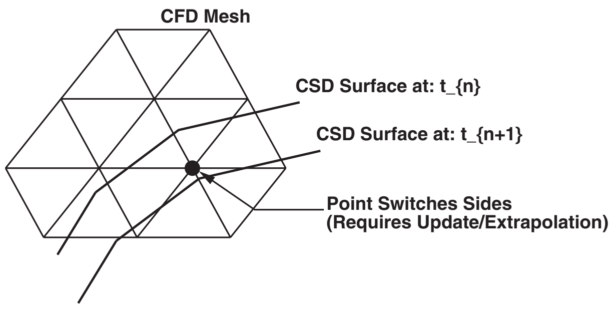


Figure 8. Extrapolation of solution.

in turn. As with spacemarching [21], the idea is to move the active/inactive if-tests to the element/edge-groups level in order to simplify and speed up the core flow solver.

7. EXTRAPOLATION OF THE SOLUTION

For problems with moving boundaries, mesh points can switch from one side of a surface to another (see Figure 8). For these cases, the solution must be extrapolated from the proper state. The conditions that have to be met for extrapolation are as follows:

(a) The edge was crossed at the previous timestep and is no longer crossed;
(b) The edge has one field point (the point donating unknowns) and one boundary point (the point receiving unknowns); and
(c) The CSD face associated with the boundary point is aligned with the edge.

8. ADAPTIVE MESH REFINEMENT

Adaptive mesh refinement is very often used to reduce CPU and memory requirements without compromising the accuracy of the numerical solution. For transient problems with moving discontinuities, adaptive mesh refinement has been shown to be an essential ingredient of production codes [6, 22, 23]. For embedded CSD triangulations, the mesh can be refined automatically close to the surfaces. This has been done in the present case by including two additional refinement indicators (on top of the usual ones based on the flow variables). The first one looks at the edges cut by CSD faces, and refines the mesh to a certain element size or refinement level. The second, more sophisticated indicator, looks at the surface curvature, and refines the mesh only in regions where the element size is deemed insufficient.

9. LOAD/FLUX TRANSFER

For fluid–structure interaction problems, the forces exerted by the fluid on the embedded surfaces need to be evaluated. This is done by computing first the stresses (pressure, shear stresses) in

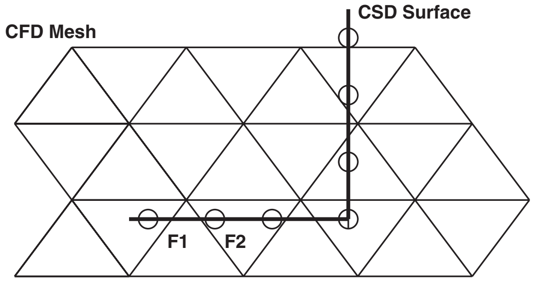


Figure 9. Transfer of stresses/fluxes.

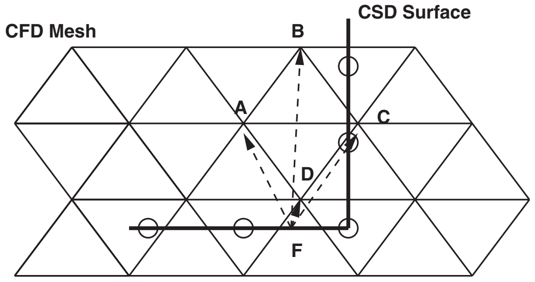


Figure 10. Transfer of stresses/fluxes.

the fluid domain, and then interpolating this information to the embedded surface triangles. In principle, the integration of forces can be done with an arbitrary number of Gauss-points per embedded surface triangle. In practice, one Gauss-point is used most of the time. The task is then to interpolate the stresses to the Gauss-points on the faces of the embedded surface. Given that the information of crossed edges is available, the immediate impulse would be to use this information to obtain the required information. However, this is not the best way to proceed, as

- The closest (end-point of crossed edge) point corresponds to a low-order solution and/or stress; i.e. it may be better to interpolate from a field point;
- The face may have no/multiple crossing edges (see Figure 9); i.e. there will be a need to construct extra information in any case.

For each Gauss-point required, the closest interpolating points are obtained with the following steps:

- Obtain a search region to find close points; this is typically of the size of the current face the Gauss-point belongs to, and is enlarged or reduced depending on the number of close points found;
- Obtain the close faces of the current surface face;
- Remove from the list of close points those that would cross close faces that are visible from the current face, and that can in turn see the current face (see Figure 10);

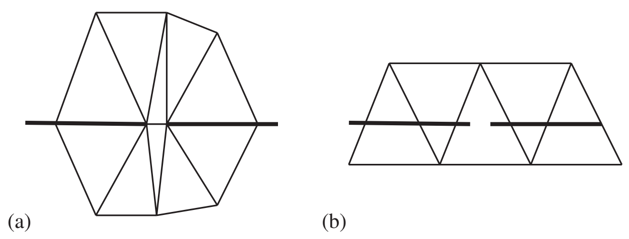


Figure 11. Treatment of gaps: (a) body fitted; and (b) embedded.

- Order the close points according to proximity and boundary/field point criteria;
- Retain the best n_{p} close points from the ordered list.

The close points and faces are obtained using octrees for the points and a modified octree for the faces.

10. TREATMENT OF GAPS OR CRACKS

The presence of 'thin regions' or gaps in the surface definition, or the appearance of cracks due to fluid-structure interaction has been an outstanding issue for a number of years. For body fitted grids (Figure 11(a)), a gap or crack is immediately associated with minuscule grid sizes, small timesteps and increased CPU costs. For embedded grids (Figure 11(b)), the gap or crack may not be seen. The solution proposed here is to allow some flow through the gap or crack without compromising the timestep. The key idea is to change the geometrical coefficients of crossed edges in the presence of gaps. Instead of setting these coefficients to zero, they are reduced by a factor that is proportional to the size of the gap δ to the average element size h in the region:

$$C_k^{ij} = \eta C_{0k}^{ij}; \quad \eta = \delta / h \tag{7}$$

Gaps are detected by considering the edges of elements with multiple crossed edges. If the faces crossing these edges are different, a test if performed to see if one face can be reached by the other via a near-neighbour search. If this search is successful, the CSD surface is considered watertight. If the search is not successful, the gap size δ is determined, and the edges are marked for modification.

11. COORDINATE MOVEMENT FOR DISPLAY

The display of information from a CFD field solver with embedded CSD faces requires some attention. The easiest way to visualize the location of the CSD surface is by removing the elements that contain the embedded surface, yielding cut-outs close to embedded surfaces that have a staircase boundary. In order to achieve a more precise, continuous surface representation, the points close to embedded surfaces are moved to the surface itself before display. This may produce some distortion in the contour lines close to the embedded surfaces, but produces a more faithful geometry representation. Close to corners or ridges multiple surface normals will appear. For each of these multiple normals, a separate direction of movement is determined. The final point movement is obtained as the sum of all of these.

12. EXAMPLES

12.1. Sod shock tube

The embedded CSD technique is demonstrated by comparing the results on the Sod shock-tube problem ($\rho_1 = p_1 = 1.0$, $\rho_2 = p_2 = 0.1$) for a 'clean-wall', body fitted mesh and an equivalent embedded CSD mesh. The flow is treated as compressible and inviscid, with no-penetration (slip) boundary conditions for the velocities at the walls.

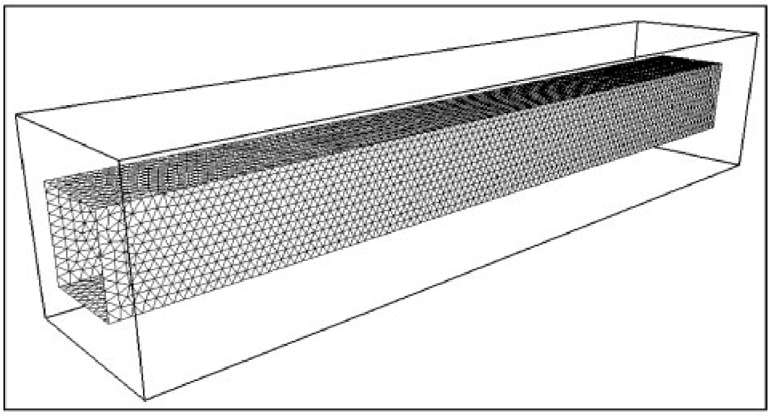
The embedded geometry can be discerned from Figure 12(a). Figure 12(b) shows the results for the two techniques. Although the embedded technique is rather primitive, the results are surprisingly good. The main difference is slightly more noise in the contact discontinuity region, which may be expected, as this is a linear discontinuity. The long-term effects on the solution for the different treatments of boundary points can be seen in Figure 12(c), which shows the pressure time history for a point located in the high pressure side (left of membrane). Both ends of the shock tube are assumed closed. One can see the different reflections. In particular, the curves for the boundary fitted approach and the second-order (ghost-point) embedded approach are almost identical, whereas the first-order embedded approach exhibits pronounced damping.

12.2. Shuttle ascend configuration

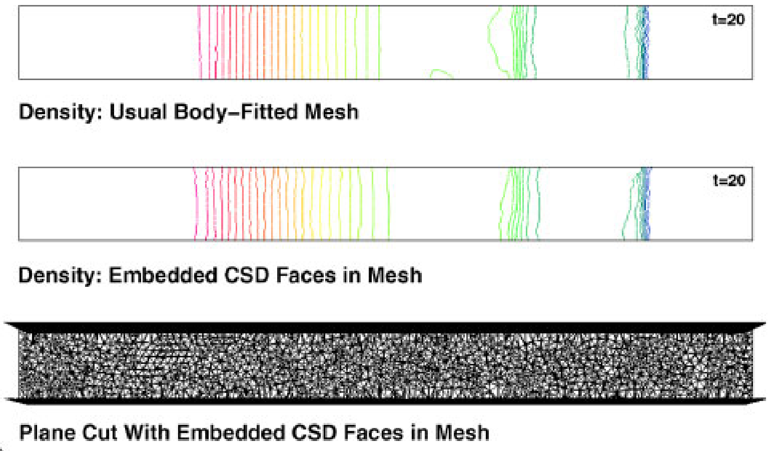
The second example considered is the space shuttle ascend configuration shown in Figure 13(a). The external flow is at $Ma = 2$ and angle of attack $\alpha = 5°$. As before, the flow is treated as compressible and inviscid, with no-penetration (slip) boundary conditions for the velocities at the walls. The surface definition consisted of approximately 161 Ktria faces. The base CFD mesh had approximately 1.1 Mtet. For the geometry, a minimum of 3 levels of refinement were specified. Additionally, curvature-based refinement was allowed up to 5 levels. This yielded a mesh of approximately 16.9 Mtet. The grid obtained in this way, as well as the corresponding solution are shown in Figures 13(b) and (c). Note that all geometrical details have been properly resolved. The mesh was subsequently refined based on density, up to approximately 28 Mtet. This physics-based mesh refinement is evident in Figures 13(c)–(d).

12.3. Blast interaction with a generic ship hull

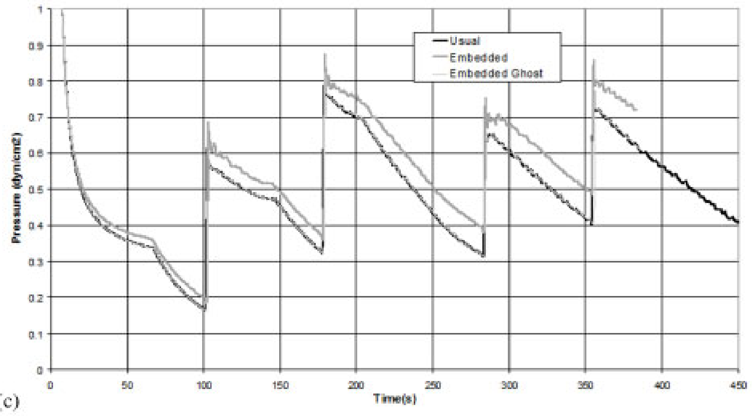
Figure 14 shows the interaction of an explosion with a generic ship hull. For this fully coupled CFD/CSD run, the structure was modeled with quadrilateral shell elements, the (inviscid) fluid as a mixture of high explosive and air, and mesh embedding was employed. The structural elements were assumed to fail once the average strain in an element exceeded 60%. As the shell elements failed, the fluid domain underwent topological changes. Figures 14(a)–(d) show the structure as well as the pressure contours in a cut plane at two times during the run. The



(a)



(b)



(c)

Figure 12. (a) Shock tube problem: embedded surface; (b) shock tube problem: density contours; and (c) pressure time history.

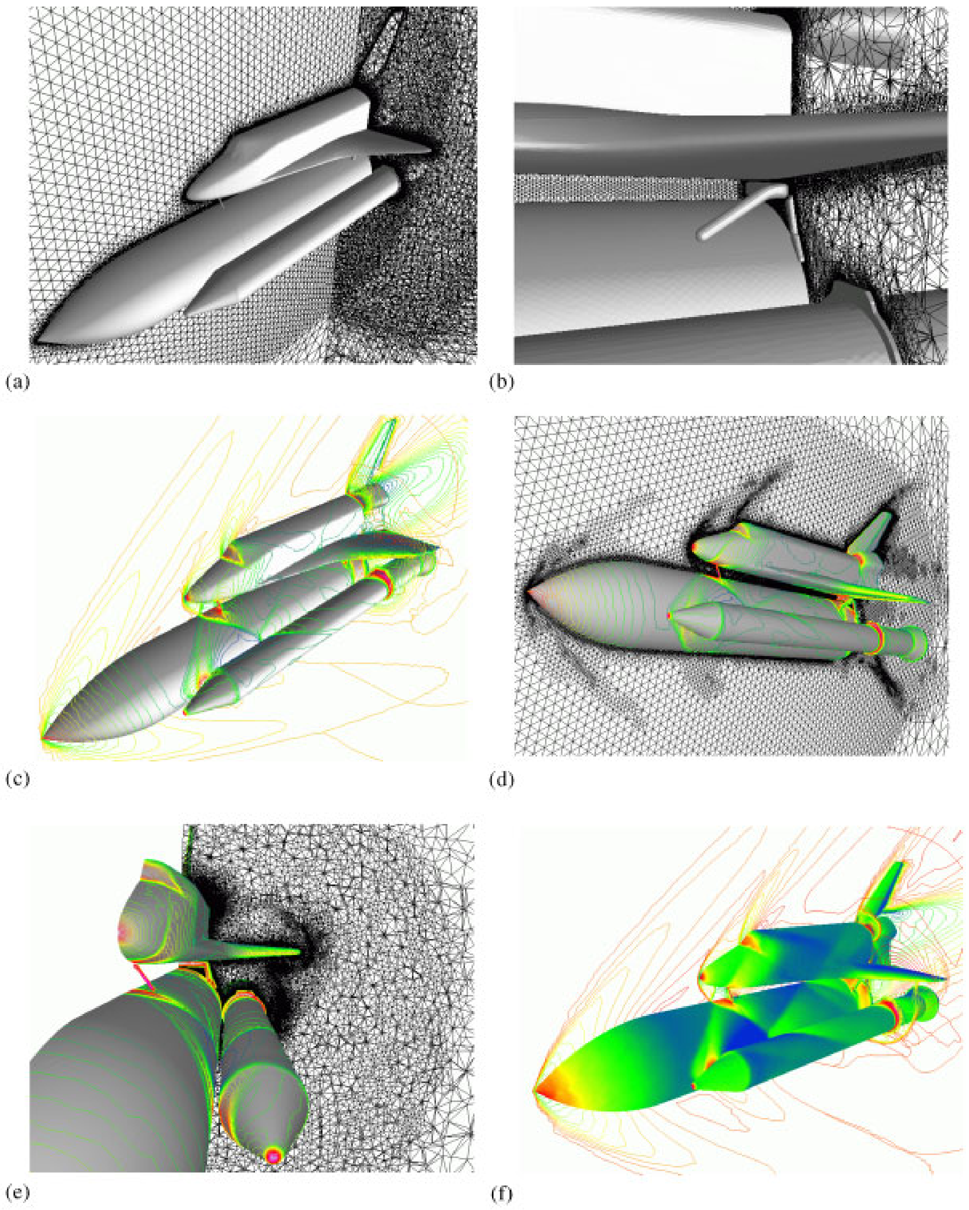


Figure 13. (a) Shuttle: general view; (b) shuttle: detail; (c) surface pressure and field Mach-Nr; (d) surface pressure and mesh (cut plane); (e) surface pressure and mesh (cut plane); (f) and surface pressure and field Mach-Nr.

influence of bulkheads on surface velocity can clearly be discerned. Note also the failure of the structure, and the invasion of high pressure into the chamber. The distortion and interpenetration of the structural elements is such that the traditional moving mesh approach (with

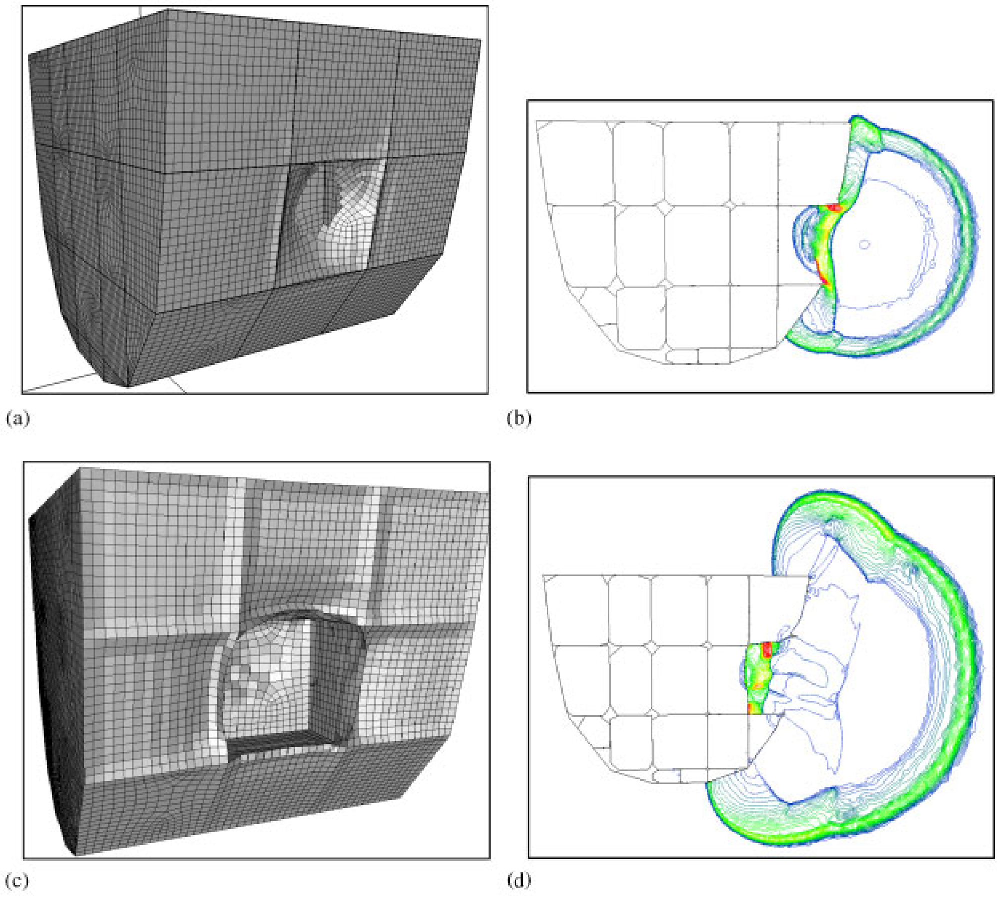


Figure 14. (a) Surface of generic ship hull; (b) pressure in cut plane at 20 ms; (c) surface of generic ship hull; and (d) pressure in cut plane at 50 ms.

topology reconstruction, remeshing, ALE formulation, remeshing, etc.) will invariably for this class of problems. In fact, it was this particular type of application that led to the development of the present embedded CSD capability.

12.4. Generic weapon fragmentation

Figure 15 shows a generic weapon fragmentation study. The CSD domain was modelled with approximately 66 Khex elements corresponding to 1555 fragments whose mass distribution matches statistically the mass distribution encountered in experiments. The structural elements were assumed to fail once the average strain in an element exceeded 60%. The high explosive was modeled with a Jones–Wilkins–Lee equation of state [22]. The CFD mesh was refined to 3 levels in the vicinity of the solid surface. Additionally, the mesh was refined based on the

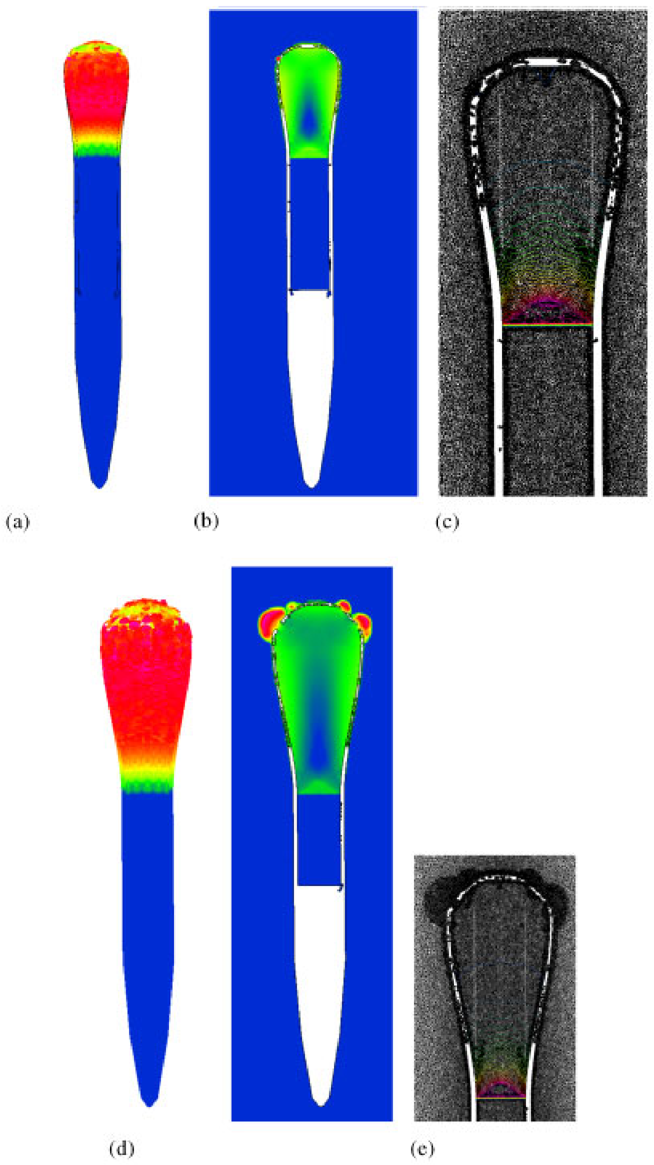


Figure 15. (a–c) CSD/flow velocity and pressure/mesh at 68 ms; and (d,e) CSD/flow velocity and pressure/mesh at 102 ms.

modified interpolation error indicator proposed in Reference [24], using the density as indicator variable.

Adaptive refinement was invoked every 5 time steps during the coupled CFD/CSD run. The CFD mesh started with 39 Mtet, and ended with 72 Mtet. Figures 15(a)–(f) show the structure as well as the pressure contours in a cut plane at two times during the run. The detonation

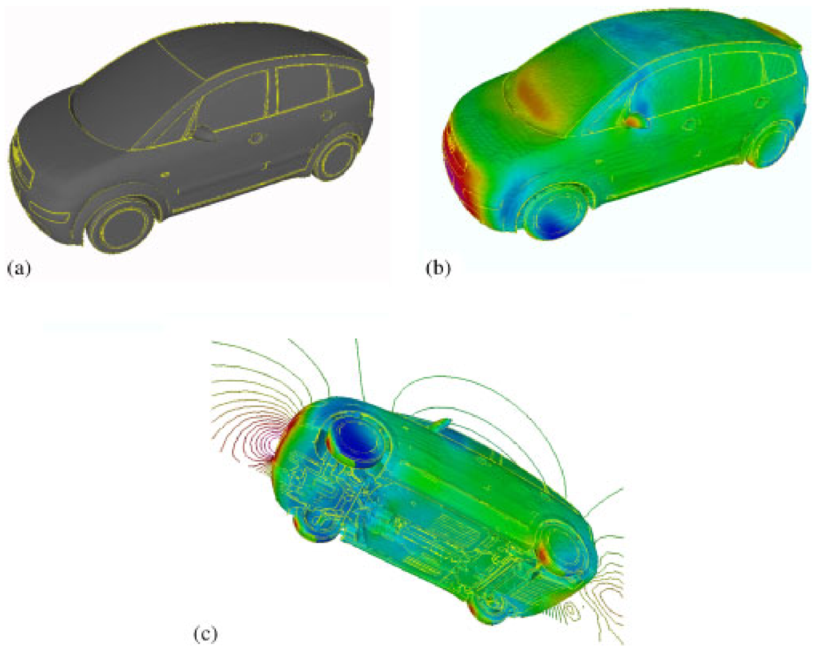


Figure 16. (a) Surface triangulation for generic car; (b) surface pressure (top); and (c) surface pressure (bottom).

wave is clearly visible, as well as the thinning of the structural walls and the subsequent fragmentation.

12.5. Flow past generic car

The next two cases consider incompressible flow. The first one of these is the flow past a generic car. The geometric information consisted of a non-watertight triangulation with approximately 350 Ktria. This triangulation was introduced into a box, and a coarse unstructured grid of approximately 800 Kels was generated. This mesh was subsequently refined based on edges crossed by the triangulation of the car geometry.

The final mesh had approximately 8 Mtet. The results obtained for a time-accurate VLES run with no-slip boundary conditions for the velocities on the car are displayed in Figure 16. The fact that this case was run 'blind', i.e. without user intervention from start to finish, attests to the power of the embedded approach where applicable.

12.6. Plate excitation behind box

This case considers a plate excited by the wake of a box. The case is of particular interest because experimental results have shown that the plate response can enter a chaotic regime under certain initial conditions [25]. The flow is assumed to be incompressible and Newtonian.

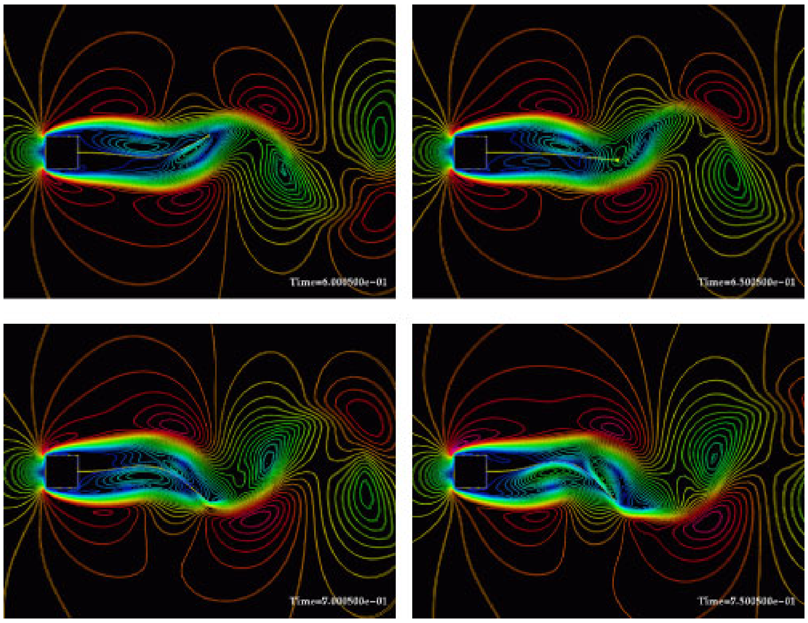


Figure 17. Plate in incompressible flow.

The plate was modeled by its first 4 eigenmodes. Altogether, the following parameters were set:

Box: 1 cm $\times$ 1 cm
Plate length: 4 cm
Plate eigenstiffness: 6.017, 37.71, 105.6, 947.8
Incoming flow conditions:
$\rho = 1.18e - 3\ \mathrm{g/cm^3}$
$v = 31.5\ \mathrm{cm/s}$
$\mu = 0.182e - 3\ \mathrm{g\ cm/s}$

For the fluid–structure coupling, an implicit technique was employed. During each implicit timestep, the surface positions/velocities and surface tensions are iterated out. This takes on average 2-3 iterations, although during startup as many as seven iterations have been observed. Although 2-D, the problem is run as 3-D. Figure 17 shows preliminary results for the velocity. The formation of the vortex street, as well as the deformation of the plate are clearly discernable.

13. CONCLUSIONS AND OUTLOOK

A CFD solver based on unstructured, body conforming grids has been extended to treat embedded or immersed CSD surfaces given by triangulations. The key modification of the original, edge-based solver is to zero out all geometry-parameters (essentially the normals) belonging to edges cut by CFD faces. In order to guarantee a minimum level of accuracy, additional boundary points, belonging to the end-points of cut edges, are introduced. The boundary conditions are enhanced further by extrapolating velocities (for Navier–Stokes) and/or pressure normals. Alternatively, higher-order boundary conditions may be imposed by duplicating the crossed edges and their end-points. Adaptive mesh refinement based on proximity to or the curvature of the embedded CSD surfaces is used to enhance the accuracy of the solution. User-defined or automatic deactivation for the regions inside immersed solid bodies is employed to avoid unnecessary work. Several examples have been included that show the viability of this approach for viscous and inviscid flow problems.

Future work will centre on improved boundary conditions, as well as faster crossing checks for transient problems.

ACKNOWLEDGEMENTS

This work was partially supported by DTRA and AFOSR. The technical monitors were Drs. Michael Giltrud, Young Sohn and Leonidas Sakell.

REFERENCES

1. Löhner R. *Applied CFD Techniques*. Wiley: New York, 2001.
2. George PL, Borouchaki H. *Delaunay Triangulation and Meshing*. Editions Hermes: Paris, 1998.
3. Cebral JR, Löhner R. From medical images to anatomically accurate finite element grids. *International Journal for Numerical Methods in Engineering* 2001; **51**:985–1008.
4. Baum JD, Luo H, Löhner R. Validation of a new ALE, adaptive unstructured moving body methodology for multi-store ejection simulations. *AIAA*-95-1792, 1995.
5. Baum JD, Luo H, Löhner R, Yang C, Pelessone D, Charman C. A coupled fluid/structure modeling of shock interaction with a truck. *AIAA*-96-0795, 1996.
6. Baum JD, Luo H, Mestreau E, Löhner R, Pelessone D, Charman C. A coupled CFD/CSD methodology for modeling weapon detonation and fragmentation. *AIAA*-99-0794, 1999.
7. Sharov D, Luo H, Baum JD, Löhner R. Time-accurate implicit ALE algorithm for shared-memory parallel computers. *First International Conference on Computational Fluid Dynamics*, Kyoto, Japan, July 10–14, 2000.
8. Clarke DK, Hassan HA, Salas MD. Euler calculations for multielement airfoils using cartesian grids. *AIAA*-85-0291, 1985.
9. de Zeeuw D, Powell K. An adaptively-refined cartesian mesh solver for the euler equations. *AIAA*-91-1542, 1991.
10. Melton JE, Berger MJ, Aftosmis MJ. 3-D applications of a cartesian grid euler method. *AIAA*-93-0853-CP, 1993.
11. Quirk JJ. A cartesian grid approach with hierarchical refinement for compressible flows. *NASA CR*-194938, *ICASE Report No.* 94-51, 1994.
12. Karman SL. SPLITFLOW: A 3-d unstructured cartesian/prismatic grid CFD code for complex geometries. *AIAA*-95-0343, 1995.
13. Pember RB, Bell JB, Colella P, Crutchfield WY, Welcome ML. An adaptive cartesian grid method for unsteady compressible flow in irregular regions. *Journal of Computational Physics* 1995; **120**:278.
14. Landsberg AM, Boris JP. The virtual cell embedding method: a simple approach for gridding complex geometries. *AIAA*-97-1982, 1997.
15. LeVeque RJ, Calhoun D. Cartesian grid methods for fluid flow in complex geometries. In *Computational Modeling in Biological Fluid Dynamics*, *IMA in Mathematics and its Applications*, Vol. 124, Fauci LG, Gueron S (eds.). Springer-Verlag: Berlin, 2001, pp. 117–143.
16. Aftosmis MJ, Berger MJ, Adomavicius G. A parallel multilevel method for adaptively refined cartesian grids with embedded boundaries. *AIAA*-00-0808, 2000.

17. Dadone A, Grossman B. An immersed boundary methodology for inviscid flows on cartesian grids. *AIAA*-02-1059, 2002.
18. Peskin CS. The immersed boundary method. *Acta Numerica* 2002; **11**:479–517.
19. Tsuboi K, Miyakoshi K, Kuwahara K. Incompressible flow simulation of complicated boundary problems with rectangular grid system. *Theoretical and Applied Mechanics* 1991; **40**:297–309.
20. Del Pino S, Pironneau O. Fictitious Domain Methods and Freefem3d; *Proceedings of the ECCOMAS CFD Conference*, Swansea, Wales, 2001.
21. Löhner R. Computational aspects of space-marching. *AIAA*-98-0617, 1998.
22. Löhner R, Yang C, Baum JD, Luo H, Pelessone D, Charman C. The numerical simulation of strongly unsteady flows with hundreds of moving bodies. *International Journal for Numerical Methods in Fluids* 1999; **31**:113–120.
23. Löhner R, Yang C, Cebral J, Baum JD, Luo H, Mestreau E, Pelessone D, Charman C. Fluid-structure interaction algorithms for rupture and topology change. *Proceedings of 1999 JSME Computational Mechanics Division Meeting*, Matsuyama, Japan, November, 1999.
24. Löhner R, Baum JD. Adaptive H-refinement on 3-D unstructured grids for transient problems. *International Journal for Numerical Methods in Fluids* 1992; **14**:1407–1419.
25. Walhorn E. Ein Simultanes Berechnungsverfahren für Fluid-Struktur-Wechselwirkungen mit finiten Raum-Zeit-Elementen. *Bericht 2002-95*, Institut für Statik, TU Braunschweig, 2002.
26. Baum JD, Luo H, Löhner R, Goldberg E, Feldhun A. Application of unstructured adaptive moving body methodology to the simulation of fuel tank separation from an F-16 C/D fighter. *AIAA*-97-0166, 1997.