Mecánica Computacional Vol. XXIII, pp. 29-42
G.Buscaglia, E.Dari, O.Zamonsky (Eds.)
Bariloche, Argentina, November 2004

ADAPTIVE EMBEDDED UNSTRUCTURED GRID METHODS

Rainald Löhner[1], Juan Cebral[1], Marcelo Castro[1],
Joseph D. Baum[2], Hong Luo[2], Eric Mestreau[2], and Orlando Soto[2]

[1]School of Computational Sciences, MS 4C7
George Mason University, Fairfax, VA 22030-4444, USA
e-mail: rlohner@gmu.edu, web page: http://www.scs.gmu.edu/∼rlohner
[2]Advanced Technology Group
Science Applications Int. Corp., McLean, VA, USA

Key Words: Embedded Grids, Adaptive Unstructured Grids, FEM, CFD

Abstract. *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. Recent work has led to a notable improvement in speed via suitable data structures, the option to treat dispersed particles in the context of embedded surfaces, a direct link to Discrete Particle Methods (DPM), a volume to surface meshing technique that obtains body-fitted grids by post-processing adaptive embedded grids; and links to simplified CSD models. 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.*

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.[18] The spatial discretization is commonly performed via polyhedra, also called (finite) volumes or elements. The final assembly of these polyhedra yields the so-called mesh. The transition from an arbitrary surface description to a proper mesh still represents a difficult task.[7,10] 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.[2,3,5,27] 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 velocites 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[1,8,9,11–13,21–24,26,29] 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.
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-based 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.[24,25] 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.

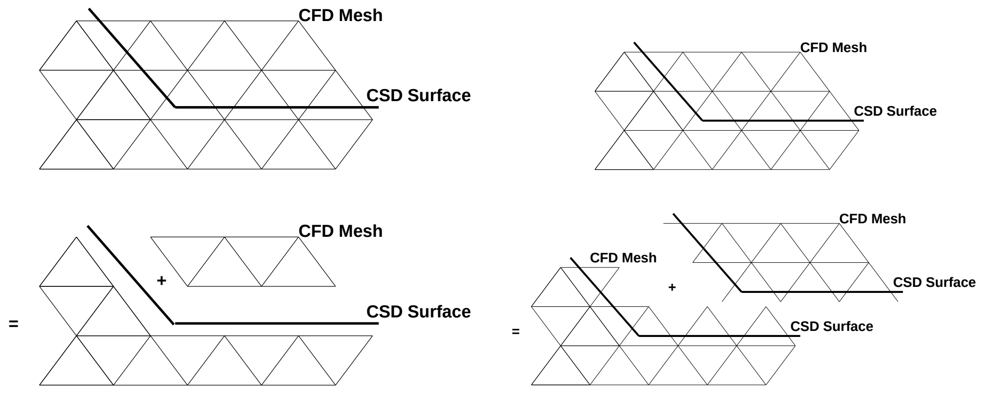


Figure 1: Treatment of Embedded Surfaces

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;

- 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 a bin or 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 bin or 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. A comparison of CPU requirements for both data structures (octree, bin) revealed that the bin is approximately an order of magnitude faster. This comparison extended to many cases, and seems to be very consistent. The main reason seems to be that both during storage and retrieval of information the number of jumps in memory (and hence cache-misses) is much lower for the bin than for the octree. Even if the bin yields more repeated faces from a query, these can be removed using local storage techniques (e.g. hash tables) in a very fast way, more than compensating for additional jumps in memory.

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 timestep. If the topology of the CSD triangulation has not changed, the crossed-edge information from the previous timestep can be re-checked. The points of edges no longer crossed by a face crossing them in the previous timestep 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 timestep is no longer valid. However, the points close to cut edges in the previous timestep 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[18] is given by:

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

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:

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

The point-coefficients C_Γ^i 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 Eqn.(1) as:

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

it would be desirable to build the C_Γ^i 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:

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

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.
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 and pressure gradients from the surface with field information.[19]

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. 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.[19] Proper handling of the interpolation is also required as the element used for the interpolation might either be crossed (Figure 2a) or not exist (Figure 2b).

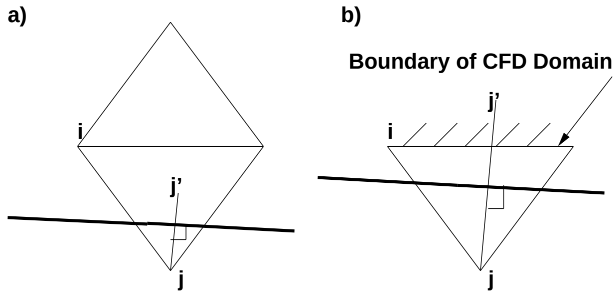


Figure 2: Problem Cases

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 3). The best way to treat such points is to simply deactivate them,[18] chapter 16.

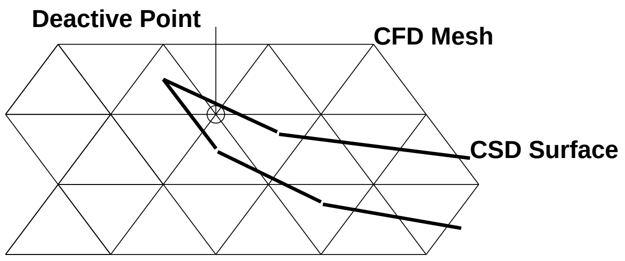


Figure 3: Deactive Point

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. We denote by seed points points that the user specifies as being in/outside an object. The closest CFD field point to a seed point is then obtained. Starting from this point, additional points are added using an advancing front (nearest neighbour layer) algorithm, and flagged as in/active. The procedure stops once points that are attached to crossed edges have been reached. 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.[19]
Once the points have been marked as active/inactive, the element and edge-groups required for vectorization are inspected in turn. As with spacemarching,[15] 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 4). 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.

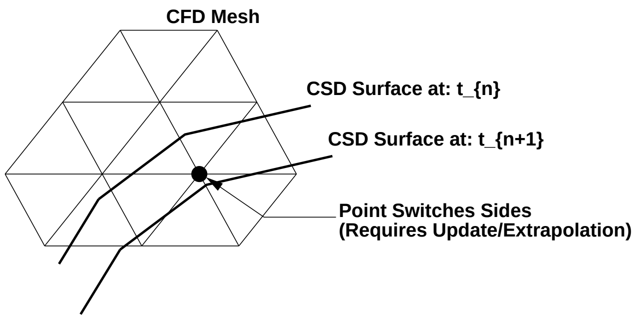


Figure 4: Extrapolation of Solution

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.[5,16,17] 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 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. 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 5);
- 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 bin or modified octree for the faces.

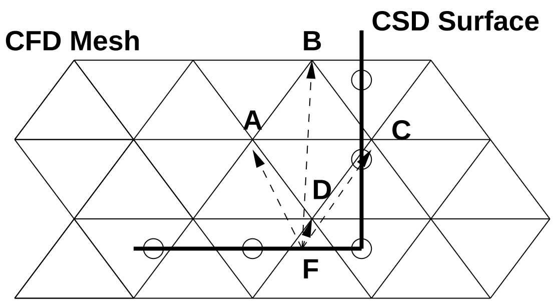
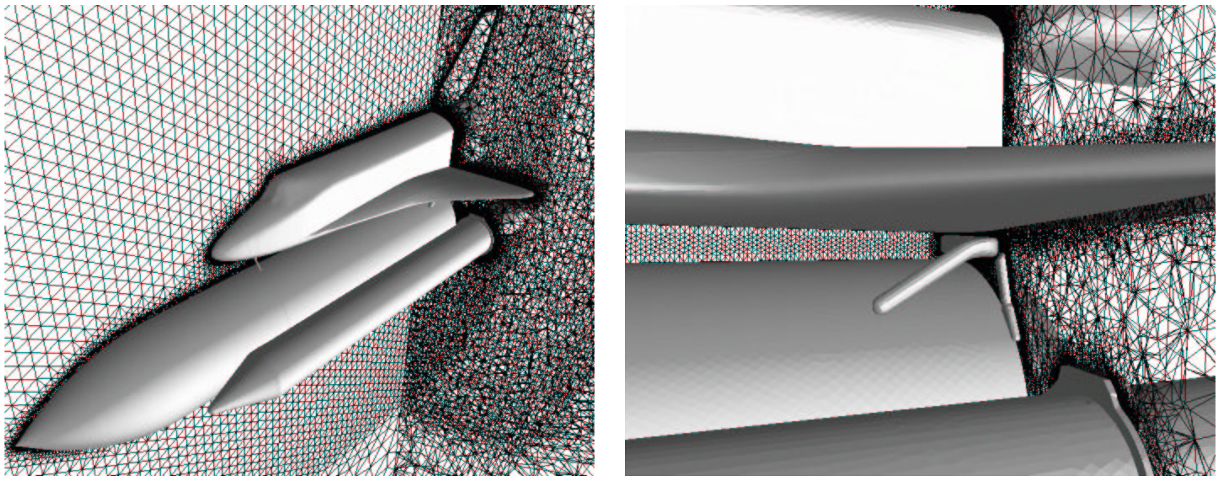


Figure 5: Transfer of Stresses/Fluxes

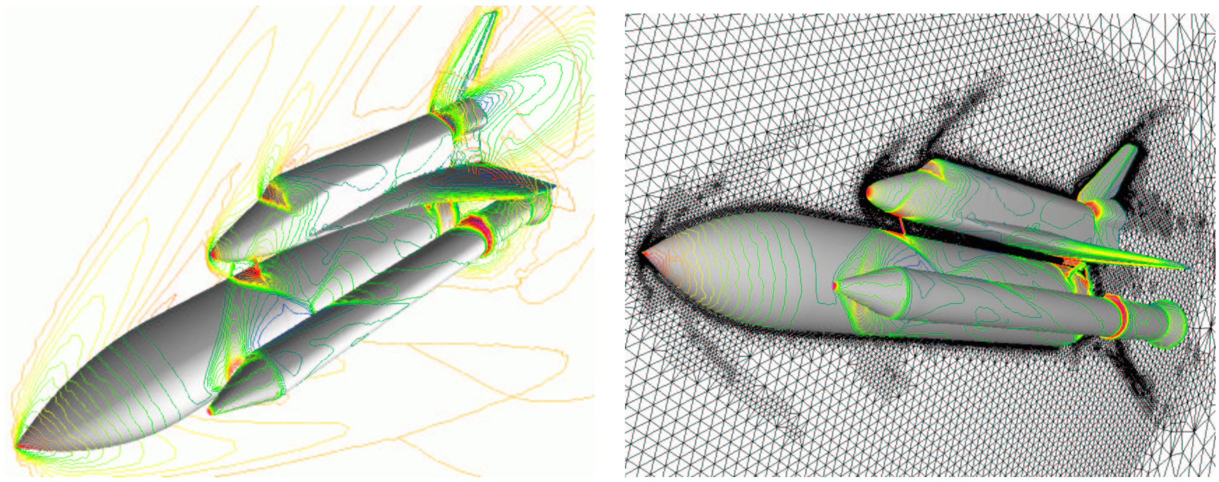
10 EXAMPLES

The embedded CSD technique has been in use for more than two years, and has been benchmarked, as well as applied to numerous examples.[6,19,20] For lack of space, we only include a few representative cases here.

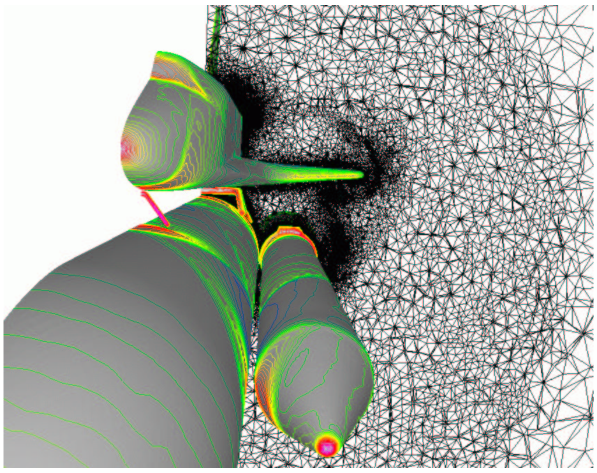
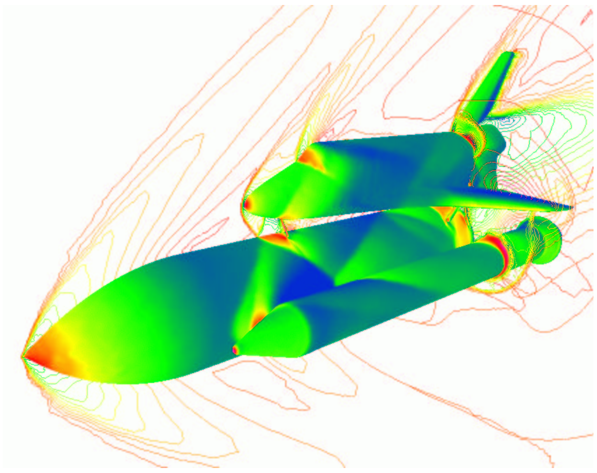
a) Shuttle Ascend Configuration: This example considers the Space Shuttle Ascend configuration shown in Figure 6a. The external flow is at $Ma = 2$ and angle of attack $\alpha = 5^o$. As before, the flow is treated as compressible and inviscid, with no-penetration (slip) boundary conditions for the velocities 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 6b,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 6c-d.



Figures 6a,b: Shuttle: General View and Detail

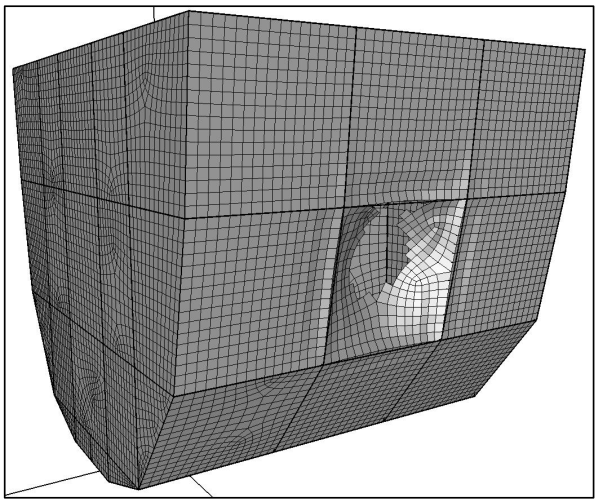
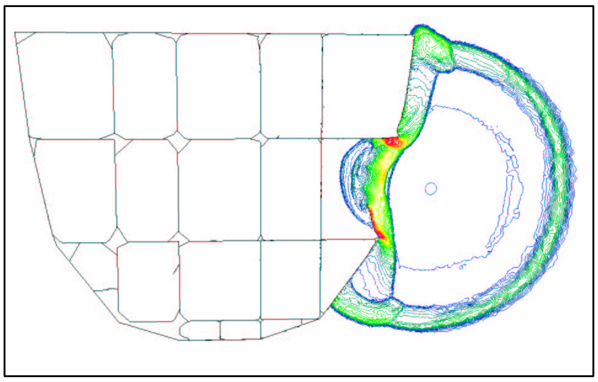


Figures 6c,d: Surface Pressure, Field Mach-Nr. and Mesh (Cut Plane)

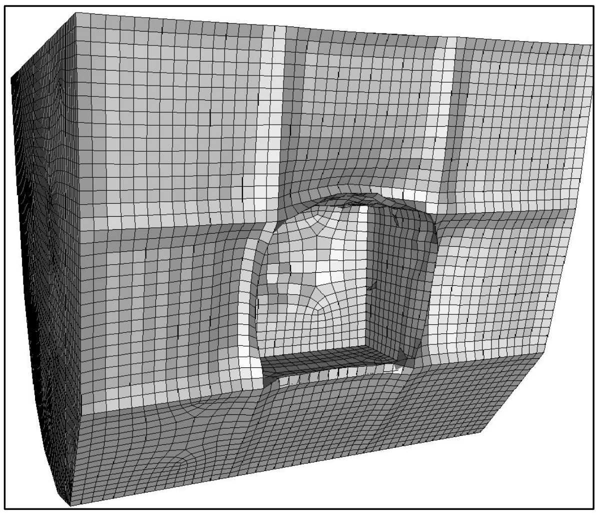
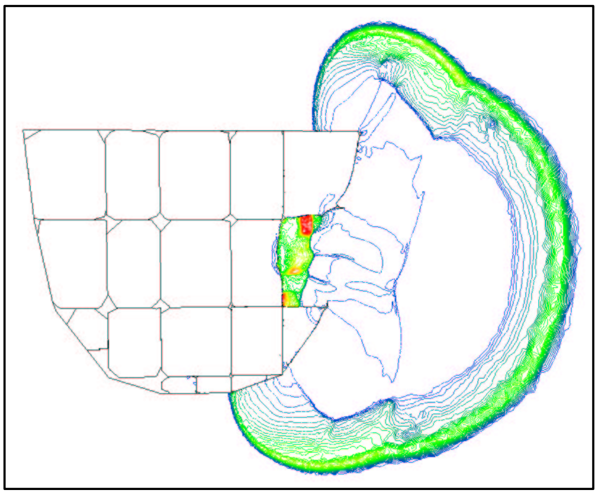



Figures 6e,f: Surface Pressure and Mesh (Cut Plane) and Adapted

12.3 Blast Interaction With a Generic Ship Hull: Figure 7 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 7a-d show the structure as well as the pressure contours in a cut plane at two times during the run. The 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 inter-penetration of the structural elements is such that the traditional moving mesh approach (with 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.

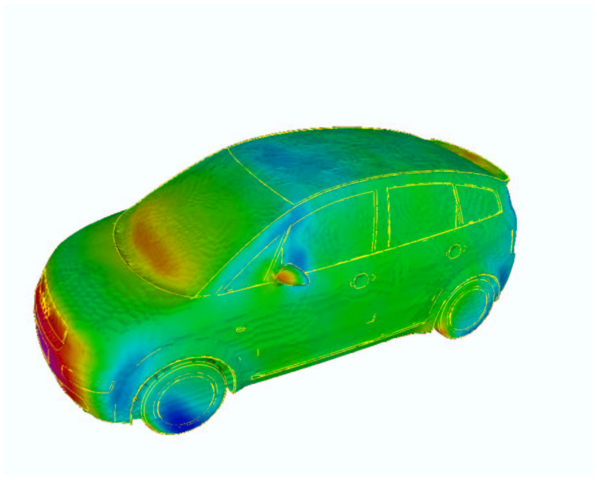
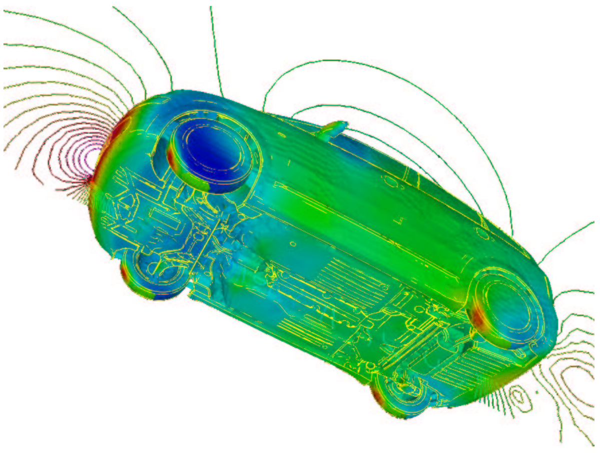



Figures 6a,b: Surface of Generic Ship Hull and Pressure in Cut Plane at 20msec

Figures 6c,d: Surface of Generic Ship Hull and Pressure in Cut Plane at 50msec

12.5 Flow Past Generic Car: The next case considers the incompressible 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.

Figures 8a,b: Generic Car: Surface Pressures

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 8. 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.

11 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 center on improved boundary conditions, faster crossing checks for transient problems, volume to mesh gridding[20] and link to DPM.

12 ACKNOWLEDGEMENTS

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

REFERENCES

[1] M.J. Aftosmis, M.J. Berger and G. Adomavicius - A Parallel Multilevel Method for Adaptively Refined Cartesian Grids with Embedded Boundaries; *AIAA*-00-0808 (2000).

[2] J.D. Baum, H. Luo and R. Löhner - Validation of a New ALE, Adaptive Unstructured Moving Body Methodology for Multi-Store Ejection Simulations; *AIAA*-95-1792 (1995).

[3] J.D. Baum, H. Luo, R. Löhner, C. Yang, D. Pelessone and C. Charman - A Coupled Fluid/Structure Modeling of Shock Interaction with a Truck; *AIAA*-96-0795 (1996).

[4] J.D. Baum, H. Luo, R. Löhner, E. Goldberg and A. Feldhun - 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).

[5] J.D. Baum, H. Luo, E. Mestreau, R. Löhner, D. Pelessone and C. Charman - A Coupled CFD/CSD Methodology for Modeling Weapon Detonation and Fragmentation; *AIAA*-99-0794 (1999).

[6] J.D. Baum, E. Mestreau, H. Luo, R. Löhner, D. Pelessone and Ch. Charman - Modeling Structural Response to Blast Loading Using a Coupled CFD/CSD Methodology; *Proc. Des. An. Prot. Struct. Impact/ Impulsive/ Shock Loads (DAPSIL)*, Tokyo, Japan, December (2003).

[7] J.R. Cebral and R. Löhner - From Medical Images to Anatomically Accurate Finite Element Grids; *Int. J. Num. Meth. Eng.* 51, 985-1008 (2001).

[8] D.K. Clarke, H.A. Hassan and M.D. Salas - Euler Calculations for Multielement Airfoils

Using Cartesian Grids; *AIAA*-85-0291 (1985).

[9] A. Dadone and B. Grossman - An Immersed Boundary Methodology for Inviscid Flows on Cartesian Grids; *AIAA*-02-1059 (2002).

[10] P.L. George and H. Borouchaki - *Delaunay Triangulation and Meshing*; Editions Hermes, Paris (1998).

[11] S.L. Karman - SPLITFLOW: A 3-D Unstructured Cartesian/ Prismatic Grid CFD Code for Complex Geometries; *AIAA*-95-0343 (1995).

[12] A.M. Landsberg and J.P. Boris - The Virtual Cell Embedding Method: A Simple Approach for Gridding Complex Geometries; *AIAA*-97-1982 (1997).

[13] R.J. LeVeque and D. Calhoun - Cartesian Grid Methods for Fluid Flow in Complex Geometries; pp. 117-143 in *Computational Modeling in Biological Fluid Dynamics* (L. J. Fauci and S. Gueron, eds.), IMA Volumes in Mathematics and its Applications 124, Springer-Verlag (2001).

[14] R. Löhner and J.D. Baum - Adaptive H-Refinement on 3-D Unstructured Grids for Transient Problems; *Int. J. Num. Meth. Fluids* 14, 1407-1419 (1992).

[15] R. Löhner - Computational Aspects of Space-Marching; *AIAA*-98-0617 (1998).

[16] R. Löhner, C. Yang, J.D. Baum, H. Luo, D. Pelessone and C. Charman - The Numerical Simulation of Strongly Unsteady Flows With Hundreds of Moving Bodies; *Int. J. Num. Meth. Fluids* 31, 113-120 (1999).

[17] R. Löhner, C. Yang, J. Cebral, J.D. Baum, H. Luo, E. Mestreau, D. Pelessone and C. Charman - Fluid-Structure Interaction Algorithms for Rupture and Topology Change; *Proc. 1999 JSME Computational Mechanics Division Meeting*, Matsuyama, Japan, November (1999).

[18] R. Löhner - *Applied CFD Techniques*; J. Wiley & Sons (2001).

[19] R. Löhner, J.D. Baum, E. Mestreau, D. Sharov, C. Charman and D. Pelessone - Adaptive Embedded Unstructured Grid Methods; *Int. J. Num. Meth. Eng.* 60, 641-660 (2004).

[20] R. Löhner, J.D. Baum and E.L. Mestreau - Advances in Adaptive Embedded Unstructured Grid Methods; *AIAA*-04-0083 (2004).

[21] J.E. Melton, M.J. Berger and M.J. Aftosmis - 3-D Applications of a Cartesian Grid Euler Method; *AIAA*-93-0853-CP (1993).

[22] S. M. Murman, M. J. Aftosmis and M. J. Berger, Simulations of 6-DOF Motion with a Cartesian Method; *AIAA*-03-1246 (2003).

[23] R.B. Pember, J.B. Bell, P. Colella, W.Y. Crutchfield and M.L. Welcome - An Adaptive Cartesian Grid Method for Unsteady Compressible Flow in Irregular Regions; *J. Comp. Phys.* 120, 278 (1995).

[24] C.S. Peskin - The Immersed Boundary Method; *Acta Numerica* 11, 479-517 (2002).

[25] S. Del Pino and O. Pironneau - Fictitious Domain Methods and Freefem3d; *Proc. ECCOMAS CFD Conf.* , Swansea, Wales (2001).

[26] J.J. Quirk - A Cartesian Grid Approach with Hierarchical Refinement for Compressible Flows; *NASA CR-194938, ICASE Report No.* 94-51, (1994).

[27] D. Sharov, H. Luo, J.D. Baum and R. Löhner - Time-Accurate Implicit ALE Algorithm

for Shared-Memory Parallel Computers; *First International Conference on Computational Fluid Dynamics*, Kyoto, Japan, July 10-14 (2000).

[28] Tsuboi, K., K. Miyakoshi and K. Kuwahara - Incompressible Flow Simulation of Complicated Boundary Problems with Rectangular Grid System; *Theoretical and Applied Mech.* 40, 297-309 (1991).

[29] D. de Zeeuw and K. Powell - An Adaptively-Refined Cartesian Mesh Solver for the Euler Equations; *AIAA*-91-1542 (1991).