Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 197 (2008) 2173–2197
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Adaptive embedded and immersed unstructured grid techniques

Rainald Löhner [a,*], Juan R. Cebral [a], Fernando E. Camelli [a], S. Appanaboyina [a], Joseph D. Baum [b], Eric L. Mestreau [b], Orlando A. Soto [b]

[a] Center for Computational Fluid Dynamics, M.S. 6A2, Department of Computational and Data Sciences, College of Sciences, George Mason University, Fairfax, VA 22030-4444, USA
[b] Advanced Technology Group, SAIC, SAIC Drive, McLean, VA 22020, USA

Received 20 February 2007; received in revised form 1 September 2007; accepted 4 September 2007
Available online 25 September 2007

Abstract

Embedded mesh, immersed body or fictitious domain techniques have been used for many years as a way to discretize geometrically complex domains with structured grids. The use of such techniques within adaptive, unstructured grid solvers is relatively recent. The combination of body-fitted functionality for some portion of the domain, together with embedded mesh or immersed body functionality for another portion of the domain offers great advantages, which are increasingly being exploited. The present paper reviews the methodologies pursued so far, addresses implementational issues and shows the possibilities such techniques offer.

Keywords: Embedded surface method; Immersed body technique; Finite elements; CFD

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. The transition from an arbitrary surface description to a proper mesh still represents a difficult task. This is particularly so when the surface description is based on data that does not originate from CAD-systems, such as data from remote sensing, medical imaging or fluid–structure interaction problems. 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.

To date, most of the field solvers based on unstructured grids have only considered body-conforming grids, i.e. grids where the external mesh faces match up with the surface (body surfaces, external surfaces, etc.) of the domain. The assumption was that one the perceived strengths of field solvers based on unstructured grids is precisely the ability to mesh arbitrary domains. The present paper will consider the case when elements and points do not match up perfectly with the body. Solvers or methods that employ these nonbody-conforming grids are known by a variety of names: embedded mesh, fictitious domain, immersed boundary, immersed body, Cartesian method etc. The key idea is to place the bodies inside a flow region inside a large mesh (typically a regular parallelepiped), with special treatment of the elements and points close to the surfaces and/or inside the bodies. If we consider the general case of moving or deforming surfaces with topology change, as the mesh is not body-conforming, it does not have to move. Hence, the PDEs describing the flow can be left in the simpler Eulerian frame of reference even for moving surfaces. At every timestep, the elements/edges/points close to and/or inside the embedded/immersed surface are identified and proper boundary conditions are applied in their vicinity. While used extensively [1,15,17–19,21,22,27,29–32,40,41,45,46,51–53,55,56,62,65,66] this solution strategy also exhibits some shortcomings:

* Corresponding author.
E-mail address: rlohner@gmu.edu (R. Löhner).

0045-7825/$ - see front matter © 2007 Published by Elsevier B.V.
doi:10.1016/j.cma.2007.09.010

- the boundary, which, in the absence of field sources has the most profound influence on the ensuing physics, is also the place where the worst elements/approximations are found;
- near the boundary, the embedding boundary conditions need to be applied, in many cases 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;
- for problems with moving boundaries the information required to build the proper boundary conditions for elements close to the surface or inside the bodies can take a considerable amount of time; and
- for fluid–structure interaction problems, obtaining the information required to transfer forces back to the structural surface can also take a considerable amount of time.

In nearly all cases reported to date, embedded or immersed boundary techniques were developed as a response to the treatment of problems with:

- 'dirty geometries' [1,7,9,30];
- moving/sliding bodies with thin/vanishing gaps [7,63]; and
- physics that can be handled with isotropic grids:
  - potential flow or Euler: Quirk [55], Karman [27], Pember et al. [52], Landsberg and Boris [30], Aftosmis et al. [1], Le Veque and Calhoun [32], Dadone [18], Baum et al. [7],
  - Euler with boundary layer corrections: Aftosmis et al. [2],
  - low Reynolds-number laminar flow and/or LES: Angot et al. [3], Turek [62], Fadlun et al. [20], Kim et al. [28], Gilmanov et al. [21], Balaras [5], Gilmanov and Sotiropoulos [22], Mittal and Iaccarino [46], Cho et al. [14], Yang and Balaras [64].

The human cost of repairing bad input data sets can be overwhelming in many cases. For some cases, it may even constitute an impossible task. Consider, as an example, the class of fluid–structure interaction problems where surfaces may rupture and self-contact due to blast damage [6,7,37]. The contact algorithms of most computational structural dynamics (CSD) codes are based on some form of spring analogy, and hence can not ensure strict no-penetration during contact. As the surface is self-intersecting, it becomes impossible to generate a topologically consistent, body fitted volume mesh. In such cases, embedded or immersed boundary techniques represent the only viable solution.

Two basic approaches have been proposed to modify field solvers in order to accommodate embedded surfaces or immersed bodies. They are based on either *kinetic* or *kinematic* boundary conditions near the surface or inside the bodies in the fluid. The first type applies an *equivalent balancing force* to the flowfield in order to achieve the kinematic boundary conditions required at the embedded surface or within the embedded domain [3,5,14,20,25,28,46,47,50,53,54,60,63,64]. The second approach is to apply *kinematic boundary conditions* at the nodes close to the embedded surface [1,18,30,40,41].

At first sight, it may appear somewhat contradictory to even consider embedded surface or immersed body techniques in the context of a general unstructured grid solver. Indeed, most of the work carried out to date was in conjunction with Cartesian solvers [1,14,15,18,21,22,27,30,32,45,46,52,61,66], the argument being that flux evaluations could be optimized due to coordinate alignment and that so-called fast Poisson solvers (multigrid, fast Fourier transform) could easily be employed. However, the achievable gains of such coordinate alignment may be limited due to the following mitigating factors:

- 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;
- the fact that any of these schemes (Cartesian, unstructured) requires mesh adaptation in order to be successful immediately implies the use of indirect addressing (and also removes the possibility of using solvers based on fast Fourier transforms); given current trends in microchip design, indirect addressing, present in both types of solvers, may outweigh all other factors;
- three specialized (x,y,z) edge-loops versus one general edge-loop, and the associated data reorganization implies an increase in software maintenance costs.

Indeed, empirical evidence from explicit compressible flow solvers indicates that the gains achievable when comparing general, edge-based, unstructured grid solvers versus optimized cartesian solvers amount to no more than a factor of 1:4, and in most cases are in the range of 1:2 [42].

For a general unstructured grid solver, surface embedding represents just another addition to a toolbox of mesh handling techniques (mesh movement, overlapping grids, remeshing, h-refinement, deactivation, etc.), and one that allows to treat 'dirty geometry' problems with surprising ease, provided the problem is well represented with isotropic grids. It also allows for a combination of different surface treatment options. A good example where this has been used very effectively is the modeling of endovascular devices such as coils and stents [13]. The arterial vessels are gridded using a body-fitted unstructured grid while the endovascular devices are treated via an embedded technique.

This paper is organized as follows: Section 2 describes in general terms the treatment of embedded surfaces or immersed bodies, and details the techniques used to mask edges crossed by surface faces, as well as the points close

to it. Sections 3 and 4 deal with deactivation of interior regions and the extrapolation of the solution for moving embedded surfaces. Adaptive mesh refinement is considered in Section 5, the transfer of loads and fluxes in Section 6, the treatment of gaps or cracks in Section 7, and the direct link to particles in Section 8. Enhancements for visualization of results are mentioned in Section 9, and numerical examples are presented in Section 10. Finally, some conclusions and an outlook for future developments are given in Section 11.

2. Treatment of embedded surfaces or immersed bodies

In what follows, we denote by CSD faces the surface of the computational domain (or surface) 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, medical images or from a CSD code (hence the name) in coupled fluid–structure applications. For immersed body methods we assume that the embedded object is given by a tetrahedral mesh. Furthermore, we assume that in both cases besides the connectivity and coordinate information, also the velocity of the points is given.

2.1. Kinetic treatment of embedded or immersed objects

As stated before, one way of treating embedded objects is via the addition of suitable force-functions that let the fluid 'feel' the presence of the surface, and push away any fluid trying to penetrate the same. If we consider a rigid, closed body, as sketched in Fig. 1, an obvious aim is to enforce, within the body, the condition $\mathbf{v} = \mathbf{w}_b$ (recall that rigid body motion is a solution of the Navier–Stokes equations). This may be accomplished by applying a force term of the form:

$$\mathbf{f} = -c_0(\mathbf{w}_b - \mathbf{v}) \tag{1}$$

for points that are inside (and perhaps just outside) of the body. This particular type of force function is known as the penalty force technique [3,20,25,28,47].

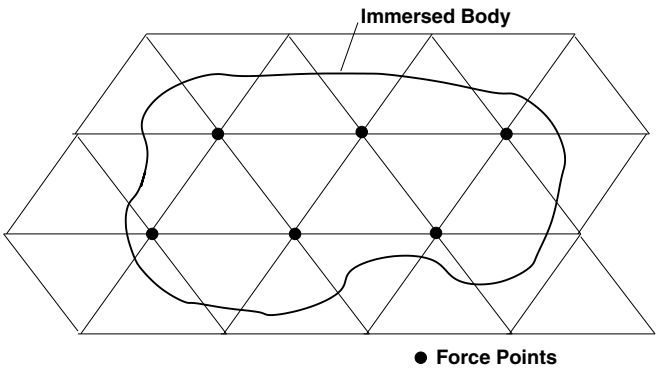

Fig. 1. Kinetic treatment of embedded surfaces.

Of course, other functional forms of $\mathbf{w}_b - \mathbf{v}$ are possible, e.g. the quadratic form:

$$\mathbf{f} = -c_0|\mathbf{w}_b - \mathbf{v}|(\mathbf{w}_b - \mathbf{v}), \tag{2}$$

exponential forms, etc. The damping characteristics in time for the relaxation of a current velocity to the final state will vary, but the basic idea is the same. The advantage of the simple linear form given by Eq. (1) is that a point-implicit integration of the velocities is possible, i.e. the stiffness of the large coefficient c_0 can be removed with no discernable increase in operations [63]. The main problem with force fields given by Eqs. (1) and (2) is the choice of the constants c_0. Values that are too low do not allow the flow to adjust rapidly enough to the motion of the body, while values that are too high may produce artificial stiffness. Moreover, for body motions that are not completely divergence-free a large pressure buildup is observed (see [63] for a case of lobe-pumps). A major improvement was put forward by Mohd-Yusof [47], who proposed to evaluate first the usual right-hand side for the flow equations at immersed points (or cells), and then add a force such that the velocity at the next timestep would satisfy the kinematic boundary condition $\mathbf{v} = \mathbf{v}_b$. Writing the spatially discretized form of the momentum equations at each point (or cell) i as

$$\mathbf{M}\frac{\Delta \mathbf{v}_i}{\Delta t} = \mathbf{r}_i + \mathbf{f}_i, \tag{3}$$

where $\mathbf{M}$, $\mathbf{v}$, $\mathbf{r}_i$ and $\mathbf{f}_i$ denote the mass-matrix, nodal values of the velocity, right-hand side vector due to the momentum equations (advection, viscous terms, Boussinesque and gravity forces) and body force, respectively, $\mathbf{f}^i$ is obtained as

$$\mathbf{f}^i = \mathbf{M}\frac{\mathbf{w}_i^{n+1} - \mathbf{v}_i^n}{\Delta t} - \mathbf{r}^i. \tag{4}$$

Here $\mathbf{w}_i$ denotes the velocity of the immersed body at the location of point (or cell) i, and n the timestep. For explicit timestepping schemes, this force function in effect imposes the (required) velocity of the immersed body at the new timestep, implying that it can also be interpreted as a *kinematic* boundary condition treatment. Schemes of this kind have been used repeatedly (and very successfully) in conjuction with fractional step/ projection methods for incompressible flow [3,5,20,28,40,41,46,49,60,64]. In this case, while the kinematic boundary condition $\mathbf{v}^{n+1} = \mathbf{w}^{n+1}$ is enforced strictly by Eq. (3) in the advective–diffusive prediction step, during the pressure correction step the condition is relaxed, offering the possibility of imposing the kinematic boundary conditions in a 'soft' way.

The imposition of a force given by Eq. (4) for all interior points will yield a first-order scheme for velocities (uncertainty of $O(h)$ in boundary location). This low-order boundary condition may be improved by extrapolating the velocity from the surface with field information to the layer of points surrounding the immersed body. The location where the flow velocity is equal to the surface velocity

is the surface itself, and not the closest boundary point. As shown in Fig. 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 iteration. This lagging of the velocities by one iteration can easily be implemented in any iterative solver.

For cases where the bodies are not rigid, and all that is given is the embedded surface triangulation and movement, the force-terms added take the general form:

$$\mathbf{f} = \int_{\Gamma} \mathbf{F}\delta(\mathbf{x} - \mathbf{X}_{\Gamma})\mathrm{d}\Gamma, \tag{5}$$

where Γ denotes the location of the embedded surface, $\mathbf{X}_{\Gamma}$ the nearest embedded surface point to point $\mathbf{x}$ and $\mathbf{F}$ is the force. In theory, $\mathbf{F}$ should be applied to the fluid using a Dirac delta function δ in order to obtain a sharp interface. In most cases the influence of this delta-function is smeared over several grid points, giving rise to different methods [4,8,23,24,50]. If instead of a surface we are given the volume of the immersed body, then the penalization force may be applied at each point of the flow mesh that falls into the body.

While simple to program and employ, the force-based enforcement is particularly useful if the 'body thickness' covers several CFD mesh elements. This is because the pressures obtained are continuous (and computed !) across the embedded surface/immersed body. This implies that for thin embedded surfaces such as shells, where the pressure is different on both sides, this method will not yield satisfactory results [43].

2.1.1. Implementation details

The search operations required for the imposition of kinetic boundary conditions are as follows:

- given a set of CSD faces (triangulation): find the edges of the CFD mesh cut by CSD faces (and possibly 1–2 layers of neighbours);
- given a set of CSD volumes (tetrahedrization): find the points of the CFD mesh that fall into a CSD tetrahedron (and possibly 1–2 layers of neighbours).

The first task is dealt with extensively below (see 2.2.1). Let us consider the second task in more detail. The problem can be solved in a number of ways:

(a) loop over the *immersed body elements*
  - initialization:
    - store all CFD mesh points in a bin, octree, or any other similar data structure [38].
  - loop over the immersed body elements:
    - determine the bounding box of the element;
    - find all points in the bounding box;
    - detailed analysis to determine the shape function values.

(b) loop over the *CFD mesh points*
  - initialization:
    - store all immersed body elements in a bin, modified octree, or any other similar data structure.
  - loop over the CFD mesh points:
    - obtain the elements in the vicinity of the point;
    - detailed analysis to determine the shape function values.

In both cases, if the immersed body only covers a small portion of the CFD domain, one can reduce the list of points stored or points checked via the bounding box of all immersed body points. Both approaches are easily parallelized on shared memory machines.

2.2. Kinematic treatment of embedded surfaces

Embedded surfaces may be alternatively be treated by applying *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. Figs. 3a and 3b illustrates the basic difference between these approaches for edge-based solvers [38]. Note that in both cases the treatment of infinitely thin surfaces with fluid on both sides (e.g. fluid–structure interaction simulations with shells) is straightforward.

A first-order scheme can be achieved by

- eliminating the edges crossing the embedded surface;
- forming boundary coefficients to achieve flux balance;

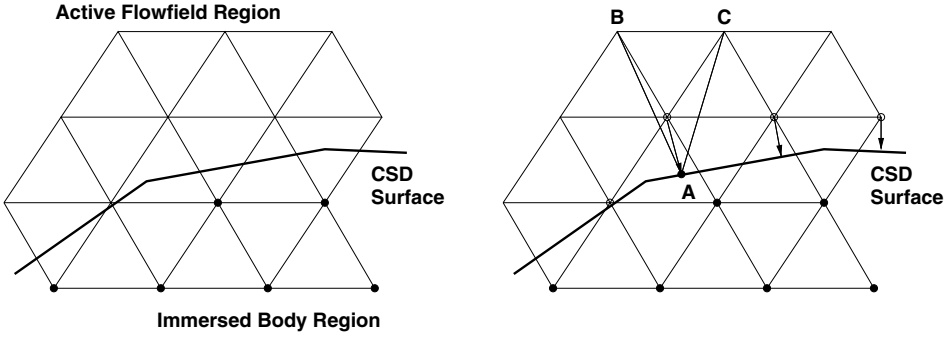

Fig. 2. Extrapolation of velocity for immersed bodies.

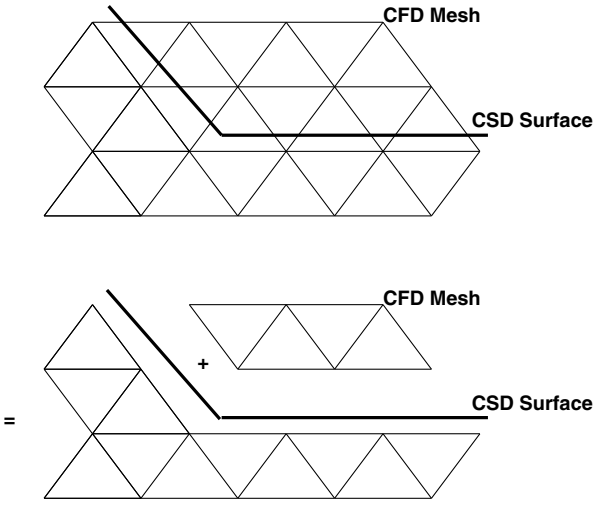

Fig. 3a. First-order treatment of embedded surfaces.

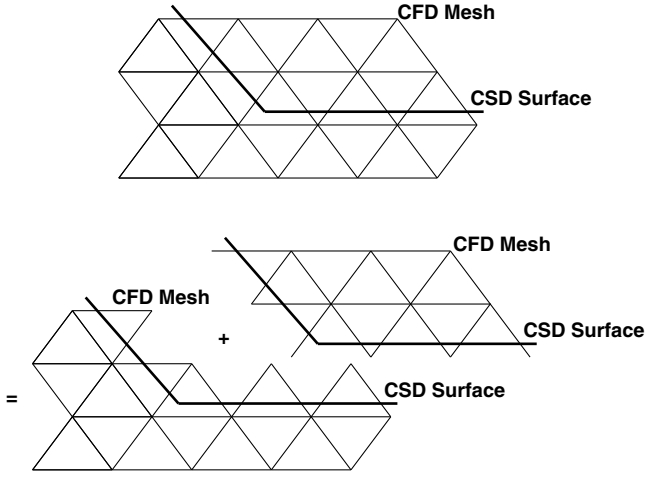

Fig. 3b. Second-order treatment of embedded surfaces.

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

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.

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.

The following sections describe in more detail each one of the steps required, as well as near-optimal techniques to realize them.

2.2.1. 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 is given by Löhner [38]

$$M^i \Delta \mathbf{u}^i = \Delta t \sum_{ij_\Omega} C^{ij}(F_i + F_j). \tag{6}$$

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 $\mathbf{u} = \text{const.}$ 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

$$M^i \Delta \mathbf{u}^i = \Delta t \left[\sum_{ij_\Omega} C^{ij}(F_i + F_j) + C^i_\Gamma F_i \right]. \tag{7}$$

The 'boundary contribution' is the numerical equivalent of the boundary integral that would appear if a boundary fitted mesh had been placed instead. The point-coefficients C^i_Γ are obtained from the condition that $\Delta \mathbf{u} = 0$ for $\mathbf{u} = \text{const.}$ Given that gradients (e.g. for limiting) are also constructed using a loop of the form given by Eq. (6) as

$$M^i \mathbf{g}^i = \sum_{ij_\Omega} C^{ij}(u_i + u_j), \tag{8}$$

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. The simplest possible modification of M^i is given by the so-called 'cut edge fraction' method.

In a pass over the edges, the smallest 'cut edge fraction' ξ for all the edges surrounding a point is found (see Fig. 4). The modified mass-matrix is then given by

$$M^i_* = \frac{1 + \xi_{min}}{2} M^i. \tag{9}$$

Note that the value of the modified mass-matrix can never fall below half of its original value, implying that timestep sizes will always be acceptable. For edges that are along embedded boundaries (see Fig. 5), the original edge-coefficient must be modified to reflect the loss of volume resulting from the presence of the boundary. In principle, a complex quadrature may be used to recompute the edge-coefficients. A very simple approach to obtain an approximation to these values is again via the 'cut edge fraction'

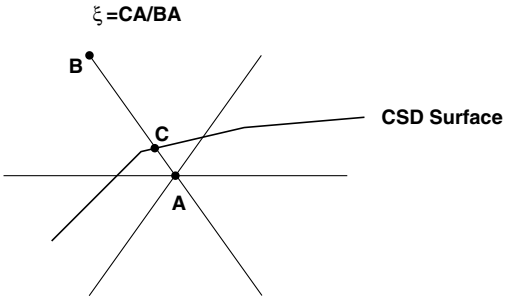

Fig. 4. Cut edge fraction.

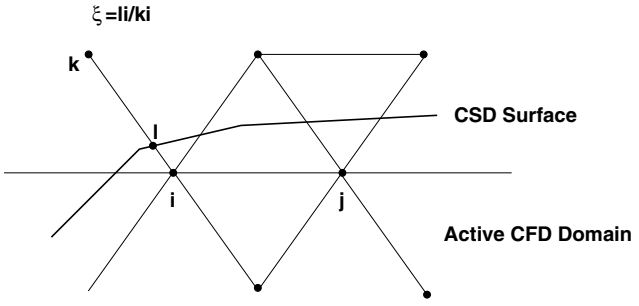

Fig. 5. Modification of boundary edge coefficients.

technique. Given the 'cut edge fraction' of the end-points, we simply modify C^{ij} as follows:

$$C^{ij}_{*} = \frac{1 + max(\xi_i, \xi_j)}{2} C^{ij}. \tag{10}$$

2.2.2. 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 velocity is the surface itself, and not the closest boundary point. As shown in Fig. 6, 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 iteration (for iterative implicit schemes) or timestep (for explicit schemes).

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 Fig. 7).

2.3. 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

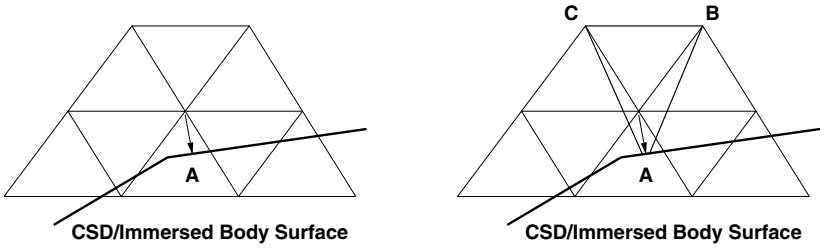

Fig. 6. Extrapolation of velocity.

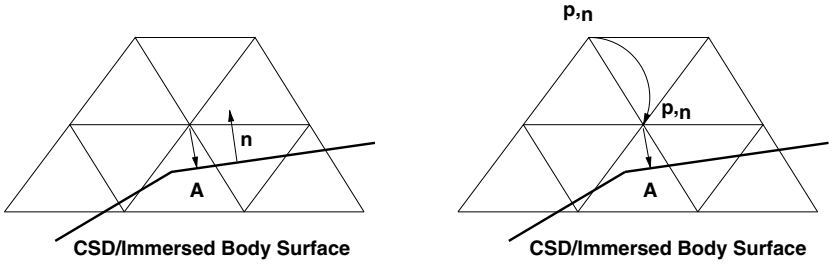

Fig. 7. Extrapolation of normal pressure gradient.

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.

2.4. 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. Fig. 8 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.

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

$$\rho_m = \rho_i, p_m = p_i, \mathbf{v}_m = \mathbf{v}_i - 2[(\mathbf{v}_i - \mathbf{w}_{\text{csd}}) \cdot \mathbf{n}]\mathbf{n}, \tag{11}$$

where $\mathbf{w}_{\text{csd}}$ is the velocity of the crossed CSD face, ρ the density, $\mathbf{v}$ the flow velocity, p the pressure and $\mathbf{n}$ the unit surface normal of the face. Proper handling of the interpolation for degenerate cases is also required, as the element used for the interpolation might either be crossed (Fig. 9a) or not exist (Fig. 9b).

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:

$$p_m = p_i - \rho_i \frac{[\mathbf{v}_i - (\mathbf{v}_i - \mathbf{w}_{\text{csd}}) \cdot \mathbf{n}]^2}{R_i} \Delta, \rho_m = \rho_i \left(\frac{p_m}{p_i}\right)^{\frac{1}{\gamma}}, \tag{12}$$

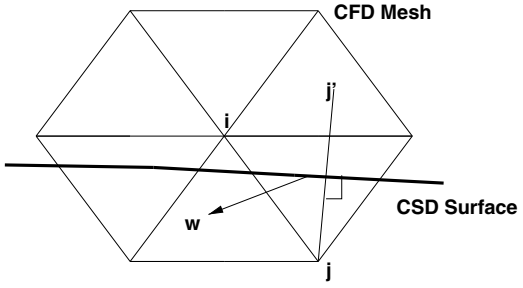

Fig. 8. Higher-order boundary conditions.

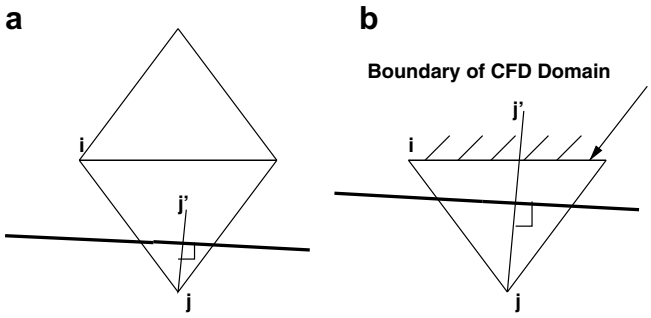

Fig. 9. Problem cases.

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 (3-D) of curvature at each CSD point [18]. The radius of curvature plays an important role for large elements but this influence can be diminished by the use of automatic h-refinement.

For problematic cases such as the one shown in Fig. 9, the interpolation will be such that the point at which the information is interpolated may not be located at the same normal distance from the wall as the point where information is required.

With the notation of Fig. 10, and assuming a linear interpolation of the velocities, the velocity values for the viscous (i.e. no-slip) case are interpolated as

$$\mathbf{w} = (1 - \xi_w)\mathbf{v}_c + \xi_w \mathbf{v}_i; \xi_w = \frac{h_o}{h_o + h_i}, \tag{13}$$

i.e.

$$\mathbf{v}_c = \frac{1}{1 - \xi_w}\mathbf{w} - \frac{\xi_w}{1 - \xi_w}\mathbf{v}_i. \tag{14}$$

Here $\mathbf{w}$ is the average velocity of the crossed CSD face, $\mathbf{v}_i$ the interpolated flow velocity and the distance factor $\xi_w \leqslant 0.5$.

2.5. 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 a fast spatial search data structure, such as an octree or a bin for the CSD faces (see Fig. 11). Without loss of generality, let us assume a bin for 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, 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

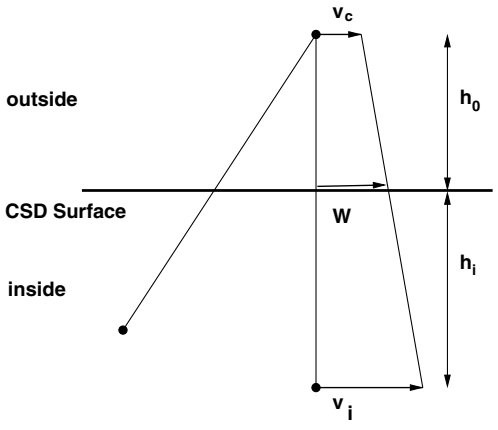

Fig. 10. Navier–Stokes boundary condition.

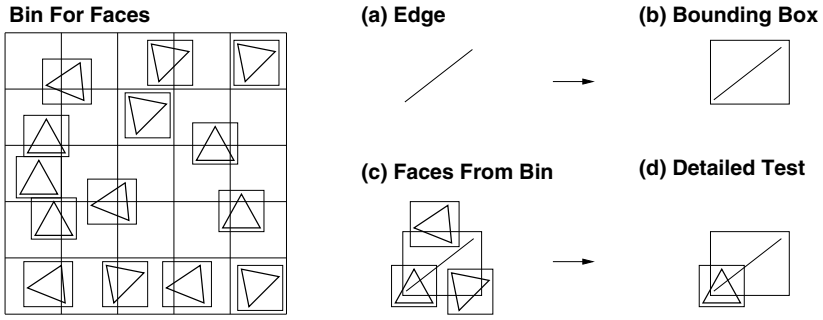

Fig. 11. Bin storage of faces and search.

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 information is required to apply boundary conditions for the points close to the embedded surface. For cases where the embedded surfaces only cut a small portion of the CFD edges, a considerable speedup may be realized by removing from the list of edges tested all those that fall outside the global bounding box of the CSD faces. The resulting list of edges to be tested in depth may be reduced further by removing all edges whose bounding boxes do not fall into an octree or bin filled with faces covering that spatial region. One typically finds that using these two filters, the list of edges to be tested in detail has been reduced by an order of magnitude.

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.

3. 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 Fig. 12). The best way to treat such points is to simply deactivate them.

This deactivation concept can be extended further in order to avoid unnecessary work for regions inside solid objects. Several approaches have been 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.

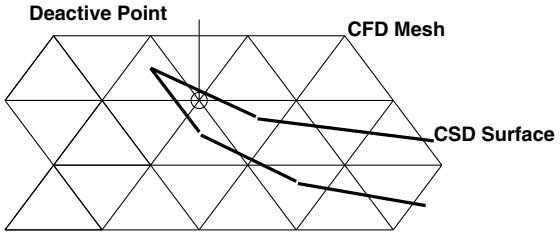

Fig. 12. Deactive point.

(b) *Automatic seed points*: For external aerodynamics problems, one can define seed points on the boundaries of the bounding box. At the beginning, all points are marked as deactive. Starting from the external boundaries, points are activated using an advancing front technique into the domain. The procedure only activates neighbour points if at least one of the edges is active and attached to an active point.

(c) *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 (Fig. 13). 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

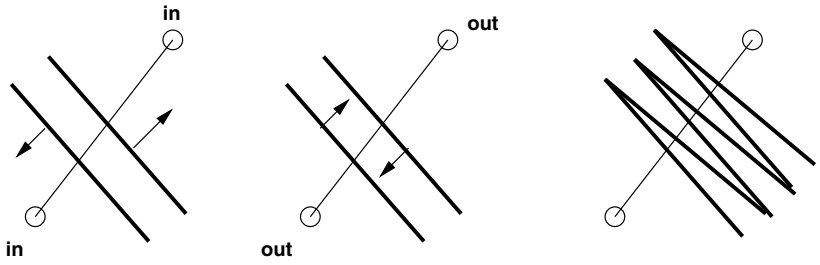

Fig. 13. Edges with multiple crossing faces.

(active). Once the points have been marked as active/inactive, the element and edge-groups required to avoid memory contention (i.e. allow vectorization) are inspected in turn. As with spacemarching [35,38,48,49] 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.

4. Extrapolation of the solution

For problems with moving boundaries, mesh points can switch from one side of a surface to another or belong/no longer belong to an immersed body (see Fig. 14). For these cases, the solution must be extrapolated from the proper state. The conditions that have to be met for extrapolation are as follows:

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

For incompressible flow problems the simple extrapolation of the solution from one point to another (or even a more sophisticated extrapolation using multiple neighbours) will not lead to a divergence-free velocity field. Therefore, it may be necessary to conduct a local 'divergence cleanup' for such cases.

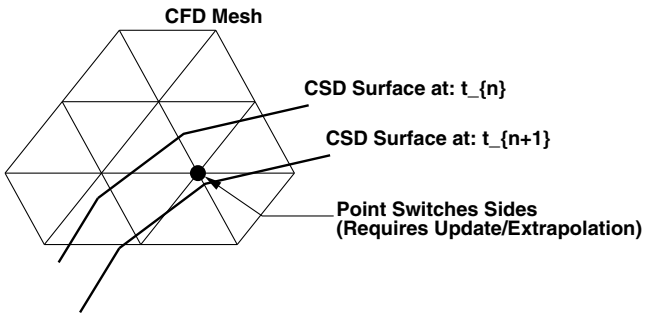

Fig. 14. Extrapolation of solution.

5. 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 [6,31,37]. For embedded CSD triangulations, the mesh can be refined automatically close to the surfaces [1,40]. One can define a number of refinement criteria, of which the following have proven to be the most useful:

- refine the elements with edges cut by CSD faces to a certain size/level;
- refine the elements so that the curvature given by the CSD faces can be resolved (e.g. 10 elements per 90° bend);
- refine the elements close to embedded CSD corners with angles up to 50° to a certain size/level;
- refine the elements close to embedded CSD ridges with angles up to 15° to a certain size/level.

The combination of adaptive refinement and embedded/immersed grid techniques has allowed to automate completely certain application areas, such as external aerodynamics. Starting from a triangulation (e.g. an STL data set obtained from CAD for a design or a triangulation obtained from a scanner for reverse engineering), a suitable box is automatically generated and filled with optimal space-filling tetrahedra. This original mesh is then adaptively refined according to the criteria listed above. The desired physical and boundary conditions for the fluid are read in, and the solution is obtained. In some cases, further mesh adaptation based on the physics of the problem (shocks, contact discontinuities, shear layers, etc.) may be required, but this can also be automated to a large extent for class-specific problems. Note that the only user input consists in flow conditions. The many hours required to obtain a watertight, consistent surface description have been eliminated by the use of adaptive, embedded flow solvers.

6. Load/flux transfer

For fluid–structure interaction problems, the forces exerted by the fluid on the embedded surfaces or immersed

bodies need to be evaluated. For *immersed bodies*, this information is given by the sum of all forces, i.e. by Eq. (4). For *embedded surfaces*, the information is obtained 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. 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;
- as can be seen from Fig. 15, a face may have multiple (F1) or no (F2) crossing edges; 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 as follows:

- 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 Fig. 16);
- 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 or bins for the faces. We remark that for many fluid–structure interaction cases this step can be more time-consuming than finding the crossed edges and modifying boundary conditions and arrays, particularly if the characteristic size of the embedded CSD faces is smaller than the characteristic size of the surrounding tetrahedra of the CFD mesh, and the embedded CSD faces are close to each other and/or folded.

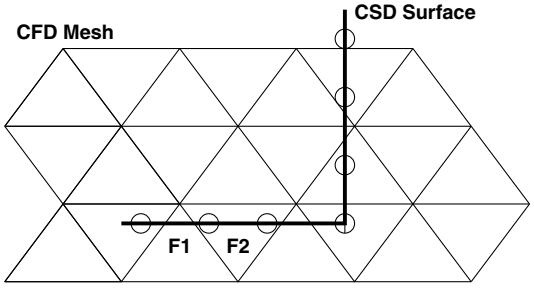

Fig. 15. Transfer of stresses/fluxes.

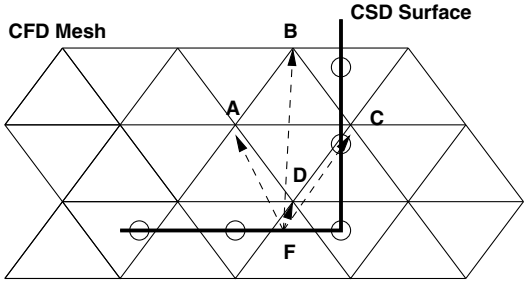

Fig. 16. Transfer of stresses/fluxes.

7. 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 (Fig. 17a), a gap or crack is immediately associated with minuscule grid sizes, small timesteps and increased CPU costs.

For embedded grids (Fig. 17b), the gap or crack may not be seen. A simple solution 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:

$$C_k^{ij} = \eta C_{0k}^{ij}; \quad \eta = \delta/h. \tag{15}$$

Gaps are detected by considering the edges of elements with multiple crossed edges. If the faces crossing these edges are different, a test is 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.

8. Direct link to particles

One of the most promising ways to treat discontinua is via so-called discrete element methods (DEMs) or discrete

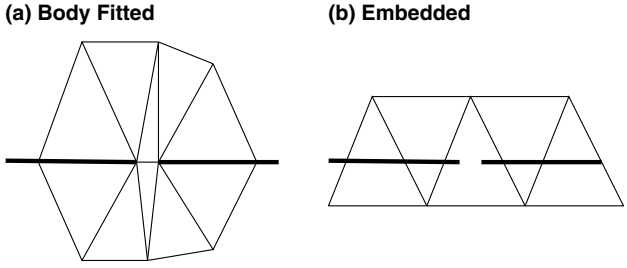

Fig. 17. Treatment of gaps.

particle methods (DPMs). A considerable amount of work has been devoted to this area in the last two decades, and these techniques are being used for the prediction of soil, masonry, concrete and particulates [16]. The filling of space with objects of arbitrary shape has also reached the maturity of advanced unstructured grid generators [39], opening the way for widespread use with arbitrary geometries. Adaptive embedded grid techniques can be linked to DPMs in a very natural way. The discrete particle is represented as a sphere. Discrete elements, such as polyhedra, may be represented as an agglomeration of spheres. The host element for each one of the discrete particles is updated every timestep and is assumed as given. All points of host elements are marked for additional boundary conditions. The closest particle to each of these points is used as a marker. Starting from these points, all additional points covered by particles are marked (see Fig. 18).

All edges touching any of the marked points are subsequently marked as crossed. From this point onwards, the procedure reverts back to the usual embedded mesh or immersed body techniques. The velocity of particles is either imposed at the endpoints of crossed edges (embedded) or for all points inside and surrounding particles (immersed).

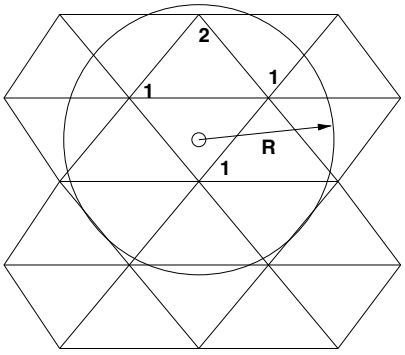

Fig. 18. Link to discrete particle method.

9. Coordinate movement for display

The display of information from a CFD field solver with embedded CSD faces or immersed bodies 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 (Fig. 19b). In order to achieve a more precise, continuous surface representation, the points close to embedded surfaces are moved to the surface itself before display (Fig. 19c and d). 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.

10. Examples

Adaptive embedded and immersed unstructured grid techniques have been used extensively for a number of years now. In this section, we include a few of these. The aim is to highlight the considerable range of applicability of these methods, show their limitations, as well as the

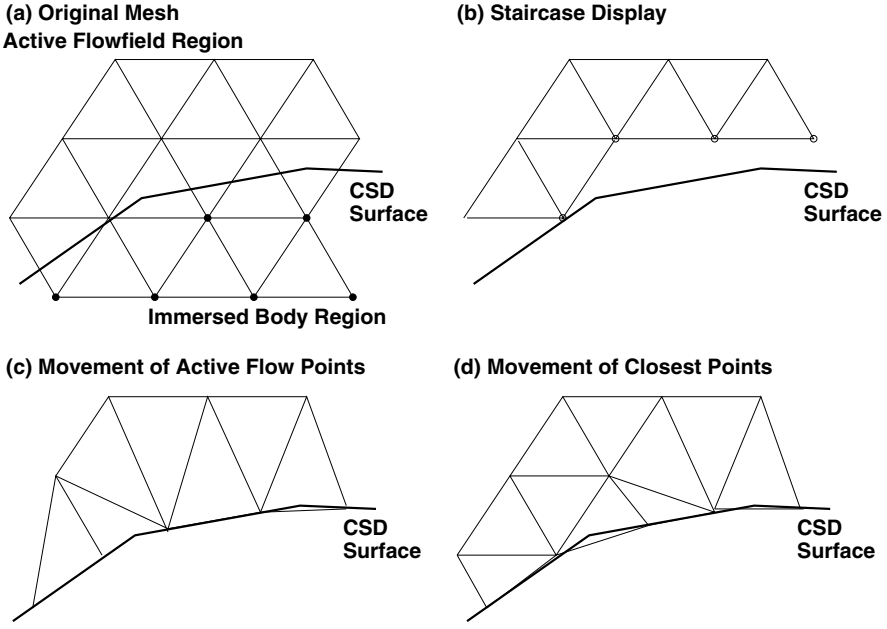

Fig. 19. Coordinate movement for display.

combination of body-fitted and embedded/ immersed techniques. We start with compressible, inviscid flow, where we consider the classic Sod shock tube problem, a shuttle ascend configuration and two fluid–structure interaction problems. We then consider incompressible, viscous flow, where we show the performance of the different options in detail for a sphere. The contaminant transport calculation for a city is then included to show a case where obtaining a body-fitted, watertight geometry is nearly impossible. Finally, we show results obtained for complex endovascular devices in aneurysms, as well as the external flow past a car.

10.1. Sod shock tube

The first case considered is the classic [58] 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 Fig. 20a. Fig. 20b 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

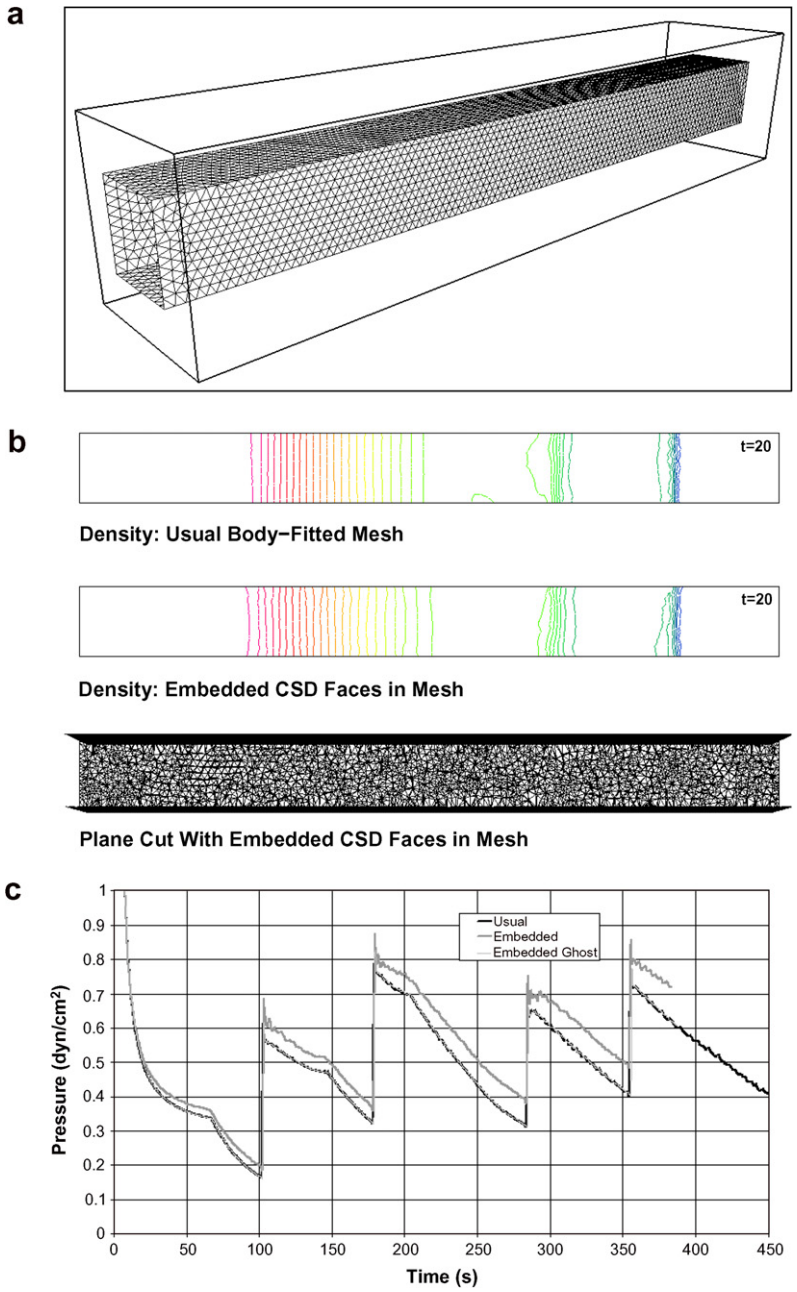

Fig. 20. Shock tube problem: (a) embedded surface; (b) density contours. (c) Pressure time history.

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 Fig. 20c, which shows the pressure time history for a point located in the high pressure side (left of the 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 some damping.

10.2. Shuttle ascend configuration

The second example considered is the Space Shuttle Ascend configuration shown in Fig. 21a. 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

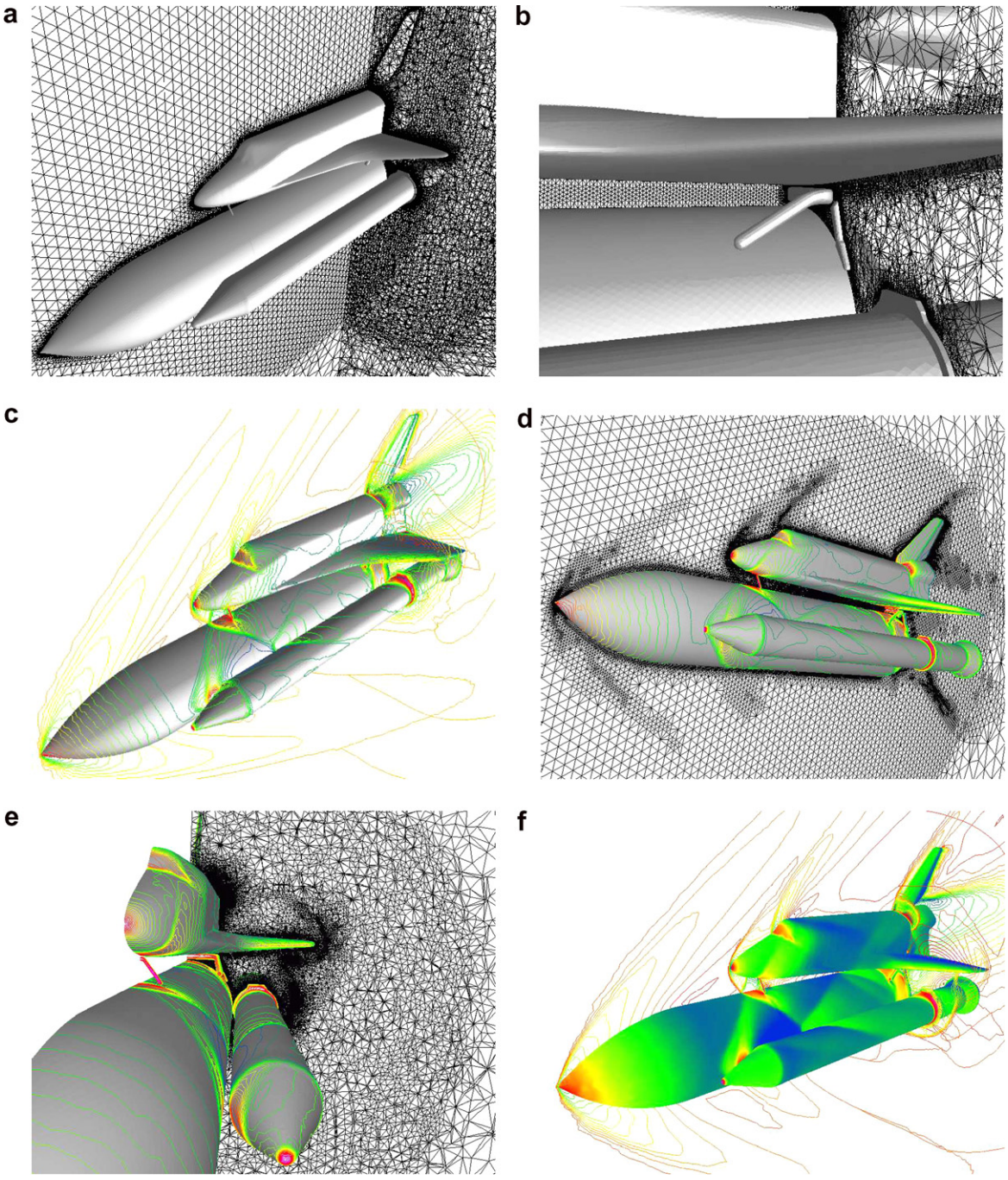

Fig. 21. (a, b) Shuttle: general view and detail, (c, d) surface pressure and field Mach-Nr., surface pressure and mesh (cut plane), (e, f) Surface pressure and mesh (cut plane), surface pressure and field Mach-Nr.

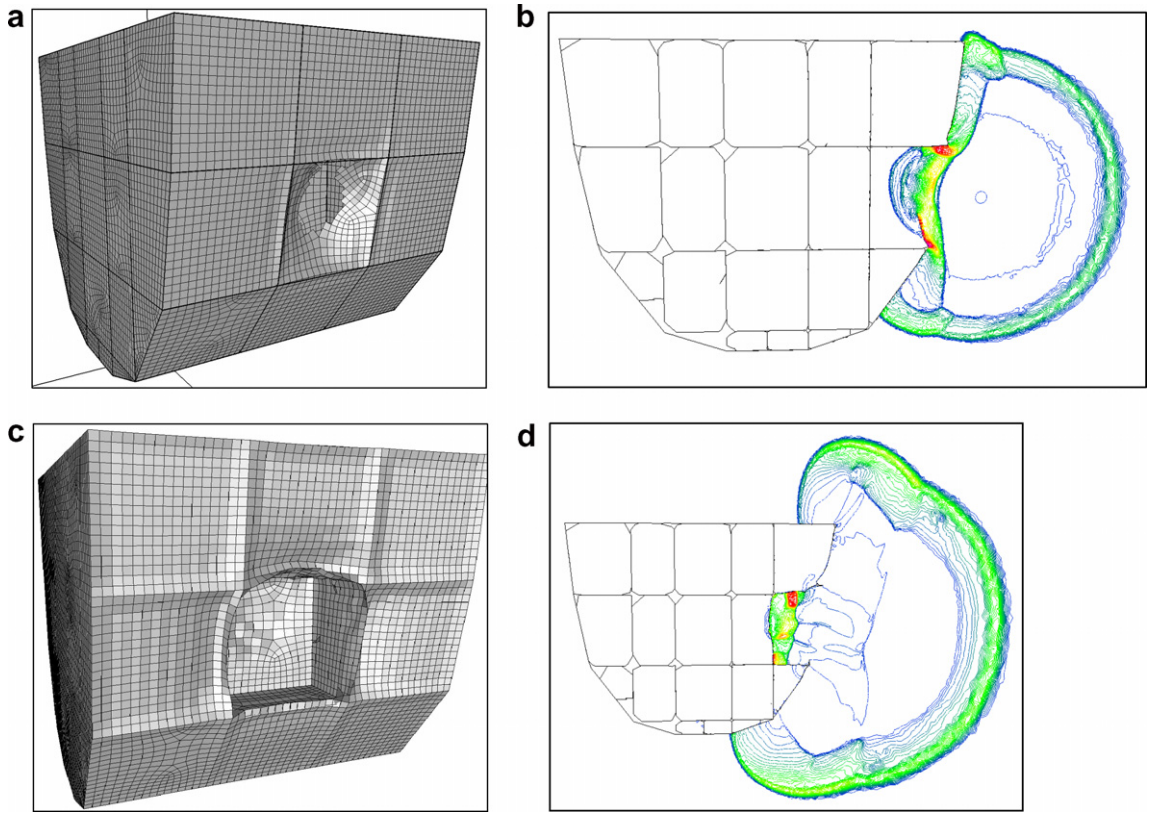

Fig. 22. Surface and pressure in cut plane at (a) 20 ms and (b) 50 ms.

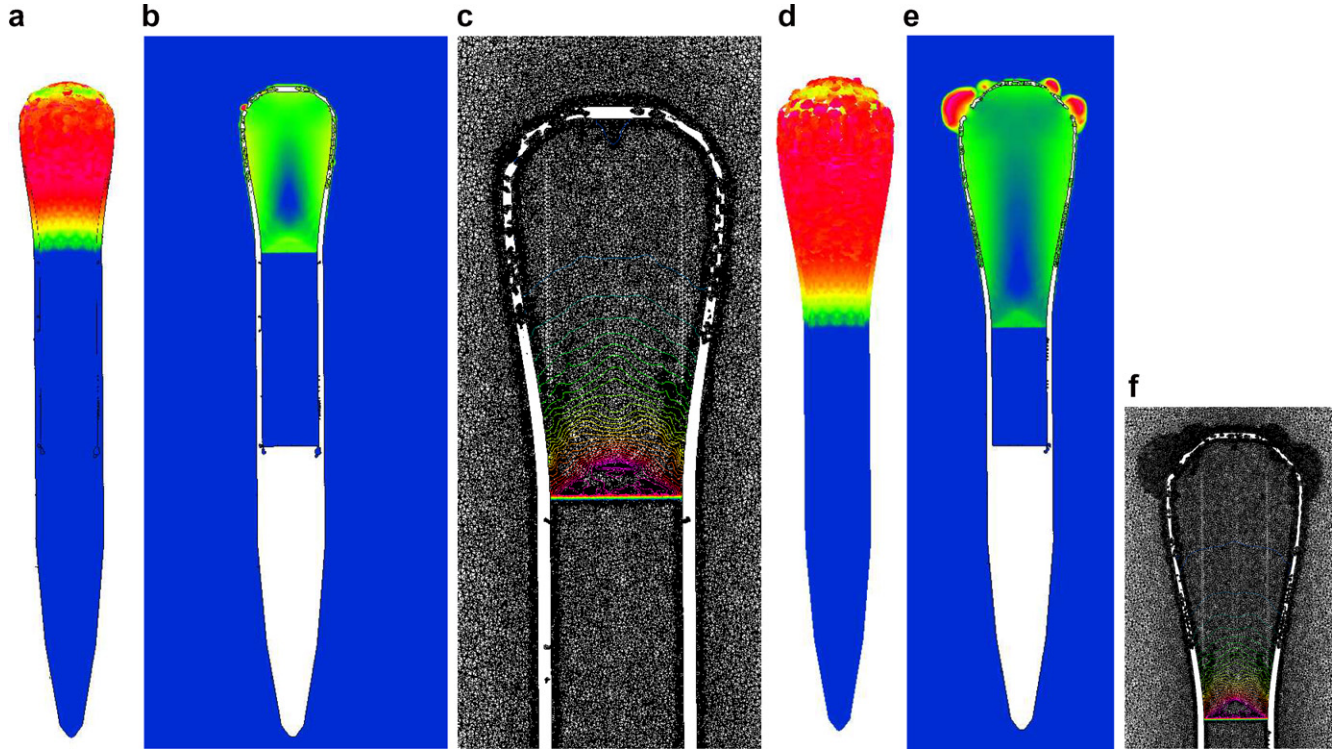

Fig. 23. CSD/flow velocity and pressure/mesh at (a–c) 68 ms and (d–f) 102 ms.

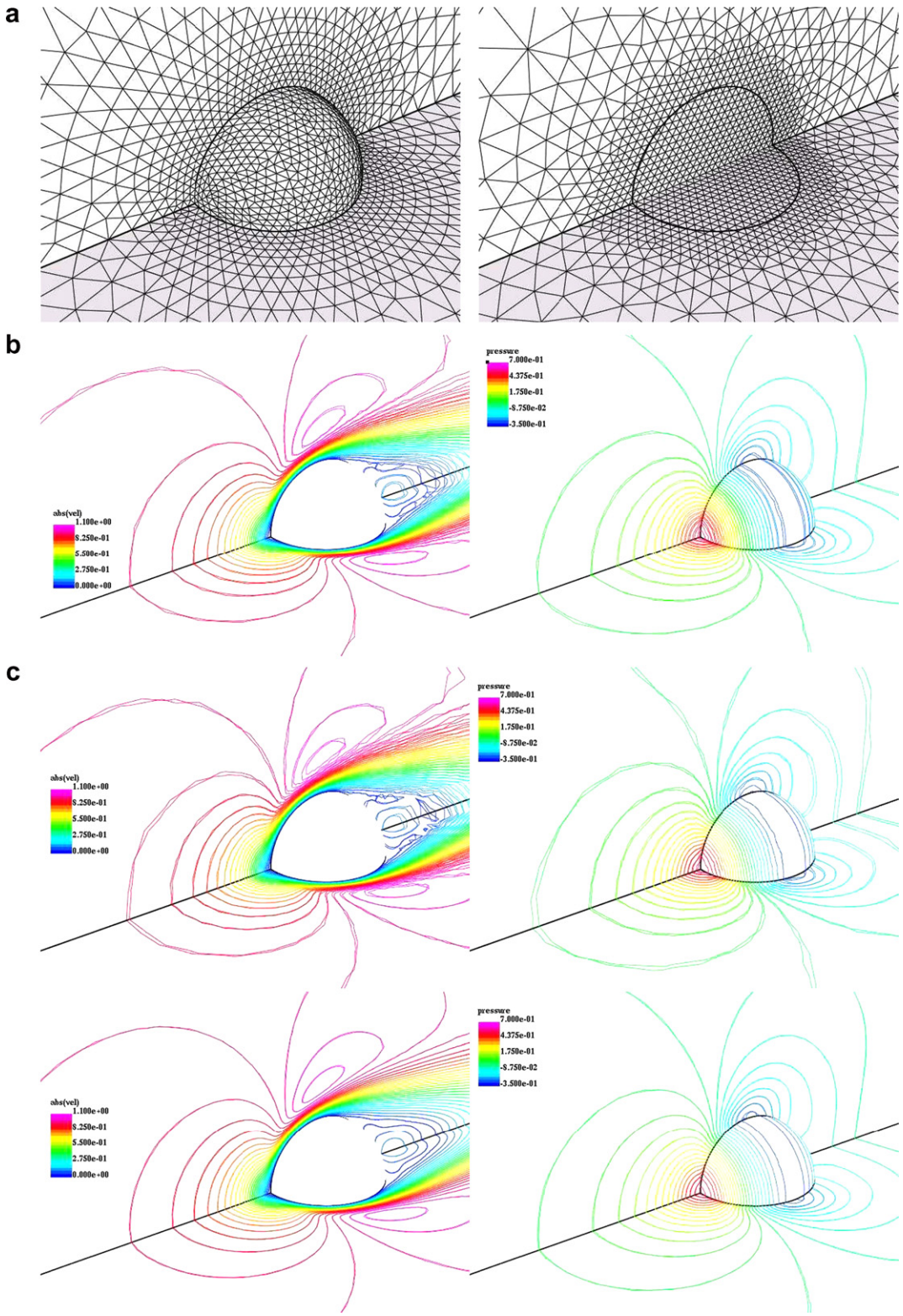

Fig. 24. (a) Sphere: surface grids (body fitted, embedded/immersed). (b) Coarse vs. fine body-fitted (left: $|\mathbf{v}|$, right: p). (c) Body-fitted vs. embedded 1 (top: coarse; bottom: fine). (d) Body-fitted vs. embedded 2 (top: coarse; bottom: fine). (e) Body-fitted vs. immersed (top: coarse; bottom: fine). (f) Velocity/pressure along line: $y,z = 0.0$ (top: coarse; bottom: fine).

(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 five levels. This yielded a

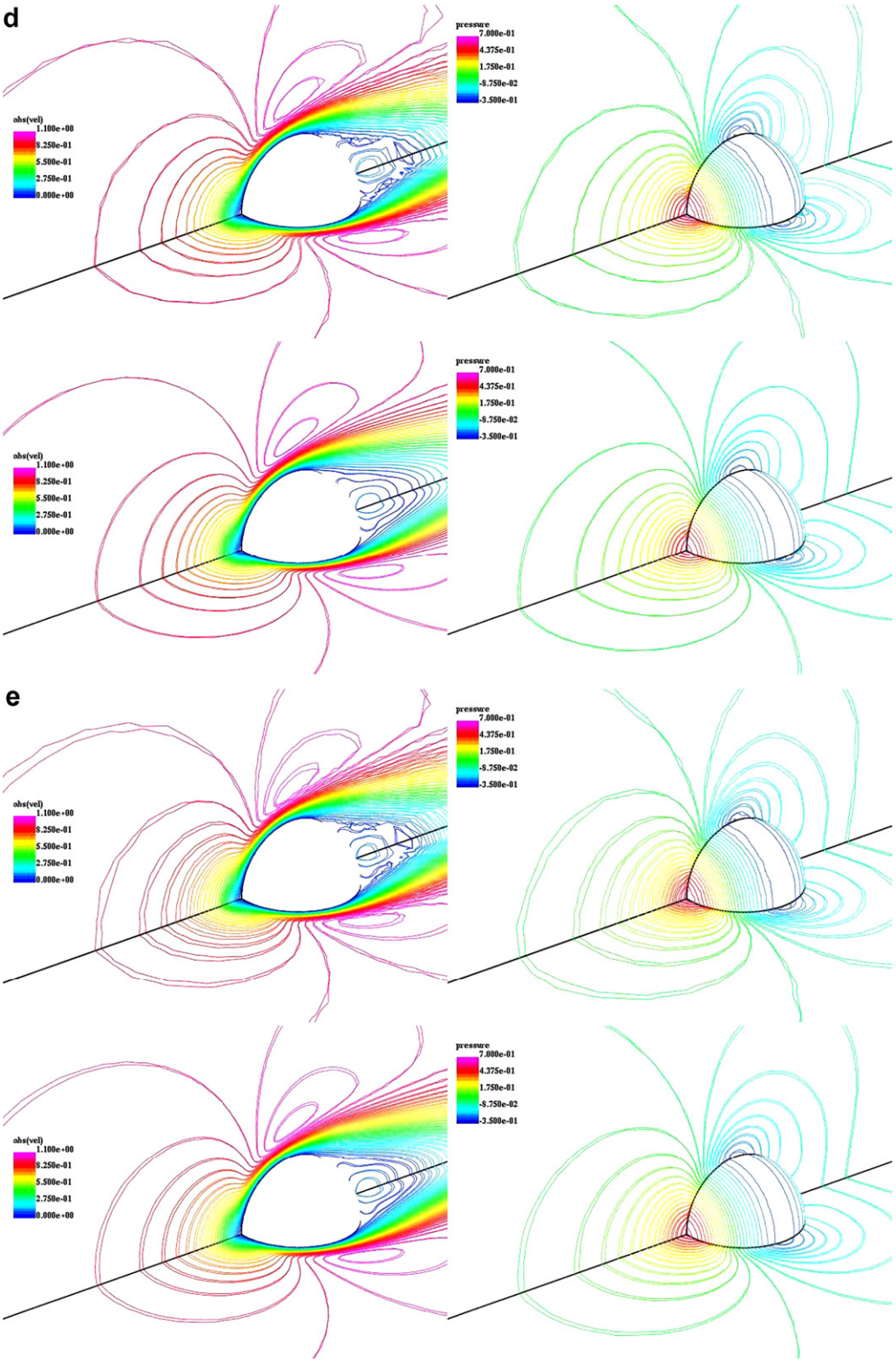

Fig. 24 (*continued*)

mesh of approximately 16.9 Mtet. The grid obtained in this way, as well as the corresponding solution are shown in Fig. 21b and c. Note that all geometrical details have been properly resolved. The mesh was subsequently refined based on a modified interpolation theory error indicator [33,34] of the density, up to approximately 28 Mtet. This physics-based mesh refinement is evident in Fig. 21c and d.

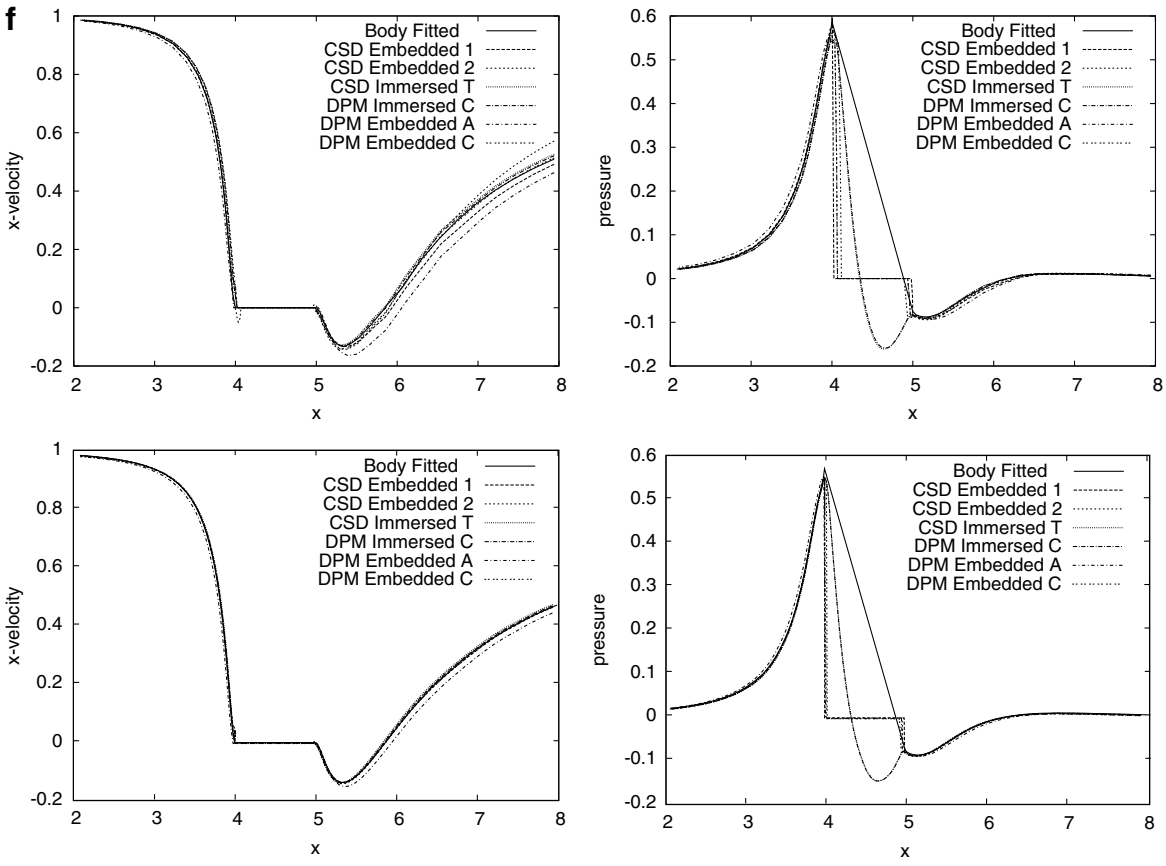

Fig. 24 (*continued*)

10.3. Blast interaction with a generic ship hull

Fig. 22 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. Fig. 22a–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 interpenetration 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 embedded CSD capability for unstructured grids.

10.4. Generic weapon fragmentation

Fig. 23 shows a generic weapon fragmentation study. The CSD domain was modeled with approximately 66 Khex elements corresponding to 1,555 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 [36]. The CFD mesh was refined to three levels in the vicinity of the solid surface. Additionally, the mesh was refined based on the modified interpolation error indicator [33,34] using the density as indicator variable.

Adaptive refinement was invoked every five timesteps during the coupled CFD/CSD run. The CFD mesh started with 39 Mtet, and ended with 72 Mtet. Fig. 23a–f show the structure as well as the pressure contours in a cut plane at two times during the run. The detonation wave is clearly visible, as well as the thinning of the structural walls and the subsequent fragmentation.

10.5. Flow past a sphere

This simple case is included here as it offers the possibility of an accurate comparison of the different techniques discussed in this paper. The geometry considered is shown in Fig. 24. Due to symmetry considerations only a quarter of the sphere is treated. The physical parameters were set as follows: $D = 1$, $\mathbf{v}_\infty = (1,0,0)$, $\rho = 1.0$, $\mu = 0.01$, yielding a

Reynolds-number of $Re = 100$. Two grids were considered: the first had an element size of approximately $h = 0.0450$ in the region of the sphere, while the corresponding size for the second was $h = 0.0225$. This led to grids with approximately 140 Kels and 1.17 Mels, respectively. The coarse mesh surface grids for the body fitted and embedded options are shown in Fig. 24a. We implicitly assumed that the body-fitted results were more accurate and therefore considered them as the 'gold standard'. Fig. 24b show the same surface contour lines of the absolute value of the velocity, as well as the pressures, obtained for the body-fitted coarse and fine grids. Note that although some differences are apparent, the results are quite close, indicating a grid-converged result on the fine mesh. The drag coefficients for the two body-fitted grids were given by $c_d = 1.07$ and $c_d = 1.08$, respectively, in good agreement with experimental results (Schlichting et al. [57]). Fig. 24c–e show the same surface contour lines for the body-fitted and the different embedded/immersed options for the two grids. Note that the contours are very close, and in most cases almost identical. This is particularly noticeable for the second-order embedded and the immersed body cases. Fig. 24f depict the x-velocity and pressure along the line $y, z = 0$ (i.e. the axis of symmetry). As seen before in the contour lines, the results are very close, indicating that even the first-order embedded scheme has converged. For more information, see Löhner et al. [44].

10.6. Dispersion in an inner city

This case, taken from Camelli and Löhner [9], considers the dispersion of a contaminant cloud near Madison Square Garden in New York city. The commercial building database Vexcel (see Hanna et al. [26]) was used to recover the geometry description of the city. Fig. 25a and b shows the wireframe of the information contained in the database.

The area of buildings covered in the simulation is 2.7 km by 1.9 km. The computational domain is 3.3 km by 2.6 km and a height of 600 m. The building database simply consists of disjoined polygons that can cross and are not watertight. Cleaning up this database, even with semi-automatic tools, in order to obtain a body-fitted mesh would have taken many man-months. The adaptive embedded approach reduced dramatically the man-hours required for reconstructing the geometry of buildings, making a run like the one shown here possible. A VLES simulation

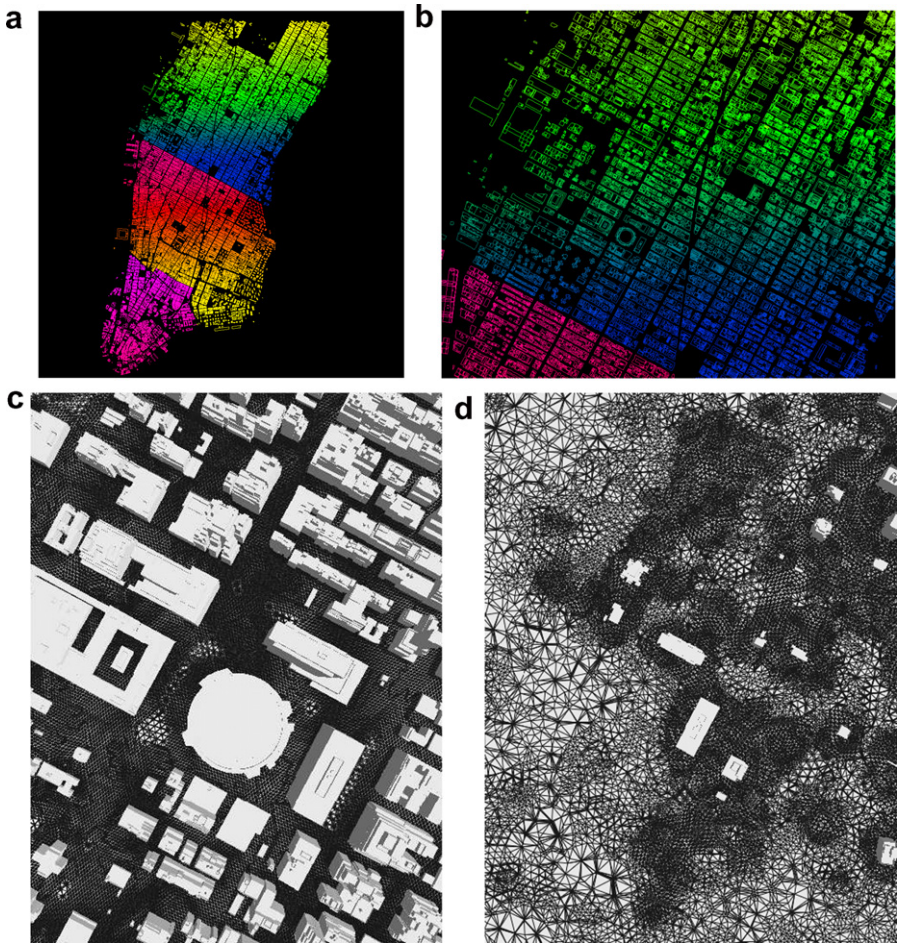

Fig. 25. (a, b) Blueprint of Manhattan, zoom into MSG. (c, d) Cut planes at 5 and 100 m above ground level. (e, f) SW case, $z = 100$ m: velocity vectors and contours of vertical velocity. (g–j) SW case: plume patterns for continuous release.

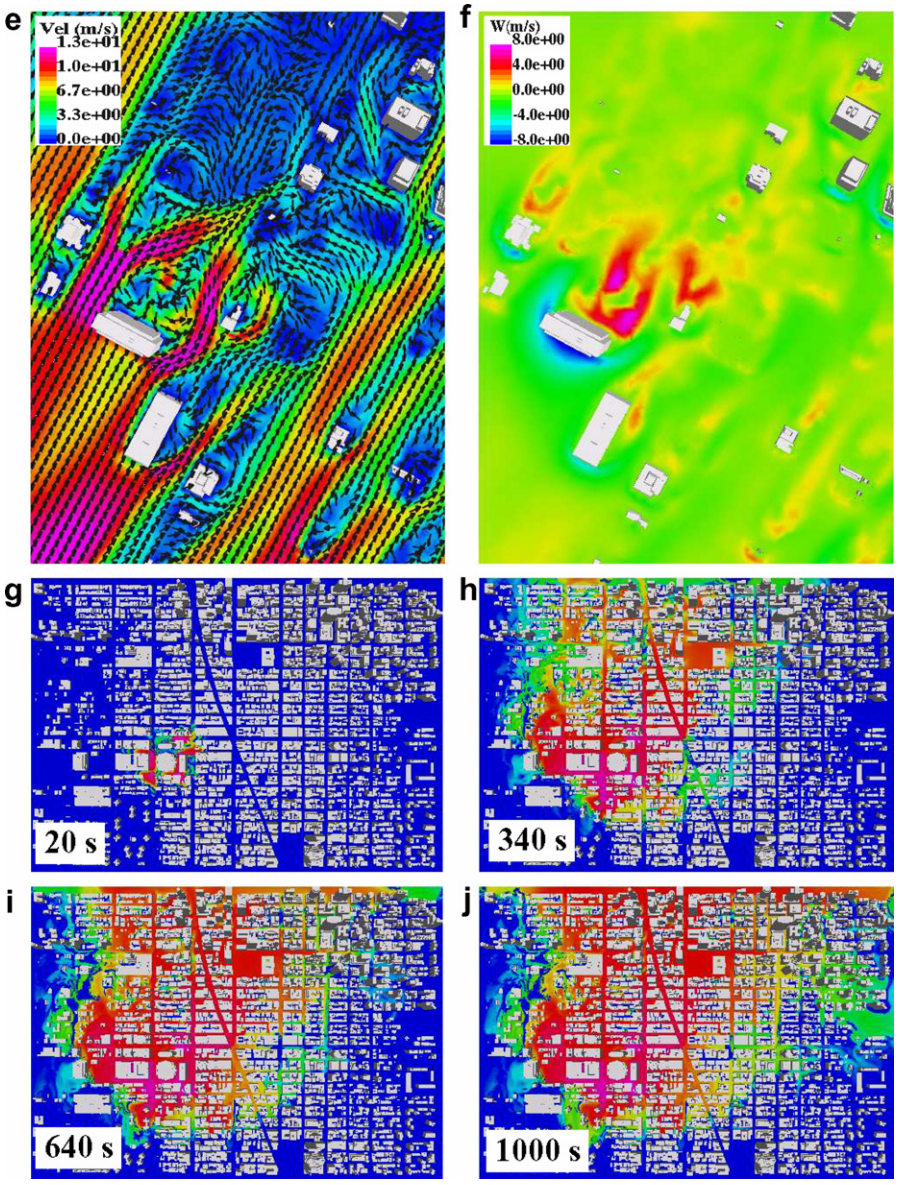

Fig. 25 (*continued*)

was performed with a mesh of 24 Mtet elements. The element size was of 2 m close to the building surfaces. Two cut planes of the volume mesh are shown in Fig. 25c and d at 5 and 100 m above ground level. These two cut planes illustrate clearly the use of adaptive gridding based on edges crossed by the embedded surfaces, and the relatively high resolution of the mesh used close to buildings and the ground. A typical velocity field obtained for a South-Westerly wind is shown in Figs. 25e and f, and the dispersion pattern for a continuous release may be discerned from Fig. 25g and h.

10.7. Complex endovascular devices

Stents are currently being considered as an interesting alternative treatment for aneurysms. While the pipeline from images to body-fitted grids to flow calculation to visualization is a rather mature process [10,11,13], the placement of stents or coils represents considerable challenges. In Cebral and Löhner [13] the use of embedded grids was first proposed as a way to circumvent lengthy input times without compromising accuracy. Fig. 26 shows the comparison of a body-fitted and embedded stent calculation for an idealized aneurysm geometry. The body-fitted idealized aneurysm is obtained by first merging a sphere with a cylinder, obtaining the isosurface of distance $\delta = 0$ [10–12], and meshing this discrete surface (Fig. 26a). The stent is described as a set of small spheres (beads), and is placed in the idealized aneurysm domain. The isosurface of distance $\delta = 0$ is again used as the starting point to mesh the complete domain (Fig. 26b). For the embedded case, the spheres describing the stent are simply placed in their position, and the proper boundary conditions are applied as described above. Fig. 26c shows the comparison of

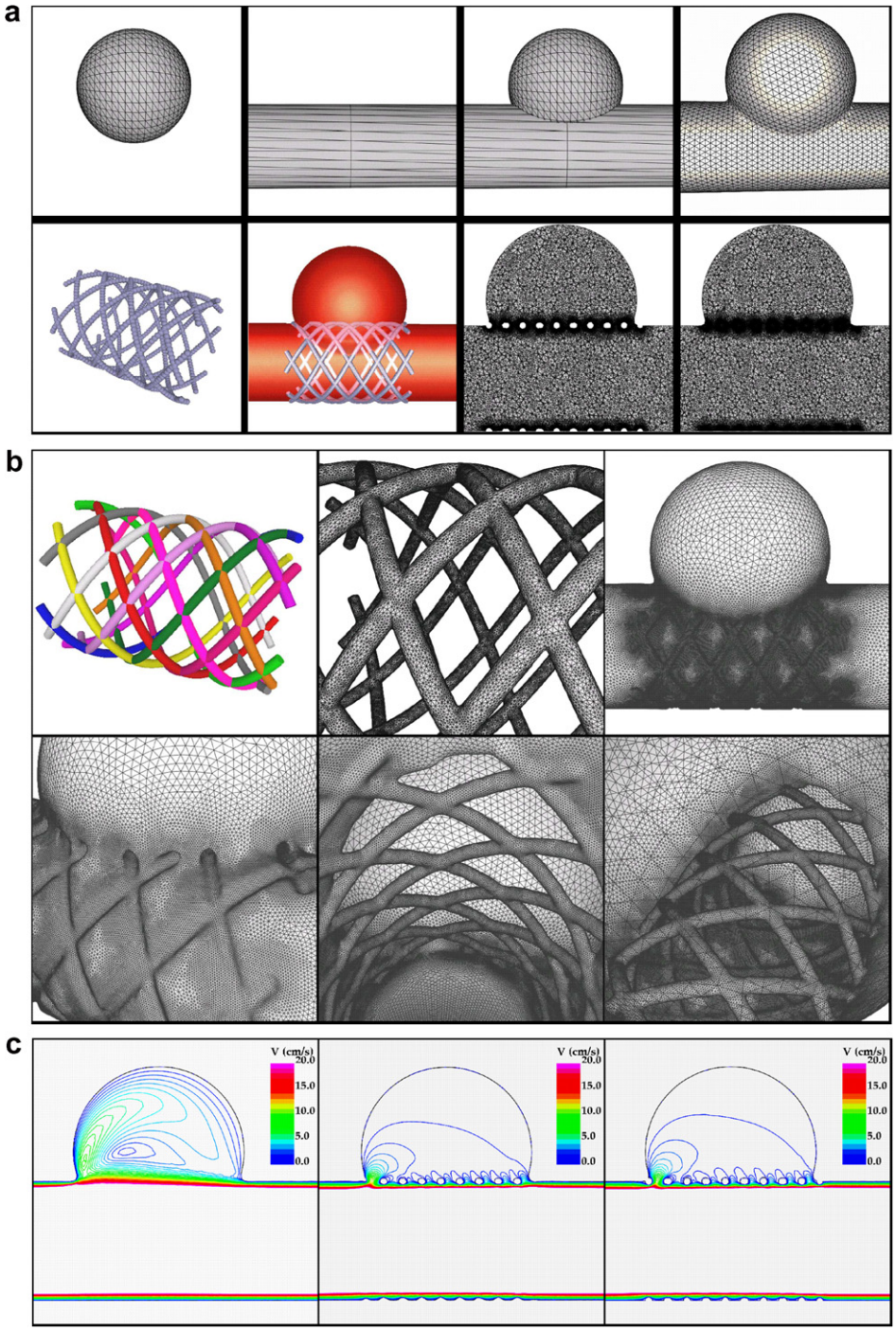

Fig. 26. (a) Idealized aneurysm with stent. (b) Idealized aneurysm with stent: body-fitted mesh. (c) Comparison of velocities in mid-plane.

velocities for the idealized aneurysm with and without the stent. One can see that the velocities for the body-fitted and embedded approaches are nearly identical. This procedure has also been used successfully for coils [13].

10.8. Flow past a VW GOLF 5

This case considers the flow past a typical passenger car, and is taken from Tilch et al. [59]. For external vehicle aerodynamics, the car industry is contemplating at present turnaround times of 1–2 days for arbitrary configurations. For so-called body fitted grids, the surface definition must be water-tight, and any kind of geometrical singularity, as well as small angles, should be avoided in order to generate a mesh of high quality. This typically presents no problems for the main 'shape' of the car (the part visible to a street-side observer), but can be difficult to obtain in such a short time for the underhood and undercarriage of a typical car

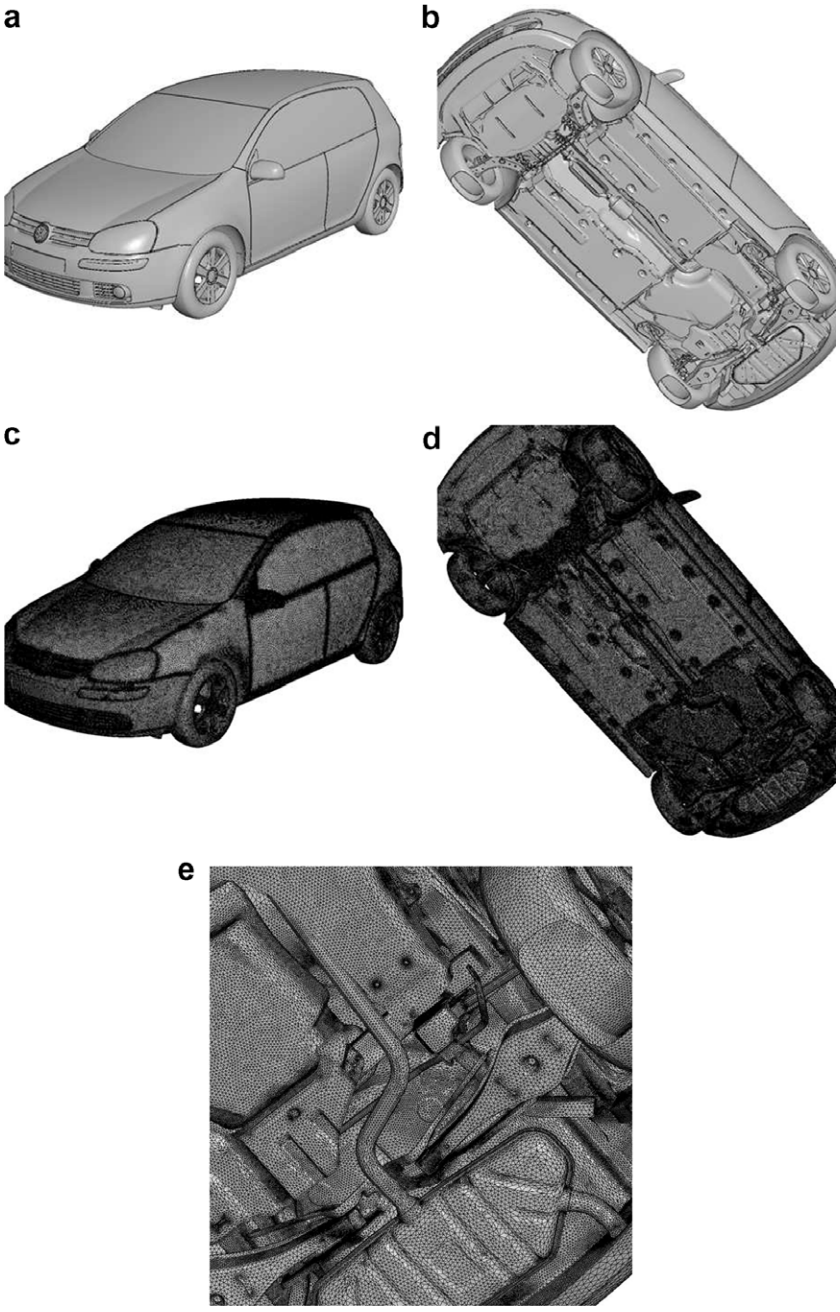

Fig. 27. (a, b) VW GOLF 5: surface definition (body fitted). (c, d) VW GOLF 5: surface grids (body fitted). (e, f) VW GOLF 5: surface grids (underhood detail). (g, h) VW GOLF 5: surface grids (body fitted, embedded). (i) VW GOLF 5: surface grid (body fitted and embedded). (j) Cut mesh in mid-plane. (k) Undercarriage detail. (l) Pressure (contours) on ground and x-planes, velocity (shaded) in back x-plane. (m) Velocities in mid-plane (top: body-fitted; bottom: body-fitted + embedded).

or truck. Experience indicates that even with sophisticated software toolkits, manual cleanup in most cases takes several days for a complete car. At first sight, the solution of high Reynolds-number flows with grids of this type seems improper. Indeed, for the external shape portion the surface is smooth, and the interplay of pressure gradient and viscous/advective terms is what decides if separation will occur. Therefore, for this portion of the vehicle, a highly detailed, body-fitted RANS is considered mandatory. However, for the underhood and undercarriage, many abrupt changes in curvature occur, the flow is massively separated, and an LES run seems sufficient. For embedded grids, this presents no problem. One is therefore in a rather fortunate position: the region where the geometry is the 'dirtiest' happens to be the region where isotropic grids are sufficient, making this region a good candidate for embedded grids. The key idea is then to obtain, quickly, the external shape of the vehicle and grid it with typical body-fitted RANS grids. Note that this portion of the surface is typically 'clean', i.e. a turnaround of 1–2 days is

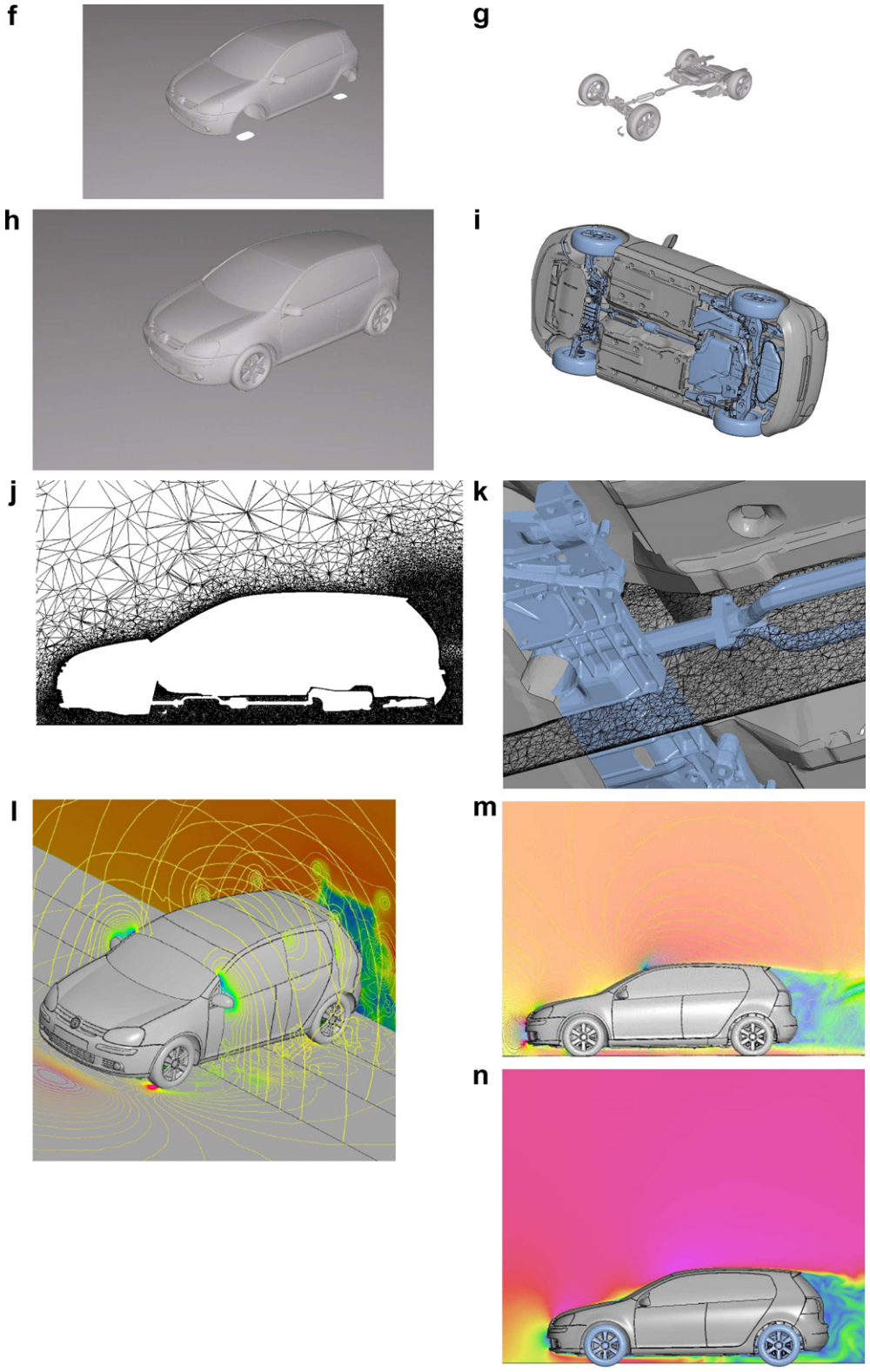

Fig. 27 (*continued*)

possible. The underhood and undercarriage, on the other hand, is then inserted into the RANS mesh generated for the external shape of the vehicle as an embedded surface. As such, it can have crossing faces (stemming, for example, from different parts of the undercarriage), and does not require elements of very high quality. A run is then conducted with the embedded mesh.

The example shown here compares a body-fitted and an embedded run of the type described above for a typical passenger car. The body-fitted mesh was obtained after

several weeks of cleanup, and may be seen in Fig. 27a and b. Note that all the undercarriage details have been taken into account. The corresponding surface grids are shown in Fig. 27c–e. For the embedded case, the starting point was given by two NASTRAN-files which came from different departments at VW. The surface which was meshed using the body-fitted approach was given by 12 patches (one surface for the top, one for the bottom, two for the mirrors), whereas the surface which was treated with the embedded approach was given by 106 parts, most of which were single patches. The complete surface triangulation used to define the car had 1.1 Mtria. The body-fitted mesh consisted of approximately 5.68 Mpts and 32.03 Mtets. Five RANS layers were used. For the car body, the first point was $y_w = 0.758$ mm away from the wall, and the mesh was increased by a factor of $c_i = 1.5$ between layers. For the ground, the first point was $y_w = 0.914$ mm away from the ground, and the mesh was increased by a factor of $c_i = 1.4$ between layers. The surface of the body-fitted domain, the embedded surface, as well as the complete vehicle, are shown in Fig. 27f–i. A closeup of the undercarriage, together with the mesh in a cut plane, is shown in Fig. 27j. The physical parameters were set as follows: $\mathbf{v}_\infty = (33.33,0,0)$ m/s, $\rho = 1.2$ kg/m^3, $\mu = 1.4 \cdot 10^{-5}$ kg/m, yielding a Reynolds-number of approximately $Re = 10^7$. A Smagorinsky turbulence model was used. The run was initialized with approximately 10^3 timesteps using local timesteps. This was followed by a time-accurate run of 10^4 timesteps, integrating explicitly the advective terms in order to obtain an accurate wake. The results obtained are shown in Fig. 27l and m.

The drag-coefficient obtained was $c_d = 0.309$, based on a reference velocity of $v_\infty = 33.33$ m/s and an area of $A = 2.2$ m^2. Experiments conducted at VW measured a drag-coefficient of $c_d = 0.330$. However, the windtunnel model exhibited an open grille. From past experience, one can infer that performing the experiment with a closed front, as was done for the present run, would reduce the c_d by 5–10% percent in comparison with an open grille. At VW, the estimated value for the closed grille case was $c_d = 0.305$. The purely body-fitted CFD run with the same code yielded a drag-coefficient of $c_d = 0.320$. Overall, this leads to the conclusion that the combined body-fitted/embedded approach leads to results that are within 5% of experimental values, not bad considering the reduction in set-up times, and well within a range to make them of interest for designers. Moreover, as experience with this approach accumulates, one may reasonably expect to be able to obtain even better results.

11. Conclusions and outlook

The use of embedded mesh, immersed body or fictitious domain techniques within adaptive, unstructured grid solvers has matured rapidly over the last decade. As could be seen from the examples shown, as well as the considerable body of literature on this subject, these methods work well for problems characterized by:

- 'dirty geometries',
- moving/sliding bodies with thin/vanishing gaps; and
- physics that can be handled with isotropic grids (potential, Euler, Euler and boundary layers, low-Re Navier–Stokes, LES).

Given that due to the continuing advances in computer hardware the envelope of problems that can be solved with isotropic grids is encompassing ever more flow regimes, and that the human cost of obtaining a body-fitted mesh for complex geometries has stagnated or even increased in recent years, it is foreseeable that these methods will find widespread use. In fact, for metier-specific applications such as external car or missile aerodynamics, a direct link of STL-triangulation output from CAD systems to adaptive, embedded grids is already in place. The user only inputs the desired flow conditions, the mesh is automatically refined to a pre-specified level obtained from 'best practice' experience, and the run proceeds automatically. The combination of body-fitted functionality for some portion of the domain, together with embedded mesh or immersed body functionality for another portion of the domain offers great advantages, which are increasingly being exploited.

Future research topics include:

- faster crossed edge/ immersed body detection, particularly for transient, moving/ deforming body problems;
- faster embedded face/flowfield interpolation procedures, particularly for overlapped embedded (i.e. really dirty) geometries; and
- improved boundary conditions, particularly for implicit time-dependent solvers.

References

[1] M.J. Aftosmis, M.J. Berger, G. Adomavicius, A Parallel Multilevel Method for Adaptively Refined Cartesian Grids with Embedded Boundaries, AIAA-00-0808, 2000.
[2] M.J. Aftosmis, M.J. Berger, J. Alonso, Applications of a Cartesian Mesh Boundary-Layer Approach for Complex Configurations, AIAA-06-0652, 2006.
[3] P. Angot, C.-H. Bruneau, P. Fabrie, A penalization method to take into account obstacles in incompressible viscous flows, Numer. Math. 81 (1999) 497–520.
[4] F.P.T. Baaijens, A fictitious domain/mortar element method for fluid–structure interaction, Int. J. Numer. Methods Fluids 35 (2001) 734–761.
[5] E. Balaras, Modeling complex boundaries using an external force field on fixed cartesian grids in large-eddy simulations, Comput. Fluids 33 (2004) 375–404.
[6] J.D. Baum, H. Luo, E. Mestreau, R. Löhner, D. Pelessone, C. Charman, A Coupled CFD/CSD Methodology for Modeling Weapon Detonation and Fragmentation, AIAA-99-0794, 1999.
[7] J.D. Baum, E. Mestreau, H. Luo, R. Löhner, D. Pelessone, Ch. Charman, Modeling structural response to blast loading using a

coupled CFD/CSD methodology, in: Proc. Des. An. Prot. Struct. Impact/Impulsive/Shock Loads (DAPSIL), Tokyo, Japan, December, 2003.
[8] F. Bertrand, P.A. Tanguy, F. Thibault, Three-dimensional fictitious domain method for incompressible fluid flow problems, Int. J. Numer. Methods Fluids 25 (1997) 136–719.
[9] F.E. Camelli, R.Löhner, VLES Study of Flow and Dispersion Patterns in Heterogeneous Urban Areas, AIAA-06-1419, 2006.
[10] J.R. Cebral, R. Löhner, From medical images to anatomically accurate finite element grids, Int. J. Numer. Methods Engrg. 51 (2001) 985–1008.
[11] J.R. Cebral, R. Löhner, P.L. Choyke, P.J. Yim, Merging of intersecting triangulations for finite element modeling, J. Biomech. 34 (2001) 815–819.
[12] J.R. Cebral, F.E. Camelli, R. Löhner, A feature-preserving volumetric technique to merge surface triangulations, Int. J. Numer. Methods Engrg. 55 (2002) 177–190.
[13] J.R. Cebral, R. Löhner, Efficient simulation of blood flow past complex endovascular devices using an adaptive embedding technique, IEEE Trans. Med. Imag. 24 (4) (2005) 468–476.
[14] Y. Cho, S. Boluriaan, P. Morris, Immersed Boundary Method for Viscous Flow Around Moving Bodies, AIAA-06-1089, 2006.
[15] D.K. Clarke, H.A. Hassan, M.D. Salas, Euler Calculations for Multielement Airfoils Using Cartesian Grids, AIAA-85-0291, 1985.
[16] B.K. Cook, R.P. Jensen (Eds.), Discrete Element Methods, ASCE, 2002.
[17] R. Cortez, M. Minion, The Blop projection method for immersed boundary problems, J. Comp. Physiol. 161 (2000) 428–453.
[18] A. Dadone, B. Grossman, An Immersed Boundary Methodology for Inviscid Flows on Cartesian Grids, AIAA-02-1059, 2002.
[19] J. Deng, X.-M. Shao, A.-L. Ren, A new modification of the immersed-boundary method for simulating flows with complex moving boundaries, Int. J. Numer. Meth. Fluids 52 (2006) 1195–1213.
[20] E.A. Fadlun, R. Verzicco, P. Orlandi, J. Mohd-Yusof, Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations, J. Comp. Physiol. 161 (2000) 35–60.
[21] A. Gilmanov, F. Sotiropoulos, E. Balaras, A general reconstruction algorithm for simulating flows with complex 3D immersed boundaries on Cartesian grids, J. Comp. Physiol. 191 (2) (2003) 660–669.
[22] A. Gilmanov, F. Sotiropoulos, A hybrid Cartesian/immersed boundary method for simulating flows with 3-D, geometrically complex moving objects, J. Comp. Physiol. 207 (2005) 457–492.
[23] R. Glowinski, T.W. Pan, J. Periaux, A fictitious domain method for external incompressible flow modeled by the Navier–Stokes equations, Comput. Meth. Appl. Mech. Engrg. 112 (1–4) (1994) 133–148.
[24] R. Glowinski, T.W. Pan, T.I. Hesla, D.D. Joseph, A distributed lagrange multiplier/fictitious domain method for particulate flows, Int. J. Multiphase Flow 25 (5) (1999) 755–794.
[25] D. Goldstein, R. Handler, L. Sirovich, Modeling a no-slip flow boundary with an external force field, J. Comp. Physiol. 105 (1993) 354366.
[26] S.R. Hanna, M.J. Brown, F.E. Camelli, S.T. Chan, W.J. Coirier, O.R. Hansen, A.H. Huber, S. Kim, R.M. Reynolds, Detailed simulations of atmospheric flow and dispersion in downtown Manhattan: an application of five computational fluid dynamics models, Bull. Am. Meteorol. Soc. (2006) 1713–1726, December 2006.
[27] S.L. Karman, SPLITFLOW: A 3-D Unstructured Cartesian/Prismatic Grid CFD Code for Complex Geometries, AIAA-95-0343, 1995.
[28] J. Kim, D. Kim, H. Choi, An immersed-boundary finite-volume method for simulation of flow in complex geometries, J. Comp. Physiol. 171 (2001) 132–150.
[29] M.C. Lai, C.S. Peskin, An immersed boundary method with formal second-order accuracy and reduced numerical viscosity, J. Comp. Physiol. 160 (2000) 132–150.
[30] A.M. Landsberg, J.P. Boris, The Virtual Cell Embedding Method: A Simple Approach for Gridding Complex Geometries, AIAA-97-1982, 1997.
[31] R.J. LeVeque, Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, SIAM J. Numer. Anal. 31 (1994) 1019–1044.
[32] R.J. LeVeque, D. Calhoun, Cartesian grid methods for fluid flow in complex geometries, in: L.J. Fauci, S. Gueron (Eds.), Computational Modeling in Biological Fluid Dynamics, IMA Volumes in Mathematics and its Applications 124, Springer-Verlag, 2001, pp. 117–143.
[33] R. Löhner, An adaptive finite element scheme for transient problems in CFD, Comput. Meth. Appl. Mech. Engrg. 61 (1987) 323–338.
[34] R. Löhner, J.D. Baum, Adaptive *h*-refinement on 3-D unstructured grids for transient problems, Int. J. Numer. Meth. Fluids 14 (1992) 1407–1419.
[35] R. Löhner, Computational Aspects of Space-Marching, AIAA-98-0617, 1998.
[36] R. Löhner, C. Yang, J.D. Baum, H. Luo, D. Pelessone, C. Charman, The numerical simulation of strongly unsteady flows with hundreds of moving bodies, Int. J. Numer. Meth. Fluids 31 (1999) 113–120.
[37] R. Löhner, C. Yang, J. Cebral, J.D. Baum, H. Luo, E. Mestreau, D. Pelessone, C. Charman, Fluid–structure interaction algorithms for rupture and topology change, in: Proc. 1999 JSME Computational Mechanics Division Meeting, Matsuyama, Japan, November, 1999.
[38] R. Löhner, Applied CFD Techniques, John Wiley & Sons, 2001.
[39] R. Löhner, E. Oñate, A general advancing front technique for filling space with arbitrary objects, Int. J. Numer. Meth. Engrg. 61 (2004) 1977–1991.
[40] R. Löhner, J.D. Baum, E. Mestreau, D. Sharov, C. Charman, D. Pelessone, Adaptive embedded unstructured grid methods, Int. J. Numer. Meth. Engrg. 60 (2004) 641–660.
[41] R. Löhner, J.D. Baum, E.L. Mestreau, Advances in Adaptive Embedded Unstructured Grid Methods, AIAA-04-0083, 2004.
[42] R. Löhner, H. Luo, J.D. Baum, D. Rice, Selective Edge Deactivation for Unstructured Grids with Cartesian Cores, AIAA-05-5232, 2005.
[43] R. Löhner, J.D. Baum, E.L. Eric L. Mestreau, D. Rice, Comparison of Body-Fitted, Embedded and Immersed 3-D Euler Predictions for Blast Loads on Columns, AIAA-07-1133, 2007.
[44] R. Löhner, S. Appanaboyina, J. Cebral, Comparison of Body-Fitted, Embedded and Immersed Solutions for Low Reynolds-Number Flows, AIAA-07-1296, 2007.
[45] J.E. Melton, M.J. Berger, M.J. Aftosmis, 3-D Applications of a Cartesian Grid Euler Method, AIAA-93-0853-CP, 1993.
[46] R. Mittal, G. Iaccarino, Immersed boundary methods, Annu. Rev. Fluid Mech. 37 (2005) 239–261.
[47] J. Mohd-Yusof, Combined Immersed-Boundary/B-Spline Methods for Simulations of Flow in Complex Geometries, CTR Annual Research Briefs, NASA Ames Research Center/Stanford Univ., 1997, pp. 317–327.
[48] H. Morino, K. Nakahashi, Space-Marching Method on Unstructured Hybrid Grid for Supersonic Viscous Flows, AIAA-99-0661, 1999.
[49] K. Nakahashi, E. Saitoh, Space-Marching Method on Unstructured Grid for Supersonic Flows with Embedded Subsonic Regions, AIAA-96-0418, 1996, see also AIAA J. 35(8) (1997) 1280–1285.
[50] N.A. Patankar, P. Singh, D.D. Joseph, R. Glowinski, T.W. Pan, A new formulation of the distributed Lagrange multiplier/fictitious domain method for particulate flows, Int. J. Multiphase Flow (April) (1999).
[51] N. Peller, A. LeDuc, F. Tremblay, M. Manhart, High-order stable interpolations for immersed boundary methods, Int. J. Numer. Meth. Fluids 52 (2006) 1175–1193.
[52] R.B. Pember, J.B. Bell, P. Colella, W.Y. Crutchfield, M.L. Welcome, An adaptive Cartesian grid method for unsteady compressible flow in irregular regions, J. Comp. Physiol. 120 (1995) 278.
[53] C.S. Peskin, The immersed boundary method, Acta Numer. 11 (2002) 479–517.
[54] S. Del Pino, O. Pironneau, Fictitious Domain Methods and Freefem3d, in: Proc. ECCOMAS CFD Conf., Swansea, Wales, 2001.
[55] J.J. Quirk, A Cartesian Grid Approach with Hierarchical Refinement for Compressible Flows, NASA CR-194938, ICASE Report No. 94-51, 1994.

[56] A.M. Roma, C.S. Peskin, M.J. Berger, An adaptive version of the immersed boundary method, J. Comp. Physiol. 153 (1995) 509–534.
[57] H. Schlichting, Boundary Layer Theory, McGraw-Hill, 1979.
[58] G. Sod, A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws, J. Comp. Physiol. 27 (1978) 1–31.
[59] R. Tilch, A. Tabbal, M. Zhu, F. Dekker, R. Löhner, Combination of Body-Fitted and Embedded Grids for External Vehicle Aerodynamics, AIAA-07-1300, 2007.
[60] M. Tyagi, S. Acharya, Large eddy simulation of turbulent flows in complex and moving rigid geometries using the immersed boundary method, Int. J. Numer. Methods Fluids 48 (2005) 691–722.
[61] K. Tsuboi, K. Miyakoshi, K. Kuwahara, Incompressible flow simulation of complicated boundary problems with rectangular grid system, Theor. Appl. Mech. 40 (1991) 297–309.
[62] S. Turek, Efficient solvers for incompressible flow problems, Springer Lecture Notes in Computational Science and Engineering, vol. 6, Springer, 1999.
[63] J. vande Voorde, J. Vierendeels, E. Dick, Flow simulations in rotary volumetric pumps and compressors with the fictitious domain method, J. Comput. Appl. Math. 168 (2004) 491–499.
[64] J. Yang, E. Balaras, An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries, J. Comp. Physiol. 215 (2006) 12–40.
[65] T. Ye, R. Mittal, H.S. Udaykumar, W. Shyy, An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries, J. Comp. Physiol. 156 (1999) 209–240.
[66] D. de Zeeuw, K. Powell, An Adaptively-Refined Cartesian Mesh Solver for the Euler Equations, AIAA-91-1542, 1991.