INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2008; **57**:13–30
Published online 20 September 2007 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1604

Comparison of body-fitted, embedded and immersed solutions of low Reynolds-number 3-D incompressible flows

Rainald Löhner*,†, Sunil Appanaboyina and Juan R. Cebral

Center for Computational Fluid Dynamics, Department of Computational and Data Sciences, College of Sciences, M.S. 6A2, George Mason University, Fairfax, VA, U.S.A.

SUMMARY

The solutions obtained for low Reynolds-number incompressible flows using the same flow solver and solution technique on body-fitted, embedded surface and immersed body grids of similar size are compared. The cases considered are a sphere at $Re = 100$ and an idealized stented aneurysm. It is found that the solutions using all these techniques converge to the same grid-independent solution. On coarser grids, the effect of higher-order boundary conditions is noticeable. Therefore, if the manual labor required to set up a body-fitted domain is excessive (as is often the case for patient-specific geometries with medical devices), and/or computing resources are plentiful, the embedded surface and immersed body approaches become very attractive options. Copyright © 2007 John Wiley & Sons, Ltd.

Received 13 February 2007; Revised 4 July 2007; Accepted 6 July 2007

KEY WORDS: embedded surface technique; immersed body method; incompressible flow; FEM; CFD

1. INTRODUCTION

With the advent of robust, accurate flow solvers and automatic grid generators, the task of defining quickly a flow domain and the required boundary conditions has become a considerable bottleneck for many numerical simulations. Presently, the interactive design of medical devices governed by hemodynamic (bloodflow) phenomena, which requires the solution of the incompressible Navier–Stokes equations in complex 3-D domains, is hindered by this key issue. For the 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 if the definition of the structure emanates from a CAD package. However, in most patient-specific cases the surface data stems from medical images, onto which medical devices such as stents or coils must be superposed. Generating body-fitted grids for such

*Correspondence to: Rainald Löhner, Center for Computational Fluid Dynamics, Department of Computational and Data Sciences, College of Sciences, M.S. 6A2, George Mason University, Fairfax, VA, U.S.A.
†E-mail: rlohner@gmu.edu

configurations has been attempted in the past [1–3], but remains labor intensive and error prone. An alternative is to use grids that are not body conforming, and simply 'embed' the triangulations of the wetted surfaces of the medical devices in them. Techniques of this kind are also known as immersed [3–12], embedded [13, 14], fictitious domain [15] or Cartesian methods [16–24]. The treatment of points and elements in the vicinity of the embedded triangulations or bodies is modified in such a way that the required kinetic or kinematic boundary conditions are properly imposed. An embedded technique of this type, combined with adaptive unstructured grids, has been pursued by the authors for a number of years for medical device modeling [14, 25]. The essential elements of this technique may be summarized as follows:

- The key modification of the original, body-fitted edge-based solver was the removal of all geometry parameters (essentially the area normals) belonging to edges cut by embedded surface faces.
- Higher-order boundary conditions are achieved by duplicating crossed edges and their end-points, or by extrapolating values and gradients from inside the domains to the surface.
- Geometric resolution and solution accuracy may be enhanced by adaptive mesh refinement that is based on the proximity to or the curvature of the embedded surfaces or objects.
- In order to save work, user-defined or automatic deactivation for regions inside immersed solid bodies is employed.

Naturally, the question arises as to how accurate these embedded techniques are, and whether they could be used for those cases where manual labor is expected to be prohibitively high.

To this end, low Reynolds-number incompressible flow past a sphere—a well-documented testcase—and an idealized stented aneurysm were analyzed. The cases were run with the same solver and code, exercising the body-fitted, embedded-mesh and immersed-body options. Pressure and velocity data were compared.

The remainder of this paper is organized as follows. Section 2 describes the overall methodology and algorithms used for incompressible flow calculations. Sections 3 and 4 detail the embedded-mesh and immersed-body techniques used. Section 5 presents the comparison of several relevant runs, and discusses the implications. Some conclusions and an outlook are given in Section 6.

2. METHODOLOGY

Any CFD run proceeds through the following stages:

- pre-processing;
- grid generation;
- flow solver;
- post-processing.

In the pre-processing phase, the data and boundary conditions are acquired and defined, the desired mesh size is specified and all run-time files are prepared. For the applications shown here, these tasks were carried out with the special pre-processor ZHEMO [25] that is tailored to patient-specific hemodynamic problems, and the more general-purpose computational fluid dynamics (CFD) pre-processor FECAD [26]. ZHEMO is employed to acquire, process and segment the medical images, and to define the overall computational domain. FECAD is used to define the boundary conditions for the CFD solver, and to define the spatial variation of mesh size. For embedded grids or

DOI: 10.1002/fld

immersed bodies, the (discrete) surface data are acquired, and the mesh in the region of the embedded or immersed object is automatically specified to be small enough. An advantage of the present methodology is that the embedded, immersed and body-fitted approaches may be combined in a single run. Thus, the arterial walls (and domains) are treated using the body-fitted methodology, while medical devices such as stents and coils are treated using the embedded or immersed option. The computational domain is then filled with tetrahedral elements of specified size by using FEGEN, which incorporates the advancing front technique [27, 28]. For the flow solver, a fractional step, projection-type finite element solver [26, 29–31] is used. Post-processing is carried out using FEPOST or ZFEM [32, 33] as well as several x/y plotting tools such as gnuplot. All the pre- and post-processing, as well as any run requiring less than 4 Mtet elements are run on the PC, allowing for full integration under a single graphical user interface (GUI).

3. EMBEDDED MESH TECHNIQUES

In what follows, we denote the surface of the object that is embedded by computational structural dynamics (CSD) faces. We implicitly assume that this information is given by a triangulation, which is typically obtained from a CAD package *via* STL files, but may also originate from remote sensing data, medical images or from a CSD code—hence the name—in coupled fluid–structure applications. For immersed methods, we assume that the embedded object is given by a tetrahedral mesh.

Embedded grids are treated by imposing appropriate *kinematic boundary conditions* for the fluid nodes close to the embedded surfaces. 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. Note that in both cases the treatment of infinitely thin surfaces with fluid on both sides (e.g. membranes) is straightforward.

A first-order scheme may 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.

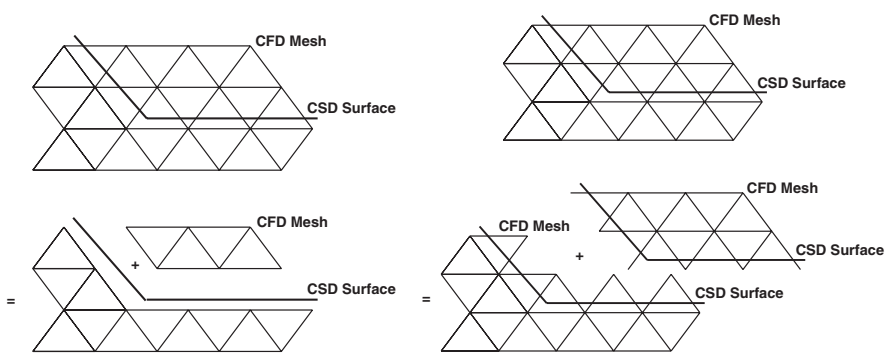

Figure 1. First/second-order treatment of embedded surfaces.

DOI: 10.1002/fld

A second-order scheme may be obtained 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.

3.1. 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. Without loss of generality, let us assume an octree for the CSD faces. Then, a (parallel) loop is performed over the edges. For each edge, the bounding box of the edge is computed (i.e. the min/max extent of the edge in each coordinate direction). From the octree, all the faces in the region of the bounding box are found. This is followed by an in-depth test to determine which faces cross the given edge. The crossing face closest to each of the edge end nodes is stored. This allows us 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 can 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 covering that spatial region. One typically finds that the list of edges to be tested in detail has been reduced by an order of magnitude.

3.2. 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

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

Here C, F, M denote, respectively, the edge coefficients, fluxes and mass matrix. For any edge ij crossed by a CSD face, the coefficients C^{ij} are set to zero. This implies that for a uniform state $\mathbf{u} = \text{const.}$ 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

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

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

DOI: 10.1002/fld

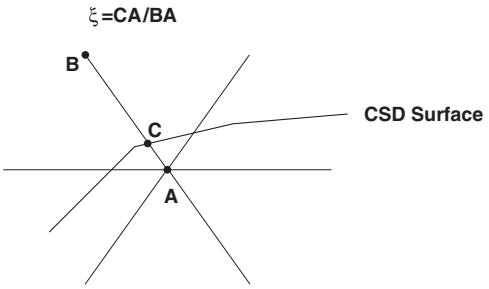

Figure 2. Cut edge fraction.

Equation (1) as

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

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

The mass matrix M^i of points surrounded by cut edges must be modified to reflect the reduced volume due to cut elements. The simplest possible modification of M^i is given by the so-called 'cut edge fraction' method [14].

In a pass over the edges, the smallest 'cut edge fraction' ξ for all the edges surrounding a point is found (see Figure 2). The modified mass matrix is then given by

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

As stated before, this is the simplest possible modification of M^i. It is exact for flat-embedded surfaces, i.e. also in the limit of very small element size. It is certainly incorrect for corners or cusps. Despite this drawback, previous experience [14] indicates that it works well even for the complex, dirty geometries for which embedded techniques were originally developed. Besides simplicity, an important advantage of the 'cut edge fraction' approach is that the value of the modified mass matrix can never fall below half its original value, implying that timestep sizes will always be acceptable. Furthermore, for a thin-embedded surface both point A as well as point B in Figure 2 could be inside the flow domain.

3.2.1. Boundary conditions. The points belonging to cut edges are now part of a boundary, and thefore require the imposition of the proper partial differential equation (PDE) boundary conditions. For the low *Re*-number cases considered here, these are given by an imposed 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 Figure 3, for each boundary

DOI: 10.1002/fld

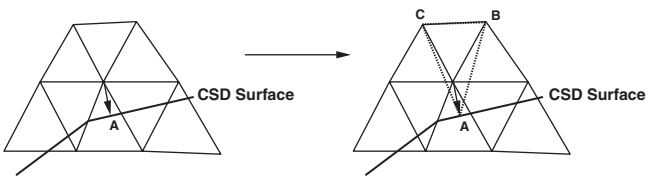

Figure 3. Extrapolation of velocity.

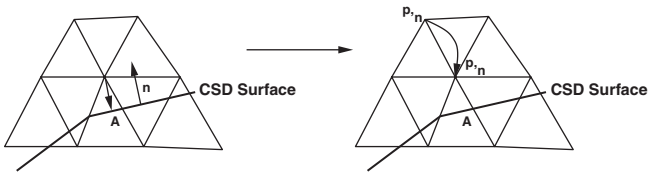

Figure 4. Extrapolation of normal pressure gradient.

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 timestep.

The normal gradients at the boundary points can be improved by considering the 'most aligned' field (i.e. non-boundary) point to the line formed by the boundary point and the closest point on the CSD face (see Figure 4).

3.3. 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 memory due to the 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.

3.3.1. Boundary conditions. By duplicating the edges, the points are treated in the same way as in the original (non-embedded) case. The boundary conditions are imposed indirectly by mirroring and interpolating the unknowns as required. Figure 5 depicts the contribution due to the edges surrounding point i. A CSD boundary crosses the edges of 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' (i.e. the ghost point is j') based on the crossed CSD face (distance, velocity).

The element used for the interpolation might either be crossed (Figure 6(a)) or not exist (Figure 6(b)). In this case, the information interpolated is based solely on points that are truly inside the fluid domain, discarding the information of points that require interpolated information. This is done by first finding the shape functions $\eta_k, k = 1, 4$ of the host element that contains the ghost point j' in the usual way. For all the points that may not be used for interpolation, the shape

DOI: 10.1002/fld

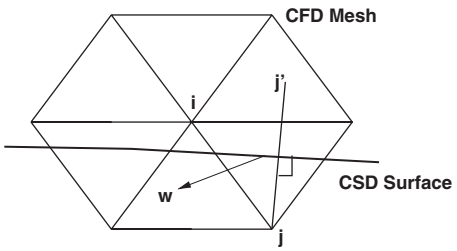

Figure 5. Higher-order boundary conditions (**w** denotes the wall velocity).

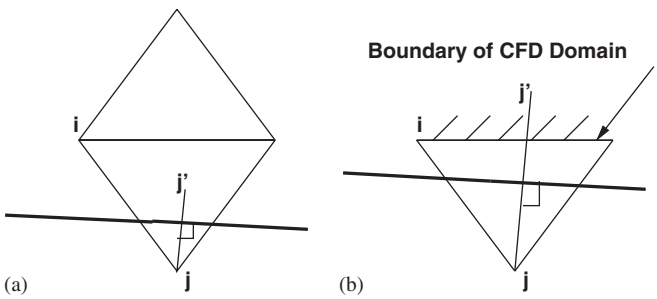

Figure 6. Problem cases.

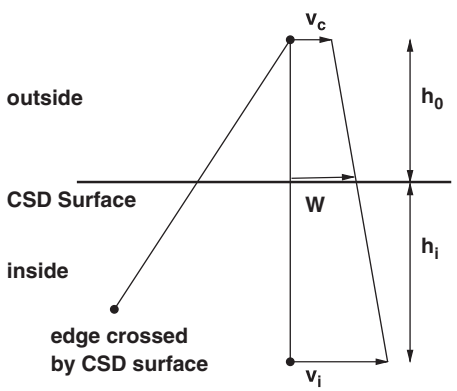

Figure 7. Navier–Stokes boundary condition.

functions are simply set to zero. The shape-function values are then renormalized so that their sum still adds up to unity. This has the effect of moving the ghost points towards the closest face, edge, or the point that contains interpolable information (i.e. the points that are not ghost points).

Geometrically, this implies 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 Figure 7, and assuming a linear interpolation of the velocities, the values are interpolated as

$$\mathbf{w} = (1 - \xi_w)\mathbf{v}_c + \xi_w \mathbf{v}_i, \quad \xi_w = \frac{h_o}{h_o + h_i} \tag{5}$$

DOI: 10.1002/fld

i.e.

$$\mathbf{v}_c = \frac{1}{1-\xi_w}\mathbf{w} - \frac{\xi_w}{1-\xi_w}\mathbf{v}_i \tag{6}$$

Here $\mathbf{w}$ is the average velocity of the crossed CSD face, $\mathbf{v}_i$ the interpolated flow velocity, h_o may not be the same as h_i due to the movement of the ghost point (but $h_i \geqslant h_o$), and the distance factor $\xi_w \leqslant 0.5$.

4. IMMERSED BODY TECHNIQUE

The presence of immersed bodies is imposed in the flowfield by adding suitable force functions. If we consider a rigid, closed body, as sketched in Figure 8, an obvious aim is to enforce, within the body, that the fluid velocity is the same as the body velocity, i.e. $\mathbf{v} = \mathbf{v}_b$. This may be accomplished by applying a force term of the form

$$\mathbf{f} = -c_0(\mathbf{v}_b - \mathbf{v}) \tag{7}$$

for the points that are inside of the body. This particular type of force function is known as the penalty force technique [4–8].

The penalty force technique used here follows the approach of Mohd-Yusof [5]. The usual right-hand side for the flow equations at immersed points (or cells) is evaluated first. Then, a force is added such that the velocity at the next timestep satisfies the kinematic boundary conditions. The spatially discretized form of the momentum equations at each point (or cell) i is written as

$$\mathbf{M}\frac{\Delta \mathbf{v}_i}{\Delta t} = \mathbf{r}_i + \mathbf{f}_i \tag{8}$$

$\mathbf{f}^i$ is obtained as

$$\mathbf{f}_i = \mathbf{M}\frac{\mathbf{w}_i^{n+1} - \mathbf{v}_i^n}{\Delta t} - \mathbf{r}_i \tag{9}$$

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 time-stepping schemes, this force function in effect imposes the

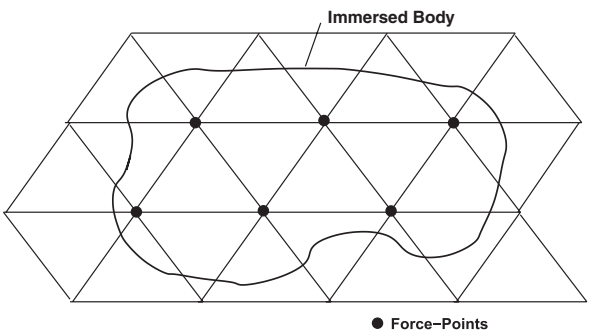

Figure 8. Kinetic treatment of embedded surfaces.

DOI: 10.1002/fld

(required) velocity of the immersed body at the new timestep. We therefore denote this treatment of the immersed bodies as 'kinetic–kinematic'. Schemes of this kind have been used repeatedly in conjunction with fractional step/projection methods for incompressible flow [4–8, 10, 11], and, recently, also for compressible flows [34].

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 across the embedded surface/immersed body. This implies that for thin-embedded surfaces such as membranes, where the pressure is different on both sides, this method will not yield satisfactory results.

4.1. 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 that fall into a CSD tetrahedron (and possibly 1–2 layers of neighbours).

The first of these is basically the same as that required for embedded CSD methods, and has been dealt with extensively above. The second task may 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;
- 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 of the immersed body host element for each CFD mesh point covered by immersed body elements.

(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 of the immersed body host element for each CFD mesh point covered by immersed body elements.

In both the 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. Also, note that both approaches are easily parallelized on shared memory machines.

5. TREATMENT OF PARTICLES

A promising way to treat complex filaments such as coils is to consider them as a collection of spheres [25]. With the aid of automatic tools to fill space with objects of arbitrary shape [35] the

DOI: 10.1002/fld

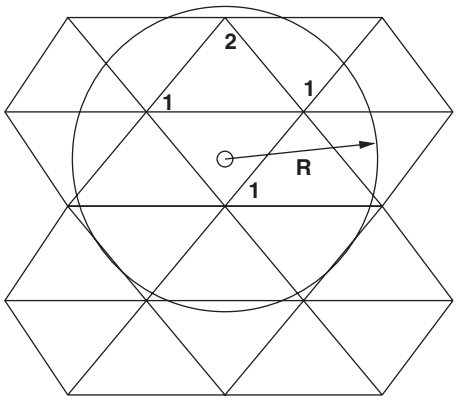

Figure 9. Link to discrete particle method.

generation of coils no longer represents a burden on the analyst. Adaptive embedded or immersed grid techniques can be linked to a particle representation of objects in a very natural way. The host element for each one of the discrete particles is updated every timestep and is assumed as given. All points of elements covered by particles are marked for additional boundary conditions. The closest particle to each of these points is used to impose the desired velocities (in the general case particles are assumed to be moving at arbitrary, different velocities). The points of the element into which the center of the particle falls (central dot in Figure 9) are marked (1's in Figure 9). Thereafter, additional points are added in layers (2's in Figure 9), until all points covered by particles are marked.

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 technique. The velocity of particles is imposed at the endpoints of crossed edges. One may distinguish an 'aggressive' (take also the points exterior to particles as boundary points) and 'conservative' (take only the points interior to particles as boundary points) option. Particles may also be used in conjunction with the immersed body technique by imposing on all points interior to particles the velocity of the particles.

6. RESULTS

6.1. Flow past a sphere

The first case considered is shown in Figure 10. 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/immersed options are shown in Figure 10(a). We implicitly assumed that the body-fitted results were more accurate and therefore considered them as the 'gold standard'. Figure 10(b) shows the same 50 surface

DOI: 10.1002/fld

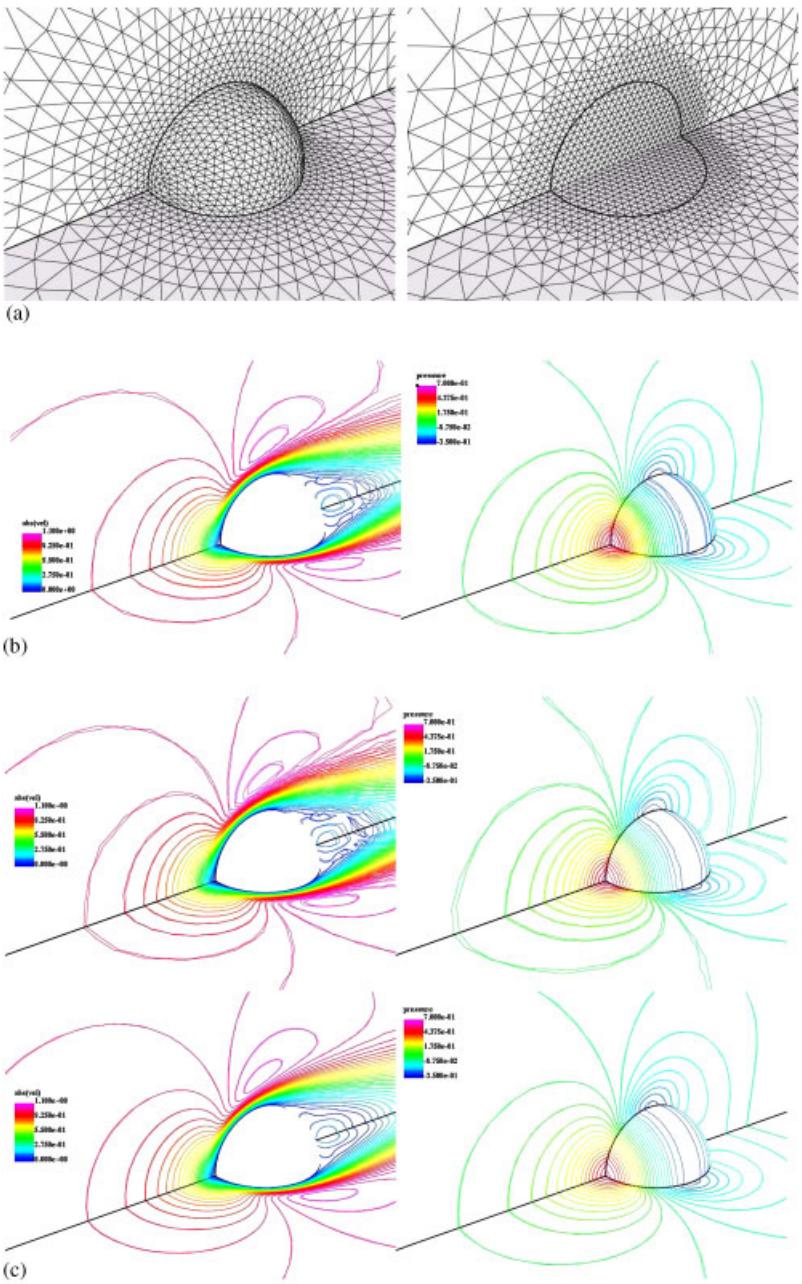

Figure 10. (a) Sphere: surface grids (body fitted, embedded/immersed, coarse mesh); (b) coarse *vs* fine body fitted (left: $|\mathbf{v}|$, right: p); (c) body fitted *vs* embedded 1 (top: coarse, bottom fine); (d) body fitted *vs* embedded 2 (top: coarse, bottom fine); (e) body fitted *vs* immersed (top: coarse, bottom fine); (f) body fitted *vs* DPM A (top: coarse, bottom fine); (g) body fitted *vs* DPM C (top: coarse, bottom fine); (h) body fitted *vs* DPM immersed (top: coarse, bottom fine); and (i) velocity and pressure along line: $y, z = 0.0$ (top: coarse, bottom fine).

DOI: 10.1002/fld

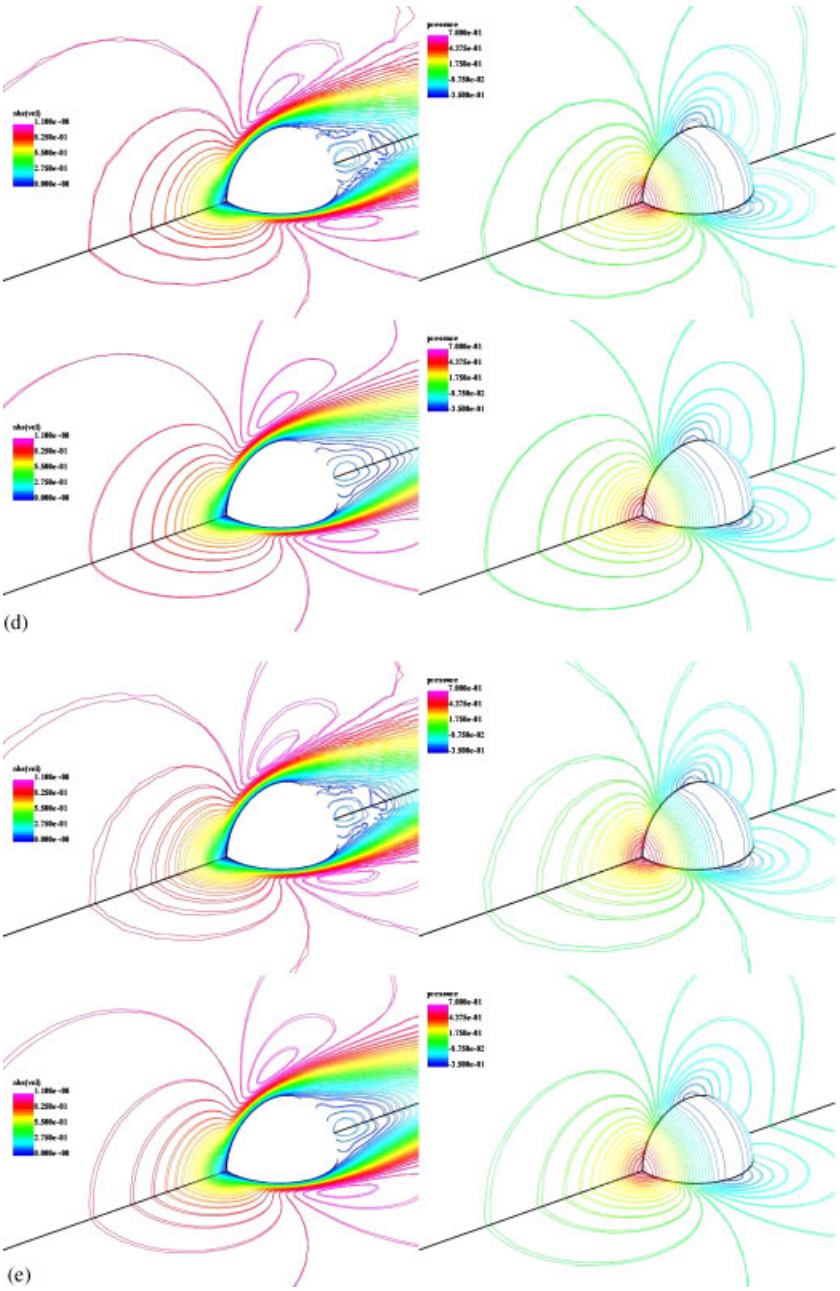

Figure 10. *Continued.*

DOI: 10.1002/fld

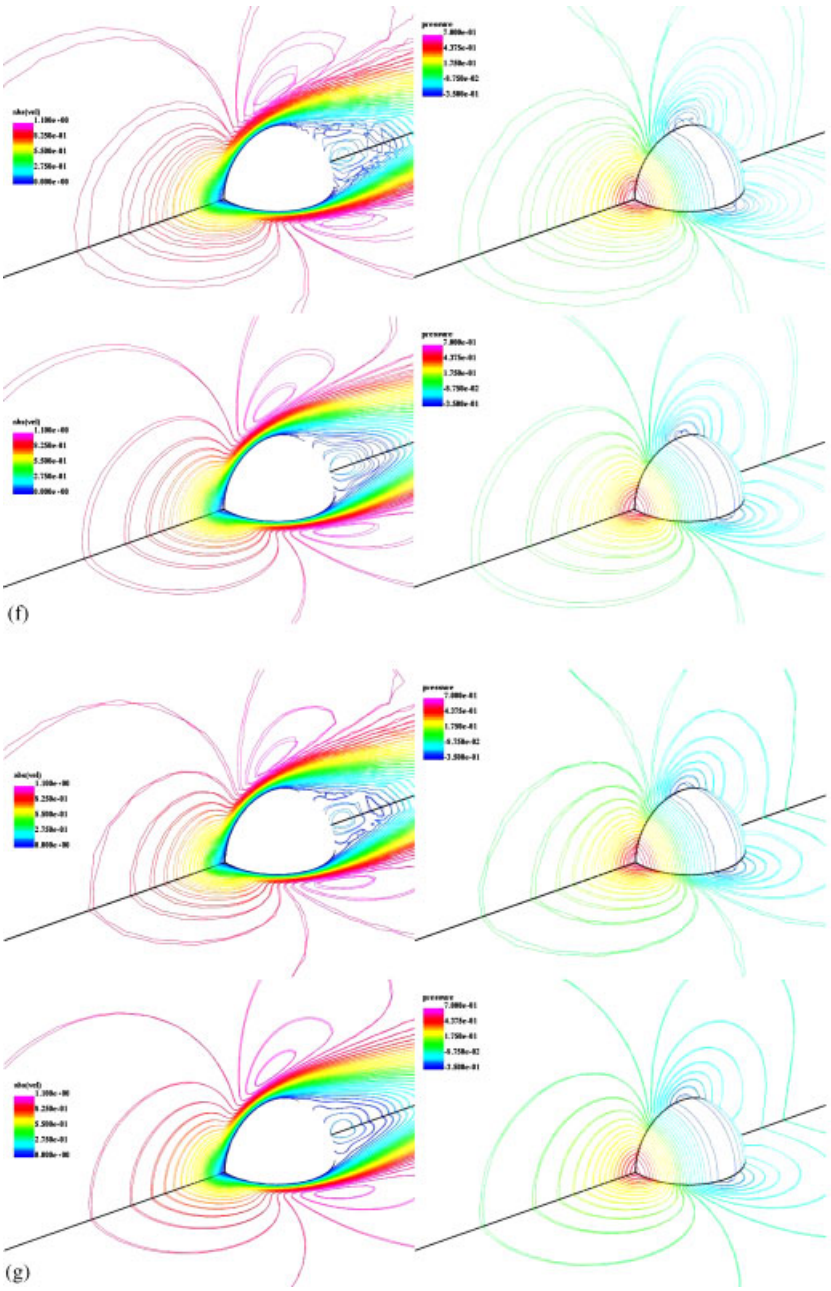

Figure 10. *Continued.*

DOI: 10.1002/fld

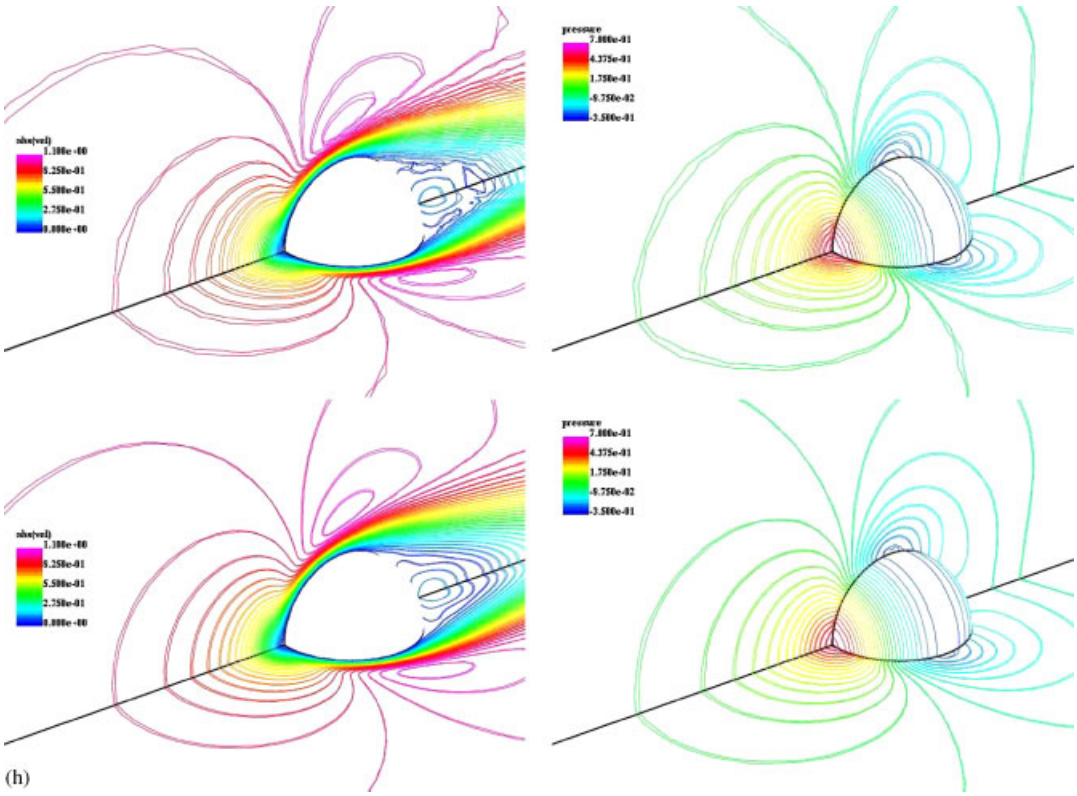

Figure 10. *Continued.*

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. Given that 50 contour lines were used, and that the contour lines are nested, a difference in location between any two contour lines corresponds to less than 2% difference between the solutions. In fact, the closeness of the contour lines indicates that the difference between the solutions (and therefore the accuracy of the coarse mesh solution) is less than 0.2% in absolute terms. Given all other uncertainties present in patient-specific calculations, pointwise accuracy to even 1% is deemed more than sufficient. The drag coefficients for the two body-fitted grids were given by $c_D = 1.07$ and 1.08, respectively, in good agreement with experimental results [36]. Figures 10(c)–(h) 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. If one had to judge the accuracy of the different options, the order (starting from the solutions closest to the body-fitted ones) would be as follows: embedded 2, embedded 1, DPM-immersed, DPM-C, immersed, DPM-A. Remarkably, on the fine mesh the contour lines for the first two options are almost indistinguishable from the those of the body-fitted solution. Figure 10(i) depicts 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,

DOI: 10.1002/fld

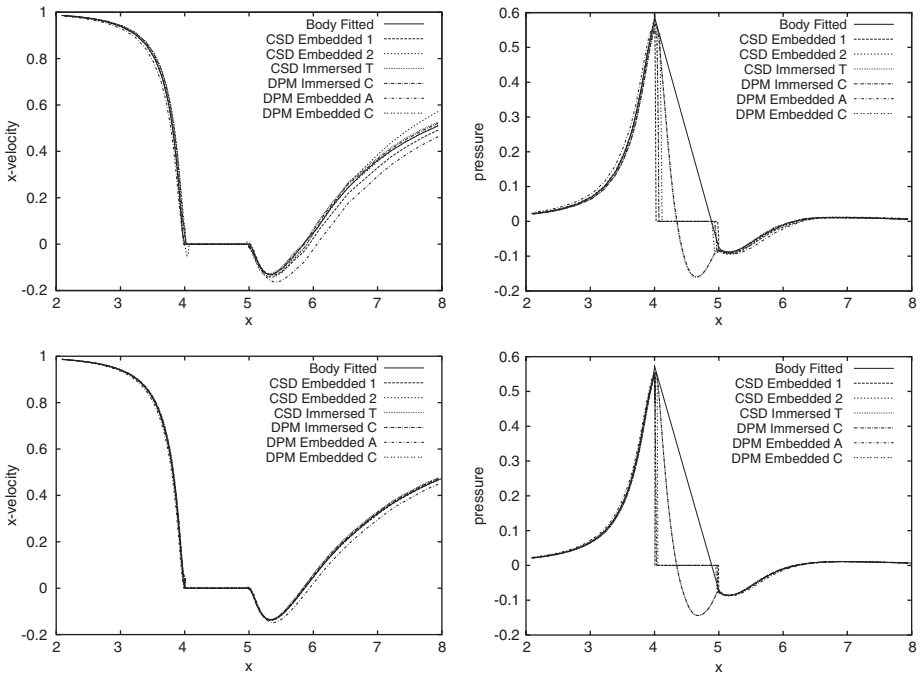

Figure 10. *Continued.*

indicating that even the first-order embedded scheme has converged. The recirculation length is in good agreement with other numerical results [8, 37]. The CPU requirements of all methods were very similar, differing by less that 25%. For the embedded cases the points inside the sphere are deactivated, so the expected (and observed) CPU requirements are very close to that of the body-fitted mesh.

6.2. Aneurysm with stent

The second case considered is typical of medical device predictions. An idealized aneurysm in an artery is surrounded by a stent. The intersection of the stent with the arterial walls represents a formidable challenge to CAD-based body-fitted grid generation due to the presence of the different length scales, as well as the sharp angles at the intersections. An alternative approach is to define the arterial domain using body-fitted grids, and to embed or immerse the stent. The stent is represented as a collection of spheres. The overall domain is shown in Figure 11(a). The surface of the adapted body-fitted mesh, which had approximately 10 Mels, is shown in Figures 11(a) and (b). The pressures and velocities obtained for the embedded and immersed approaches are shown in Figures 11(c)–(e). Note that the results are almost identical. This is to be expected, as the Reynolds number based on the sphere diameter is below $Re = 25$. From a design perspective, the effect of the stent is striking, whereas without the stent a large recirculation vortex is present in the aneurysm, the insertion of the stent achieves the desired objective of very low velocities in the aneurysm, and reverts the flow direction.

DOI: 10.1002/fld

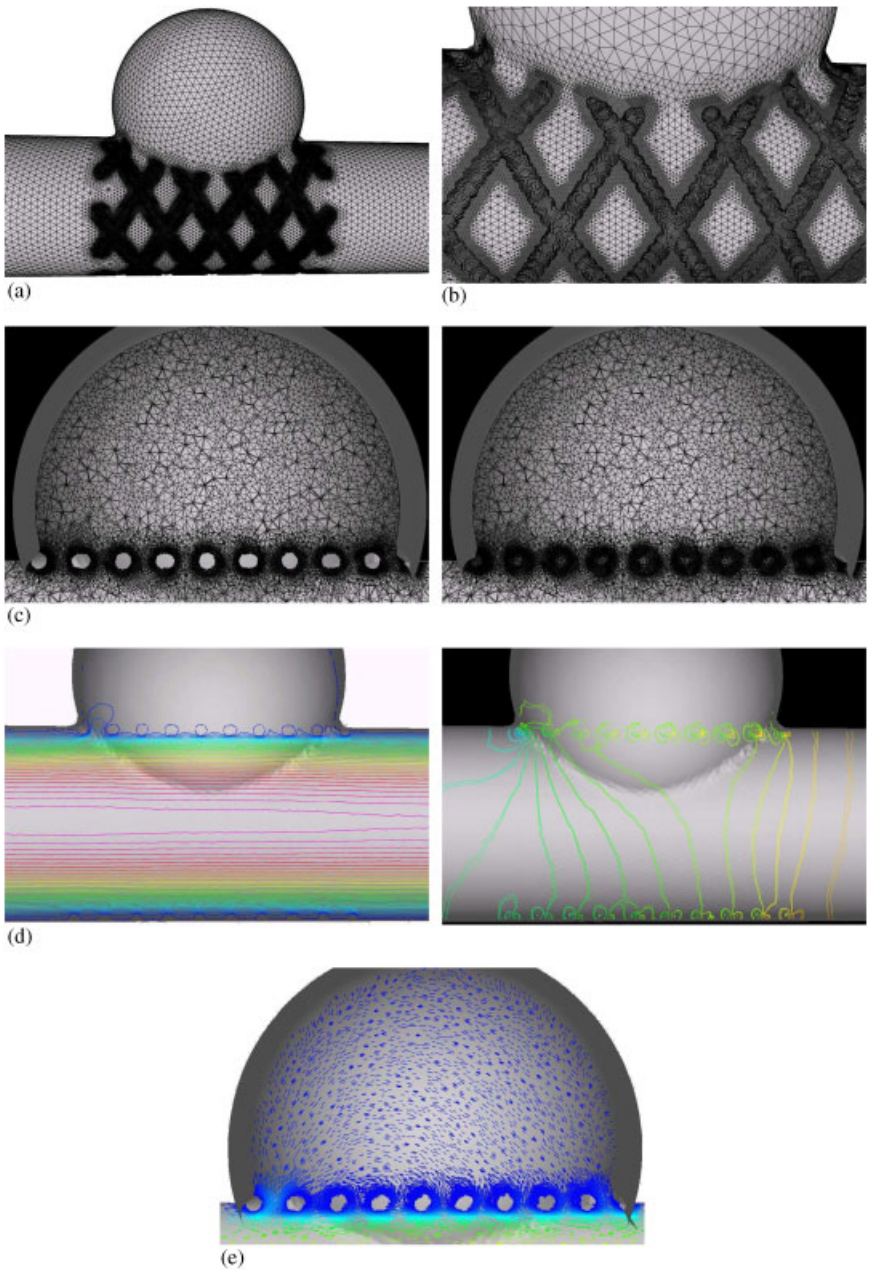

Figure 11. (a, b) Surface grid; (c) cut-plane $z = 0.0$ grids (left: embedded, right: immersed); (d) contours of $|\mathbf{v}|$ and p in the mid-plane: $z = 0.0$ (embedded and immersed); and (e) velocities in the mid-plane: $z = 0.0$ (embedded and immersed).

DOI: 10.1002/fld

7. CONCLUSIONS AND OUTLOOK

Solutions obtained for low Reynolds-number incompressible flows using the same flow solver and solution technique on body-fitted, embedded surface and immersed body grids of similar size have been compared. The cases considered are a sphere at $Re = 100$ and a stented aneurysm. It is found that the solutions using all these techniques converge to the same grid-independent solution. On coarser grids, the effect of higher-order boundary conditions is noticeable. Therefore, if the manual labor required to set up a body-fitted domain is excessive (as is often the case for patient-specific geometries with medical devices), and/or computing resources are plentiful, the embedded surface and immersed body approaches become very attractive options.

Given that computers will continue to increase in performance and available memory, the prospects for the embedded mesh and immersed body methods are very bright. Development of improved boundary condition treatment remains an active area of research, and should render these methods even more competitive.

REFERENCES

1. Stuhne GR, Steinman DA. Finite element modeling of the hemodynamics of stented aneurysms. *Journal of Biomechanical Engineering* 2004; **126**(3):382–387.
2. de Putter S, Laffargue F, Breeuwer M, van de Vosse FN, Gerritsen FA. Computational mesh generation for vascular structures. *International Journal of CARS* 2006; **1**:1–11.
3. van Loon R, Anderson PD, van de Vosse FN. A fluid–structure interaction method with solid-rigid contact for heart valve dynamics. *Journal of Computational Physics* 2006; **217**(2):806–823.
4. Goldstein D, Handler R, Sirovich L. Modeling a no-slip flow boundary with an external force field. *Journal of Computational Physics* 1993; **105**:354–366.
5. Mohd-Yusof J. Combined immersed-boundary/B-spline methods for simulations of flow in complex geometries. *CTR Annual Research Briefs*, NASA Ames Research Center/Stanford University, 1997; 317–327.
6. Angot P, Bruneau C-H, Fabrie P. A penalization method to take into account obstacles in incompressible viscous flows. *Numerische Mathematik* 1999; **81**:497–520.
7. Fadlun EA, Verzicco R, Orlando P, Moud-Yusof J. Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *Journal of Computational Physics* 2000; **161**:33–60.
8. Kim J, Kim D, Choi H. An immersed-boundary finite-volume method for simulation of flow in complex geometries. *Journal of Computational Physics* 2001; **171**:132–150.
9. Dadone A, Grossman B. An immersed boundary methodology for inviscid flows on Cartesian grids. *AIAA-02-1059*, 2002.
10. Peskin CS. The immersed boundary method. *Acta Numerica* 2002; **11**:479–517.
11. Gilmanov A, Sotiropoulos F. A hybrid Cartesian/immersed boundary method for simulating flows with 3-D, geometrically complex moving bodies. *Journal of Computational Physics* 2005; **207**(2):457–492.
12. Mittal R, Iaccarino G. Immersed boundary methods. *Annual Review of Fluid Mechanics* 2005; **37**:239–261.
13. Landsberg AM, Boris JP. The virtual cell embedding method: a simple approach for gridding complex geometries. *AIAA-97-1982*, 1997.
14. Löhner R, Baum JD, Mestreau E, Sharov D, Charman C, Pelessone D. Adaptive embedded unstructured grid methods. *International Journal for Numerical Methods in Engineering* 2004; **60**:641–660.
15. Del Pino S, Pironneau O. Fictitious domain methods and freefem3D. *Proceedings of the ECCOMAS CFD Conference*, Swansea, Wales, 2001.
16. Clarke DK, Hassan HA, Salas MD. Euler calculations for multielement airfoils using Cartesian grids. *AIAA-85-0291*, 1985.
17. de Zeeuw D, Powell K. An adaptively-refined Cartesian mesh solver for the Euler equations. *AIAA-91-1542*, 1991.
18. Melton JE, Berger MJ, Aftosmis MJ. 3-D applications of a Cartesian grid Euler method. *AIAA-93-0853-CP*, 1993.

DOI: 10.1002/fld

19. Quirk JJ. A Cartesian grid approach with hierarchical refinement for compressible flows. *NASA CR-194938*, *ICASE Report No. 94-51*, 1994.
20. Karman SL. SPLITFLOW: a 3-D unstructured Cartesian/prismatic grid CFD code for complex geometries. *AIAA-95-0343*, 1995.
21. Pember RB, Bell JB, Colella P, Crutchfield WY, Welcome ML. An adaptive Cartesian grid method for unsteady compressible flow in irregular regions. *Journal of Computational Physics* 1995; **120**:278.
22. LeVeque RJ, Calhoun D. Cartesian grid methods for fluid flow in complex geometries. In *Computational Modeling in Biological Fluid Dynamics*, Fauci LJ, Gueron S (eds). IMA Mathematics and its Applications, vol. 124. Springer: Berlin, 2001; 117–143.
23. Aftosmis MJ, Berger MJ, Adomavicius G. A parallel multilevel method for adaptively refined Cartesian grids with embedded boundaries. *AIAA-00-0808*, 2000.
24. Murman SM, Aftosmis MJ, Berger MJ. Implicit approaches for moving boundaries in a 3-D Cartesian method. *AIAA-03-1119*, 2003.
25. Cebral JR, Löhner R. Efficient simulation of blood flow past complex endovascular devices using an adaptive embedding technique. *IEEE Transactions on Medical Imaging* 2005; **24**(4):468–476.
26. Löhner R, Yang C, Cebral J, Soto O, Camelli F, Baum JD, Luo H, Mestreau E, Sharov D, Ramamurti R, Sandberg W, Oh Ch. Advances in FEFLO. *AIAA-01-0592*, 2001.
27. Löhner R, Parikh P. Three-dimensional grid generation by the advancing front method. *International Journal for Numerical Methods in Fluids* 1988; **8**:1135–1149.
28. Löhner R. Extensions and improvements of the advancing front grid generation technique. *Communications in Numerical Methods in Engineering* 1996; **12**:683–702.
29. Löhner R, Yang C, Cebral J, Soto O, Camelli F, Baum JD, Luo H, Mestreau E, Sharov D. Advances in FEFLO. *AIAA-02-1024*, 2002.
30. Löhner R. Multistage explicit advective prediction for projection-type incompressible flow solvers. *Journal of Computational Physics* 2004; **195**:143–152.
31. Löhner R, Yang C, Cebral JR, Camelli F, Soto O, Waltz J. Improving the speed and accuracy of projection-type incompressible flow solvers. *Computer Methods in Applied Mechanics and Engineering* 2006; **195**:23–24, 3087–3109.
32. Löhner R, Parikh P, Gumbert C. Some algorithmic problems of plotting codes for unstructured grids. *AIAA-89-1981-CP*, 1989.
33. Cebral JR, Löhner R. Interactive on-line visualization and collaboration for parallel unstructured multidisciplinary applications. *AIAA-98-0077*, 1998.
34. Cho Y, Boluriaan S, Morris PJ. Immersed boundary method for viscous flow around moving bodies. *AIAA-06-1089*, 2006.
35. Löhner R, Oñate E. A general advancing front technique for filling space with arbitrary objects. *International Journal for Numerical Methods in Engineering* 2004; **61**:1977–1991.
36. Schlichting H. *Boundary Layer Theory*. McGraw-Hill: New York, 1979.
37. Fornberg B. Steady viscous flow past a sphere at high Reynolds number. *Journal of Fluid Mechanics* 1988; **190**:471.

DOI: 10.1002/fld