Journal of Computational Physics 257 (2014) 83–101

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

An immersed boundary method using unstructured anisotropic mesh adaptation combined with level-sets and penalization techniques

R. Abgrall [c,a,b,*], H. Beaugendre [a,b,c], C. Dobrzynski [a,b,c]

[a] Univ. Bordeaux, IMB, UMR 5251, F-33400 Talence, France
[b] CNRS, IMB, UMR 5251, F-33400 Talence, France
[c] INRIA, F-33400 Talence, France

ARTICLE INFO

Article history:
Received 30 May 2012
Received in revised form 2 May 2013
Accepted 19 August 2013
Available online 25 September 2013

Keywords:
Penalization technique
Unstructured mesh
Level-set
Anisotropic mesh
Mesh adaptation
Embedded method
Navier–Stokes equations

ABSTRACT

The interest on embedded boundary methods increases in Computational Fluid Dynamics (CFD) because they simplify the mesh generation problem in the case of the Navier–Stokes equations. The same simplifications occur for the simulation of multi-physics flows, the coupling of fluid–solid interactions in situation of large motions or deformations, to give a few examples. Nevertheless an accurate treatment of the wall boundary conditions remains an issue of the method. In this work, the wall boundary conditions are easily taken into account through a penalization technique, and the accuracy of the method is recovered using mesh adaptation, thanks to the potential of unstructured meshes. Several classical examples are used to demonstrate that claim.

1. Introduction

When dealing with CFD simulations two types of grids are most commonly used: body-fitted grids and embedded grids. In the case of body-fitted grids, the external mesh faces match up with the body surfaces and external boundary faces of the domain. This is different for embedded approach also known as fictitious domain, immersed boundary method (IBM) or Cartesian method. Indeed when considering embedded techniques, the bodies are immersed inside a large mesh, most of the time a Cartesian mesh, and special treatments of the elements close to the body surfaces are performed. When considering general cases of moving or deforming surfaces along with topological changes, both approaches have complementary strengths and weaknesses.

When dealing with moving bodies using body-fitted grids, the partial differential equations (PDEs) describing the flow have to be cast in an Arbitrary Lagrangian–Eulerian frame of reference (ALE), see e.g. [5,37,32,16,17]. The idea is to move the mesh in such a way as to minimize its distortion and if required the mesh can be regenerated and/or adapted and the solution interpolated [31] or not such as in [29] where a tricky space–time formalism is used. Each of these steps (ALE, mesh movement, interpolation) have been optimized but the topology reconstruction may fail for singular surface points and the interpolation required between grids may lead to some loss of information.

On the contrary, embedded grids methods are attractive: the PDE formulation remains in an Eulerian frame of reference even when moving bodies are considered, this formulation simplify *a priori* the meshing issue. Most developments made

* Corresponding author.

0021-9991/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jcp.2013.08.052

using embedded-grids are performed using structured grids [3,36,35,28], though some work using unstructured grids using fixed embedded grids have already been proposed to solve fluid/structure interactions by Wang et al. [42] and Farhat et al. [42], or in Löhner et al. [33] for Computational Structural Dynamics (CSD), with special boundary treatments in order to prevent penetration during contact. In the last decades, different embedded approaches (fictitious domain, IBM, penalization, etc.) have been developed such that flows around complex geometries can be computed [10]. Immersed boundary methods can now deal with incompressible flows [10] or compressible viscous flows [24] and even turbulent flows using mesh stretching and wall law turbulence formulation [30]. However, the drawback of embedded boundary methods remains the complicated treatment of wall boundary conditions in general. Recent developments in embedded boundary methods have focused on that point with algorithms for interface treatment to improve high order accuracy [25,28,34,41] and/or ghost-cell technique [24].

In this work, we introduce the Immersed Boundary Method with Level Sets and Adapted Unstructured Meshes (IBM-LS-AUM) method. The idea is to combine the strength of mesh adaptation, that is to provide an accurate flow description especially when dealing with wall boundary conditions, to the simplicity of embedded grids techniques, that is to simplify the meshing issue and the wall boundary treatment when combined with penalization term to enforce boundary conditions. The bodies are described using the level-set method [40] and are embedded in an unstructured grid. The wall boundary conditions are enforced by a penalization term [3]. Once a first numerical solution is computed, mesh adaptation [20,9] using quality criteria on the level-set and the solution is re-evaluated. In our opinion, the advantage of this strategy is the following: though simple to implement, the IBM techniques suffer a lack of accuracy near the walls. There, the method is consistent but not very accurate. This is why mesh adaptation is employed, in order to remedy to this behavior by a reduced mesh size near the boundaries of interest. Since most of the IMB schemes use Cartesian meshes, this leads to AMR-like grids, see for example [39], where the mesh is no longer conformal. Here we have chosen a different strategy. In order to have simple data structure, we have focused on a numerical method that uses conformal meshes. Our particular choice is not fundamental. In order to adapt the mesh easily, we focus on simplicial meshes using triangles and tets. The localization of the solid bodies is done via a level set method, as in [42], but with mesh adaptation in order to improve both the quality of the surface representation and of the solution. The quality of the adapted mesh is controlled automatically. Of course, the challenge is to control the increase of refined elements, so that the CPU cost of the simulation remains of the same order as the cost of a simulation with a similar mesh, but with a body-fitted geometry. Up to our knowledge, none of the existing embedded method combine these three features together.

This paper is organized as follows: we first describe the system we use, including the penalization terms, and we explain the numerical strategy for one given mesh. We discuss the structure of the penalization. Section 3 describes our anisotropic mesh adaptation strategy and how it is coupled to the system of interest. The IBM-LS-AUM method is tested against several classical problems, the results and the discussion are reported in Section 4. We discuss the choice of the penalization parameter and the cost associated to the method in term of memory requirements. We also discuss qualitatively the quality of the solution, and show that the accuracy of the IBM-LS-AUM solver is similar to the same CFD solver where the boundary conditions are weakly imposed, as in [18]. Some conclusions and perspectives for future work follow.

2. Penalization

2.1. Penalization method for compressible flows

We consider in this work a laminar flow described by the compressible Navier–Stokes equations. The conservative form of the Navier–Stokes equations can be written as follows (Gravity effects are assumed negligible):

$$\begin{aligned}
&\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial \boldsymbol{x}} \cdot (\rho \boldsymbol{u}) = 0 \\
&\frac{\partial}{\partial t}(\rho \boldsymbol{u}) + \frac{\partial}{\partial \boldsymbol{x}} \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u}) + \frac{\partial p}{\partial \boldsymbol{x}} = \frac{\partial}{\partial \boldsymbol{x}} \cdot \underline{\boldsymbol{\pi}} \\
&\frac{\partial}{\partial t}(\rho e) + \frac{\partial}{\partial \boldsymbol{x}} \cdot \big((\rho e + p)\boldsymbol{u}\big) = \frac{\partial}{\partial \boldsymbol{x}} \cdot (\underline{\boldsymbol{\pi}} \boldsymbol{u} + \boldsymbol{q})
\end{aligned} \tag{1a}$$

where ρ is the density, $\boldsymbol{u}$ the velocity, e the specific total energy. The pressure p and the heat flux $\boldsymbol{q}$ are respectively defined by the perfect gas equation of state and the Fourier law:

$$p = (\gamma - 1)\rho T \quad \text{and} \quad \boldsymbol{q} = -\frac{c_p(\mu)}{Pr}\frac{\partial T}{\partial \boldsymbol{x}} \quad \text{with} \quad \epsilon = e - \frac{1}{2}|\boldsymbol{u}|^2 \text{ and } \epsilon = c_v T.$$

Pr is the laminar Prandtl number, c_p is the heat capacity at constant pressure, c_v is the heat capacity at constant volume (here set to unity), ϵ the specific internal energy, T is the temperature and μ is the laminar viscosity. Here, $Pr = 0.72$. For Newtonian compressible fluids the stress tensor is given by

$$\underline{\boldsymbol{\pi}} = \mu\left(\left[\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}}\right] + \left[\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}}\right]^T - \frac{2}{3}\left[\frac{\partial}{\partial \boldsymbol{x}} \cdot \boldsymbol{u}\right]\underline{\mathbf{Id}}\right).$$

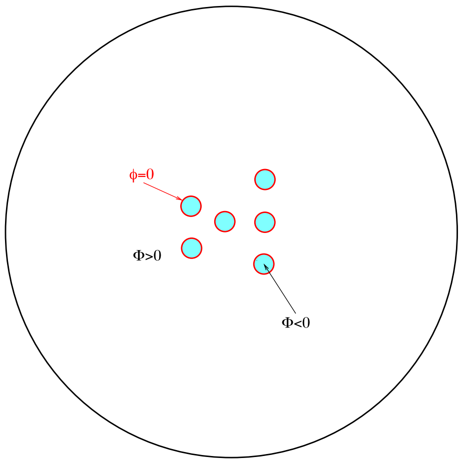


Fig. 1. Schematic representation of a computational domain $\Omega = \Omega_f \cup \Omega_s$. The solid domain is $\Omega_s = \cup_{i=1}^{6} S^i$ where the S_is are the individual solid domains. Ω_f is the fluid one. The outer boundary is $\Gamma = \partial\Omega$. The solid boundaries are $\partial\Omega_s = \bigcup \partial S_i$.

A penalization method [3], is used to enforce the no-slip boundary condition inside the solid wall boundaries. The solids around which the flow is computed are defined using the so-called penalization method or Brinckman–Navier–Stokes equations. Here, the solids are considered as porous media with a very small intrinsic permeability. One level set function, Φ, is used to capture interfaces and compute rigid motions of the solid bodies, [40]. Given a computational domain Ω, we consider a compressible flow in Ω_f around rigid solids S^i. A schematic representation of a computational domain composed of six solids is sketched on Fig. 1.

The fluid–solid interaction problem can be modeled by the compressible Navier–Stokes equations (1a) along with the following conditions:

$$\begin{aligned} &\boldsymbol{u} = \boldsymbol{u}_{s^i} \quad \text{on } \partial S^i, \\ &\boldsymbol{u} = \boldsymbol{u}_f \quad \text{on } \partial\Omega_f \end{aligned} \tag{1b}$$

and Dirichlet and Neuman conditions on the temperature. Depending on the location in the computational domain, the velocity is either the fluid velocity $\boldsymbol{u}_f$, either the solid velocity $\boldsymbol{u}_{s^i}$. The idea is to extend the velocity field inside the solid body and to solve the flow equations with a penalization term to enforce rigid motion inside the solid [11].

Given a penalization parameter $\frac{1}{\eta} \gg 1$, and denoting by χ_{s^i} the characteristic function of the solid S^i, the model equation writes

$$\begin{aligned} &\frac{\partial\rho}{\partial t} + \frac{\partial}{\partial\boldsymbol{x}} \cdot (\rho\boldsymbol{u}) = 0 \\ &\frac{\partial}{\partial t}(\rho\boldsymbol{u}) + \frac{\partial}{\partial\boldsymbol{x}} \cdot (\rho\boldsymbol{u}\otimes\boldsymbol{u}) + \frac{\partial p}{\partial\boldsymbol{x}} = \frac{\partial}{\partial\boldsymbol{x}} \cdot \underline{\boldsymbol{\pi}} + \frac{1}{\eta}\sum_{i=1}^{Ns} \chi_{s^i}(\rho\boldsymbol{u} - \rho\boldsymbol{u}_{s^i}) \\ &\frac{\partial}{\partial t}(\rho e) + \frac{\partial}{\partial\boldsymbol{x}} \cdot \big((\rho e + p)\boldsymbol{u}\big) = \frac{\partial}{\partial\boldsymbol{x}} \cdot (\underline{\boldsymbol{\pi}}\boldsymbol{u} + \boldsymbol{q}) \\ &\qquad\qquad + \frac{1}{\eta}\sum_{i=1}^{Ns} \theta_{s^i}\,\chi_{s^i}\rho\big(\epsilon(T) - \epsilon(T_{s^i})\big) + \frac{1}{\eta}\sum_{i=1}^{Ns} \chi_{s^i}(\rho\boldsymbol{u} - \rho\boldsymbol{u}_{s^i}) \cdot \boldsymbol{u}. \end{aligned} \tag{2}$$

The derivation of the system (2) is carried out in Appendix A.3. It is also shown to be Galilean invariant.

Note that, since the penalization parameter is large, the actual value of the temperature inside the solid plays a role only in the vicinity of the solid surface. This gives the possibility to handle nonuniform Dirichlet condition, in which case we set $\theta_{s^i} = 1$. For a Neumann boundary condition on the temperature (adiabatic wall), we do not impose any additional constraint, i.e. we set $\theta_{s^i} = 0$. The discretization and the integration of the penalization term affect the choice of the penalization parameter $\frac{1}{\eta}$: the larger the parameter, the better the quality of the penalization. In this work χ_{s^i} is computed from a level set function Φ_{s^i}. We initialize Φ_{s^i} as the signed distance function to the boundary of S^i, because Φ_{s^i} is positive outside S^i and negative inside the solid:

$$\chi_{s^i} = H(-\Phi_{s^i}), \tag{3}$$

where H is the Heaviside function.

Note that if we are considering moving bodies Φ_{s^i} has to satisfy the same advection equation as χ_{s^i}:

$$\frac{\partial \Phi_{s^i}}{\partial t} + (\mathbf{u}_{s^i} \cdot \nabla)\Phi_{s^i} = 0 \quad \text{for } \mathbf{x} \in \Omega. \tag{4}$$

In this case, it is important to notice that, since $\boldsymbol{u}_{s^i}$ is a rigid body motion, one can guarantee that Φ_{s^i} remains a signed distance for all time. To summarize, each body–fluid interface is captured by a level set function.

The global model considered in this context can be written in the following compact form

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial}{\partial \boldsymbol{x}} \cdot \underline{\boldsymbol{F}} = \frac{\partial}{\partial \boldsymbol{x}} \cdot \underline{\boldsymbol{G}} + \boldsymbol{S} \tag{5}$$

where the vector $\boldsymbol{U}$ of conservative variables, advection $\underline{\boldsymbol{F}}$ and viscous $\underline{\boldsymbol{G}}$ flux tensors are defined as:

$$\boldsymbol{U} = \begin{pmatrix} \rho \\ \rho \boldsymbol{u} \\ \rho e \end{pmatrix}, \qquad \underline{\boldsymbol{F}} = \begin{pmatrix} \rho \boldsymbol{u} \\ \rho \boldsymbol{u} \otimes \boldsymbol{u} + p\underline{\mathbf{Id}} \\ (\rho e + p)\boldsymbol{u} \end{pmatrix} \quad \text{and} \quad \underline{\boldsymbol{G}} = \begin{pmatrix} 0 \\ \underline{\boldsymbol{\pi}} \\ \underline{\boldsymbol{\pi}}\boldsymbol{u} + \boldsymbol{q} \end{pmatrix}.$$

Finally, the penalization term holds in the vector $\boldsymbol{S}$:

$$\boldsymbol{S} = \frac{1}{\eta} \sum_{i=1}^{Ns} \chi_{s^i} \begin{pmatrix} 0 \\ \rho \boldsymbol{u} - \rho \boldsymbol{u}_{s^i} \\ \theta_{s^i} \rho(\epsilon(T) - \epsilon(T_{s^i})) + \rho(\boldsymbol{u} - \boldsymbol{u}_s) \cdot \boldsymbol{u} \end{pmatrix}.$$

The convective part of this system (when right hand size is set to zero) is hyperbolic. Indeed, for any direction $\boldsymbol{n}$ we define the matrix $\underline{\boldsymbol{A}}(\boldsymbol{n})$ as

$$\underline{\boldsymbol{A}}(\boldsymbol{n}) = \frac{\partial \underline{\boldsymbol{F}}\boldsymbol{n}}{\partial \boldsymbol{U}} \quad \text{with } \underline{\boldsymbol{F}}\boldsymbol{n} = \begin{pmatrix} \rho \boldsymbol{u} \cdot \boldsymbol{n} \\ \rho \boldsymbol{u}(\boldsymbol{u} \cdot \boldsymbol{n}) + p\boldsymbol{n} \\ (\rho e + p)\boldsymbol{u} \cdot \boldsymbol{n} \end{pmatrix}.$$

For any $\boldsymbol{n}$ with $|\boldsymbol{n}| \neq 0$, the matrix $\underline{\boldsymbol{A}}(\boldsymbol{n})$ is diagonalizable and have three different eigenvalues

$$\lambda_- = \boldsymbol{u} \cdot \boldsymbol{n} - a\|\boldsymbol{n}\|, \qquad \lambda_0 = \boldsymbol{u} \cdot \boldsymbol{n} \quad \text{and} \quad \lambda_+ = \boldsymbol{u} \cdot \boldsymbol{n} + a\|\boldsymbol{n}\|,$$

where $a = \sqrt{\frac{\gamma p}{\rho}}$ is the sound speed. We denote by $\underline{\boldsymbol{P}}(\boldsymbol{n})$ and $\underline{\boldsymbol{\Lambda}}(\boldsymbol{n})$ respectively the matrices of eigenvectors and eigenvalues of $\underline{\boldsymbol{A}}(\boldsymbol{n})$:

$$\underline{\boldsymbol{A}}(\boldsymbol{n}) = \big[\underline{\boldsymbol{P}}(\boldsymbol{n})\big]\underline{\boldsymbol{\Lambda}}(\boldsymbol{n})\big[\underline{\boldsymbol{P}}(\boldsymbol{n})\big]^{-1}$$

this spectral decomposition is useful for upwinding, streamlines stabilization and boundary conditions and will be used inside our discretization.

2.2. Discretization

Our numerical simulations are performed for simplices meshes (2D-triangles and 3D-tetrahedrons) with piecewise linear Lagrange test functions. We denote by T_h such a triangulation, the generic simplex is denoted by K. We denote the computational domain by $\Omega_h = \bigcup_{K \in T_h} K$. The obstacles are localized using the zero isovalue of a static level-set function. The system to be solved writes: find $\mathbf{U}_h$ such that

$$\int_{\Omega_h} \mathbf{W}_h \mathcal{R}(\mathbf{U}_h)\, d\Omega = 0 \tag{6}$$

where $\mathcal{R}(\mathbf{U}_h)$ is the residual of Eq. (5) that is

$$\mathcal{R}(\mathbf{U}_h) = \frac{\partial \boldsymbol{U}_h}{\partial t} + \frac{\partial}{\partial \boldsymbol{x}} \cdot \underline{\boldsymbol{F}} - \frac{\partial}{\partial \boldsymbol{x}} \cdot \underline{\boldsymbol{G}} - \boldsymbol{S} := \frac{\partial \boldsymbol{U}_h}{\partial t} + \Psi(\mathbf{U}_h). \tag{7}$$

The time domain $[0, T]$ is subdivided into nonoverlapping intervals $]t^n, t^{n+1}[$, $\Delta t_{n+1/2} = t_{n+1} - t_n$. The numerical solution at a time t^n is denoted by $\mathbf{U}_h^n$. We are given an initial condition $\mathbf{U}_h^0$ we assume that $\mathbf{U}_h^{-1} \equiv \mathbf{U}_h^0$ and $t^{-1} = t^0$. Therefore the system to be solved at each time step is to find $(\mathbf{U}_h^{n+1})_{n \geqslant 0}$ such that

$$\int_{\Omega_h} \mathbf{W}_h \frac{\mathbf{U}_h^{n+1} - \mathbf{U}_h^n}{\Delta t_{n+1/2}} d\Omega + \int_{\Omega_h} \mathbf{W}_h \Psi\big(\mathbf{U}_h^{n+\theta}\big)\, d\Omega + \int_{\partial\Omega_h} \mathbf{Bc}\big(\mathbf{W}_h, \mathbf{U}_h^{n+\theta}\big) = 0 \tag{8}$$

where $\mathbf{U}_h^{n+\theta} = \mathbf{U}_h^n + \theta(\mathbf{U}_h^{n+1} - \mathbf{U}_h^n)$, $\theta \geqslant 0$ and the operator $\mathbf{Bc}$ takes into account all contributions for boundary conditions.

We briefly describe our numerical strategy:

1. The system of equations is discretized using a classical mixed Finite Element/Finite Volume scheme. Finite Volumes are used for the convective part of the system, see [18], which is a generalization of the MUSCL method on simplicial meshes. Gradient extrapolation is done on the primitive variables, and the van Leer–van Albada limiter is used. The numerical fluxes are Roe's or HLLC's, and Steger Warming's for boundary conditions at the inlet and outlet of the domain. The diffusive part of the system and source terms are discretized using $\mathbb{P}^1$ Finite Elements on the triangulation with the Galerkin approximation. Note there is no need here to use no slip boundary conditions. The approximation of the penalization terms is related to the time approximation for stability reasons.
2. The time derivative is approximated as follows. For time dependent problems, the most accurate second order scheme is in general obtained with the semi-implicit scheme associated to $\theta = \frac{1}{2}$ (Crank–Nicholson scheme). The explicit scheme associated to $\theta = 0$ is subject to CFL stability condition and do not have proper behavior in this context of penalization, because the CFL will be directly linked to the choice of $\frac{1}{\eta}$: an Euler explicit time discretization does not allow to use $\frac{1}{\eta} > 1/\Delta t$. To obtain accuracy using penalization technique, we need to take $\frac{1}{\eta} \gg 1$. Hence, in practice, we always use implicit schemes with $\frac{1}{\eta} = 10^{12}$.
3. Therefore, we have to solve, at each time step, a nonlinear system. This resolution is achieved by Newton-type relaxations, each relaxation step is defined by a linear sparse system of large size, obtained by a linearized implicit approximation:

$$\Psi\big(\mathbf{U}_h^{n+\theta}\big) \simeq \Psi\big(\mathbf{U}_h^n\big) + \theta\left(\frac{\partial \Psi}{\partial \mathbf{U}_h}\right)^n \big(\mathbf{U}_h^{n+1} - \mathbf{U}_h^n\big)$$

$$\mathbf{Bc}\big(\mathbf{W}_h, \mathbf{U}_h^{n+\theta}\big) \simeq \mathbf{Bc}\big(\mathbf{W}_h, \mathbf{U}_h^n\big) + \theta\left(\frac{\partial \mathbf{Bc}}{\partial \mathbf{U}_h}\right)^n \big(\mathbf{U}_h^{n+1} - \mathbf{U}_h^n\big).$$

   The approximated Jacobians are again obtained by the same method as in [18].

This numerical strategy has been deployed in the computing platform "Realfluids" that provides tools for high order time integration. Parallelism is achieved by domain decomposition and message passing using MPI. The domain decomposition is performed via Metis [1] or the SCOTCH Library [38]. The linear systems are solved by standard techniques (preconditioned GMRES). The preconditioning methods are either of the ILU(0) type or by using locally or globally the direct solvers contained in the PASTIX library [27,8]. Our computations have been performed up to 64 processors on the cluster PlaFRIM from INRIA Bordeaux Sud-Ouest. However, parallel efficiency of Realfluids platform has been demonstrated up to 512 processors.

3. Anisotropic mesh adaptation

3.1. Mesh adaptation background

In this section, we briefly explain the theoretical framework for anisotropic meshing and describe the important features of the mesh adaptation algorithm. For more details, we refer to [23,2,4,7,14,15,19,20,26]. In particular it is shown in these references that the local error can be estimated on each simplex K by

$$\big\|u - \Pi_h(u)\big\|_\infty \leqslant c \max_{y\in K} \max_{x\in K}\big(x^T \big|H_u(y)\big| x\big)$$

where $\Pi_h(u)$ is the Lagrange interpolant, the constant c only depends on the type of element, x is a vector in K and $|H_u|$ is the absolute value of the Hessian of u, which is a symmetric matrix. The Hessian is estimated from discrete data by a technique detailed in the above references, keeping its symmetric nature. This metric is denoted by M and is local.

The aim of anisotropic mesh adaptation is to control the size, the shape and the orientation of mesh elements. These specifications are usually based on an error estimate and they are written via a metric tensor.

3.1.1. Metric tensors

This metric tensor $M(x)$ is represented by a $d \times d$ ($d = 2, 3$) symmetric definite positive matrix. The eigenvalues (resp. eigenvectors) are related to the desired sizes (resp. directions) of the element edges. This metric tensor is used to generate a quasi-uniform mesh of the domain in this metric. This means we would have unitary volume of an element K in the discretized domain T_h:

$$\int_K \sqrt{\det\big(M(x)\big)}\, dx = 1 \quad \forall K \in T_h.$$

The length of the vector $\mathbf{e}$ with respect to the metric $\mathcal{M}(x)$ is defined as [22]:

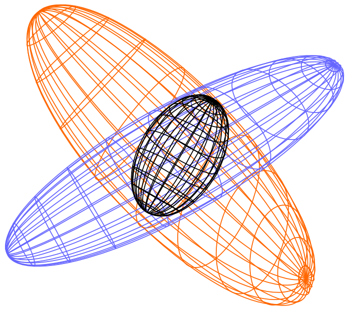


Fig. 2. Geometrical representation of metric intersection.

$$l_{\mathcal{M}}(\mathbf{e}) = \int_0^1 \sqrt{\mathbf{e}^T \mathcal{M}(t)\mathbf{e}}\, dt. \tag{9}$$

3.1.2. Metric interpolation

In practice, the metric is often supplied at mesh vertices and we need to define a metric at each point on a mesh edge. To this end, we define a continuous metric field along the mesh edge by using a linear interpolation scheme.

If P and Q are the endpoints of an edge parametrized by

$$e(t) = (1-t)P + tQ, \quad 0 \leqslant t \leqslant 1$$

and M_P (resp. M_Q) the metric associated to P (resp. Q), the interpolation scheme is defined by:

$$M(t) = \left((1-t)M_P^{-\frac{1}{2}} + tM_Q^{-\frac{1}{2}}\right)^{-2}.$$

This choice provides smoother results, see [21].

3.1.3. Metric intersection

If several metrics are defined at one vertex $P \in T_h$, we define a single metric by using the intersection metric:

$$\mathcal{M}_\cap = \mathcal{M}_1 \cap \mathcal{M}_2.$$

Geometrically speaking, it consists in defining the largest ellipsoid included in the intersection of all the ellipsoids associated to the considered vertex. This is illustrated in the case of two metrics on the Fig. 2 for which the black ellipse should be used for the adaptation of the mesh. From a practical point of view, we use simultaneous reduction of the two metric tensors, see [21] for more details.

3.2. Level-set adaptation: metric definition

Solid bodies around which flow computations are performed are defined by the zero isovalue of a level-set function. In this case the level-set function is the signed distance function, positive outside the solid and negative inside. Obtaining a good approximation of the contour of the solid bodies means to have an accurate tracking of their boundaries. Those boundaries are given by the zero isovalue, so the idea is to put enough elements in the vicinity of this isovalue and minimize the piecewise affine approximation of those boundaries. To this end, in [9], Frey et al. have defined a metric to control the geometric approximation of an isovalue of a level-set function φ. Let's ε be an error, $h_{\min}$ (resp. $h_{\max}$) the minimal (resp. max.) length edge. The metric is defined by:

$$M = R \begin{pmatrix} \frac{1}{\varepsilon^2} & 0 & 0 \\ 0 & \frac{|\lambda_1|}{\varepsilon} & 0 \\ 0 & 0 & \frac{|\lambda_2|}{\varepsilon} \end{pmatrix} R^T \tag{10}$$

with $R = (\nabla\varphi, v_1, v_2)$ where (v_1, v_2) is a basis of the tangent plane to the boundary and λ_i are the eigenvalues of the Hessian of φ (i.e. the principal curvatures of the isoline). Note that $\nabla\varphi$ is the normal to the isoline. We impose this metric to the vertices in the vicinity of the solid objects. For the other vertices, $h_{\min}$ and ε are increased linearly up to $h_{\max}$.

3.3. Mesh adaptation to flow features

In order to have a good accuracy of the physical solution with a minimum number of nodes, we use classical anisotropic mesh adaptation based on interpolation error.

Let u be a variable and $\Pi_h u$ its $\mathbb{P}^1$ interpolant. We have the following estimation of the interpolation error on a mesh element K [20]:

$$\|u - \Pi_h u\|_{\infty,K} \leqslant c \max_{e \in E_K} \langle e, \mathcal{M}(K) e \rangle,$$

with e a mesh edge and E_K the set of mesh edges. In this estimation, $\mathcal{M}(K)$ is a metric tensor computed with the Hessian of u and defined by:

$$\mathcal{M} = \mathcal{R} \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} \mathcal{R}^{-1} \tag{11}$$

where $\mathcal{R}$ is the matrix of the eigenvectors of the Hessian of u and

$$\lambda_i = \min\left(\max\left(|h_i|, \frac{1}{h_{\max}^2}\right), \frac{1}{h_{\min}^2}\right). \tag{12}$$

Here the h_i's are the eigenvalues of the Hessian of u.

We would equidistribute the interpolation error over all the mesh so if ε is the desired error. Thanks to the previous estimate, we would have for any edge vector $\mathbf{e}$: $\varepsilon = c\langle \mathbf{e}, \mathcal{M}(K)\mathbf{e}\rangle$, i.e.

$$\left\langle \mathbf{e}, \frac{c}{\varepsilon}\mathcal{M}(K)\mathbf{e} \right\rangle = 1.$$

This means that we search to have edges with their lengths equal to unity in the metric $\frac{c}{\varepsilon}\mathcal{M}(K)$. More details on the construction of the two metrics can be found in Appendix A.

Remark 1. For practical applications, it is better to consider the relative error:

$$\left\| \frac{u - \Pi_h u}{u} \right\|_{\infty,K} \leqslant c \max_{e \in E_K} \left\langle \mathbf{e}, \frac{\mathcal{M}(K)}{u}\mathbf{e} \right\rangle. \tag{13}$$

This choice is motivated by the following observation: if we need to fulfill simultaneously several criterion, we want to give each of them an equivalent weight. This is the case here since we adapt the mesh on a physical criterion and on the zero isovalue of the level-set. Of course the same adimensionalization is applied for the criteria discussed in Section 3.2.

3.4. The IBM-LS-AUM algorithm

The Immersed Boundary Method with Level Sets and Adapted Unstructured Meshes is the combination of the methods described in Section 2.2 with the anisotropic Mesh adaptation of Section 3 applied on the system (2).

We adapt the mesh on two criteria: the level set and some physical parameter which are problem dependent:

- For the adaptation with respect of the level set, the goal is to get an accurate enough description of the zero level set, i.e. the solid boundary, and to control the geometry of the mesh element in the vicinity of the zero level set. This is achieved by setting a tolerance on the knowledge of the zero level set location, this parameter is denoted ε and set to 10^{-3} in this paper. We also impose the shape of the first layer of elements in the vicinity of the zero level set, namely $h_{\min}$ their size in the normal direction, and $h_{\max}$ their size in the zero level set direction. In general, the zero level set does not correspond to element faces, so we need to introduce a layer centered around the zero level set where the size of the elements is prescribed: $h_{\min}$ in the direction normal to the isosurface of the level set, and $h_{\max}$ on the isosurfaces. After this buffer zone, $h_{\min}$ grows linearly up to $h_{\max}$ so that the elements become regular far enough from the zero isoline. These parameters are described on Fig. 3.
- Concerning the parameters for the adaptation on a physical criteria, this is more ad-hoc and depends on the flow configuration. In each case we have described what is done.

Throughout the simulations for one given configuration, the adaptation criteria are not changed. We first start from an initial mesh, then apply the scheme of Section 2.2 on (2). Then we adapt the mesh with the method of Section 3, and restart the simulation. Usually 2–3 iterations are needed before convergence is reached.

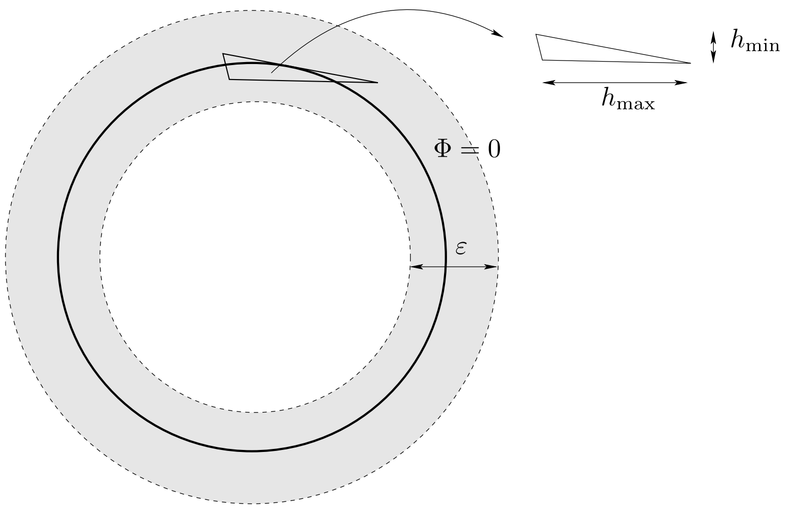


Fig. 3. Definition of the various parameters used for the level set adaptation. The plain line is the zero isoline. The zone between the dotted lines is the buffer zone where we impose $h_{\min}$ in the direction normal to the isosurface of the level set, and $h_{\max}$ on the isosurfaces.

4. Numerical results

We propose in this section four test cases to demonstrate the ability of the method to obtain accurate solutions together with an accurate wall treatment starting with initial meshes that does not contain any point on the zero level-set of the solid body.

In the first test case, we check the ability of the method to model and predict the laminar boundary layer over a flat plate. The solution is then compared to the Blasius solution. The second test case demonstrates the performance of the method to position correctly a shock wave in the laminar regime. The third test case considers the laminar flow around a NACA 0012 airfoil. In this case, the penalized solution is compared to the body fitted one. The last test case is a 3D laminar flow around an ellipse: we demonstrate that the proposed method is suitable for 3D problems.

4.1. About the choice of the penalization parameter

As said before, the penalization parameter is set to $\eta = 10^{-12}$. Formally, the smaller the penalization parameter, the closer from the "exact" Navier–Stokes solution we should be. For each cases studied, we have tried several values of the parameter η, and after $\eta = 10^{-10}$, we do not see any change in the solution. Since our strategy is implicit, we have not encountered any difficulty in the numerical procedure. In this paper, we consider only steady problems, though we are using the strategy (8) which should be suitable in principle for unsteady simulations. Unsteady problems will be the topic of future research, we do not believe that our conclusions will be changed.

4.2. Blasius test case

There are not that many cases where exact solutions are known on problems with simple geometries. With exact solution, we can evaluate in a rigorous way the numerics, with simple geometries, the artifacts generated by the implementation of the boundary conditions are certainly reduced to a minimum. The Blasius test case is one of these few examples. We use it to discuss the quality of the solution obtained, and also to compare our method to a standard one for body fitted geometry. The only difference between the two methods is the implementation of the boundary conditions. In the case of the body fitted scheme, we strongly impose the velocity $\mathbf{u} = \mathbf{0}$ and the normal temperature gradient $\frac{\partial T}{\partial n} = 0$ is weakly imposed.

The velocity distribution $U(x)$ of the outer flow of an incompressible flat plate when dealing with laminar flow is given as a power series in x, where x is the coordinate measured along the flat plate. The velocity distribution in the boundary layer is then also expanded in a power series, called the Blasius series, where the coefficients are still functions of the y coordinate. For this first test case, the numerical solution of the Navier–Stokes equations in the laminar regime ($Re = 500$) is compared to the analytical solution derived from Blasius series on a series of meshes. Our goal is to investigate the role of the penalization term, to verify the quality of the solution along with mesh adaption, the test case is constructed as a backward facing step test case, see Fig. 4, where the step is all penalized. In order to make a detailed comparison, we have run the same case, with the same meshing strategy with a body-fitted scheme.

We start the computations on coarse meshes, one for the body-fitted test case and one for the penalized test case. The density of nodes are identical for the two starting meshes, the meshes are composed of equilateral triangles having the

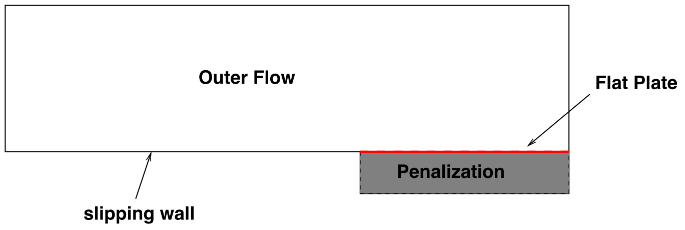


Fig. 4. Blasius setup.

Table 1
Comparison between the body-fitted solution and the penalized one with and without mesh adaptation.

Fitted mesh	# vertices	# triangles	Penalized mesh	# vertices	# triangles
initial	7058	13 794	initial	8080	15 838
adapt 1	1933	3705	adapt 1	2786	5401
adapt 2	2376	4539	adapt 2	3550	6919
adapt 3	2680	5132	adapt 3	3632	7086

same size everywhere. We note that the solution is well approximated after only two cycles of mesh adaptation for the fitted grid, and after one cycle for the penalized grid. Once the solution is computed mesh adaptation constrained by the distance to the level-set and the Hessian of the solution is performed. The criteria on the solution is done on the component U of the velocity. The adaptation parameters for U are

$$\varepsilon = 10^{-3}; \qquad h_{\min} = 10^{-3}; \qquad h_{\max} = 2$$

and for the adaptation on the level-set are

$$\varepsilon = 10^{-3}; \qquad h_{\min} = 10^{-3}; \qquad h_{\max} = 0.1.$$

We notice that the adaptation on the two criteria (density and level set) increases in one shot the density of nodes close to the wall and by the way allows to capture, in that case, the exact solution after only one mesh adaptation. Table 1 gives the number of vertices and simplices for the different meshes used in this study. We see that these numbers are of the same order, remember that the domain covered by the IBM-LS-AUM method is larger than for the body-fitted one. The method does not add more than 30% nodes. We have performed several other cases, with different step size, and the same conclusion hold: the number of points on the final adapted mesh is similar to the one of the body-fitted mesh, for the same quality of results. Figs. 5 and 6 provide zoom of the meshes at the leading edge of the flat plate, Figs. 7 and 8 show the mesh at the trailing edge of the plate. The density of point in the body-fitted case and the penalized cases are similar.

Two dimensional cuts of the velocity distribution U/U_{inf} as a function of the nondimensional coordinate Y/Y_{inf} are evaluated at the end of the flat plate, i.e. $x = 1$. The solutions obtained on the original mesh and the adapted ones for the two schemes are compared to the Blasius analytic one, see Fig. 9. Using penalization together with mesh adaptation, we obtain a solution in very good agreement with the theoretical one and very similar to the solutions obtained with a body-fitted mesh. Mesh adaptation enables to improve the quality of the solution at the interface and reduces the number of points needed for the test case, indeed the points are mainly clustered near the interface, i.e. around the zero level set.

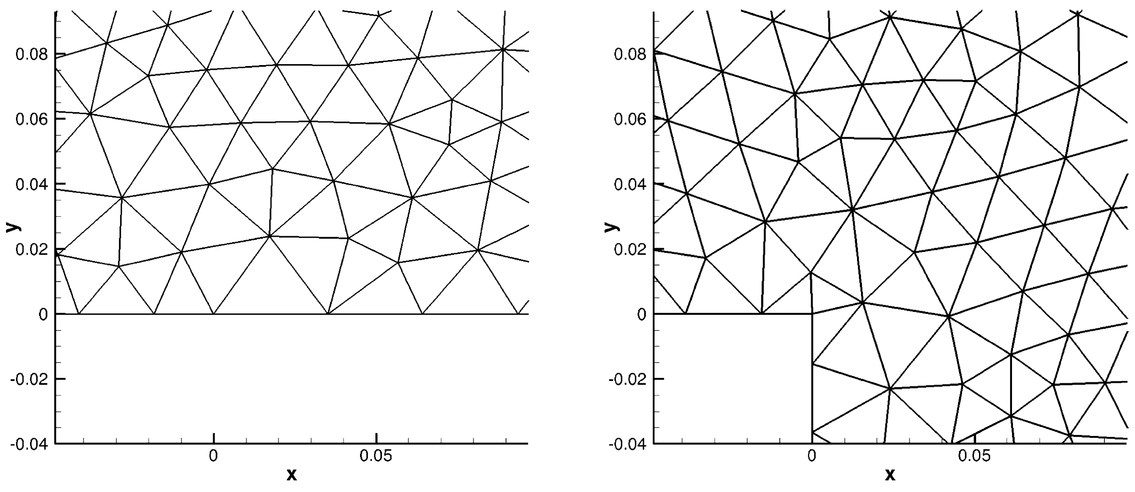
4.3. Supersonic flow around a triangle

We consider the supersonic flow around a solid body of triangular shape with height $h = 0.5$, half angle $\theta = 20°$ and $S = (0.5, 1)$, see Fig. 10. The flow is computed using the penalized Navier–Stokes equation (2). The same computational domain as [6] is chosen even if a smaller domain at the inlet could have been taken because we are dealing with supersonic flow. The computation is stopped when a steady state is obtained. The Reynolds number is fixed to $Re = 5 \times 10^4$, the Prandtl number to 0.72 and the Mach number to $M_1 = 2$. As in Boiron et al. [6] the velocity inside the triangle is set to zero and the nondimensional temperature is set to $T_s = 3$.

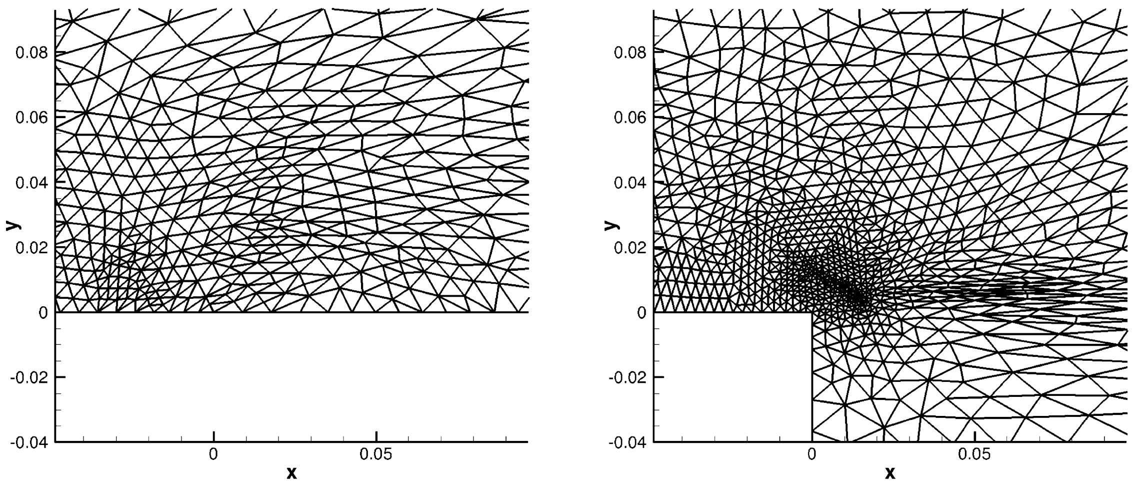
An oblique shock is predicted by the inviscid flow theory. It can be attached or detached depending on values of the angle θ and the Mach number. If the shock is attached to the triangle, its angle β with the horizontal is given by:

$$\tan\theta = 2\cot\beta\left[\frac{M_1^2\sin^2\beta - 1}{M_1^2(\gamma + \cos 2\beta) + 2}\right]. \tag{14}$$

The initial non-adapted mesh contains 28 141 nodes and 56 189 triangles, the mesh and the density distribution are presented in Fig. 11(a). The final adapted mesh, after 5 cycles of adaptations, contains 791 705 nodes and 1 583 221 triangles. The parameters for the adaptations are the following



(a) initial meshes



(b) after one adaptation

Fig. 5. Meshes for the leading edge: left fitted, right penalized. We display the initial meshes and after one adaptation.

on the velocity $\varepsilon = 10^{-4}$; $h_{\min} = 10^{-4}$; $h_{\max} = 2$

on the level set $\varepsilon = 10^{-4}$; $h_{\min} = 10^{-4}$; $h_{\max} = 2$.

The component u of the velocity and the adapted mesh are shown in Figs. 11(b) and 11(c).

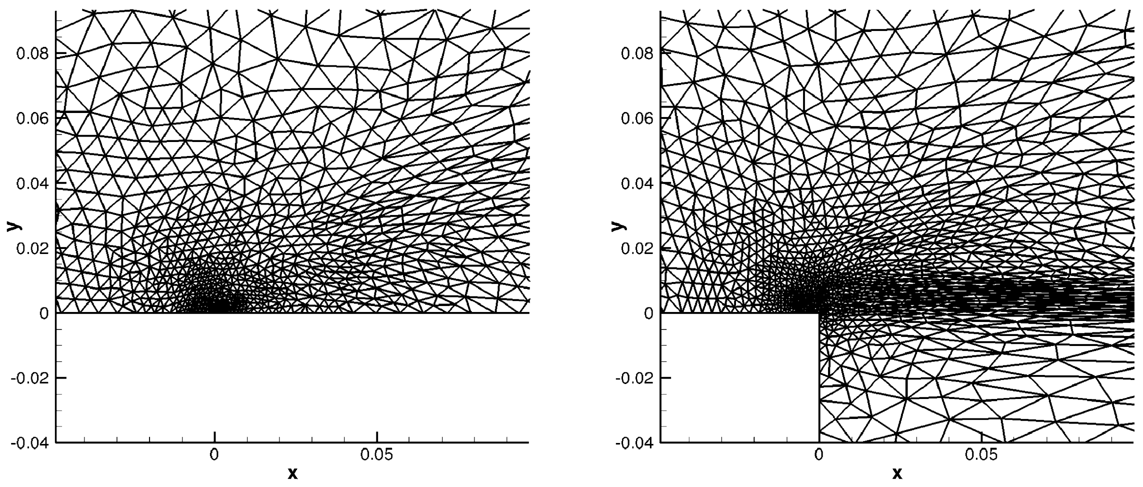
According to (14), the angle of the shock for this test case is such that

$$\tan\left(\frac{\pi}{9}\right) = 2\cot(\beta)\frac{4.\sin^2(\beta) - 1}{4(\gamma + \cos(2\beta)) + 2}, \tag{15}$$

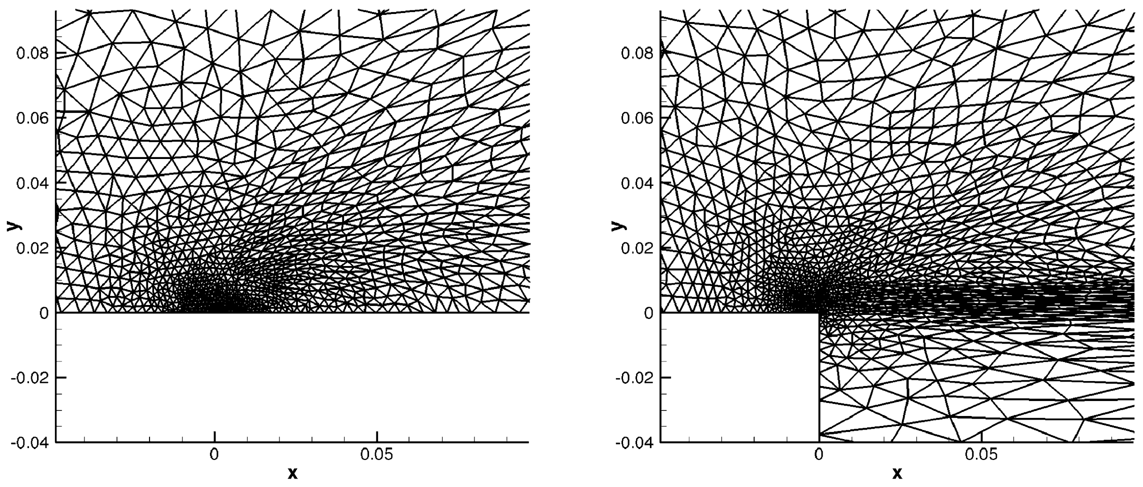
i.e. $\beta \approx 53.46°$. In our solution we measure the angle β at the same location as Boiron et al. [6], i.e. $y = 0$. We find an angle of about 53.8° compared to 53.7° found by [6]. In [6] the computation is performed on a Cartesian grid: 1024×1024, and the domain is $[0, 2] \times [0, 2]$. A 1D cut of the pressure along the line $y = 1.44$ is compared to Boiron et al. [6] and presented in Fig. 12. Our results are in good agreement with the theory and the numerical solutions performed in [6]. In Fig. 12, we notice that our solution is able to compute a sharper shock wave with less diffusion and with less points than Boiron et al. Note also that we are computing the solution on a complete domain avoiding blocking effects.

4.4. Flow around a NACA0012 airfoil

The third test case proposed consists on a laminar subsonic flow around a symmetrical NACA 0012 airfoil. The Reynolds number of the solution is set to $Re = 5000$, the Mach number is $M = 0.5$ and there is no angle of attack. To validate our



(a) after two adaptations



(b) after two adaptations

Fig. 6. Meshes for the leading edge: left fitted, right penalized. We display the second and third meshes.

approach we have chosen to perform this numerical simulation using first a body-fitted mesh along with mesh adaption on u-velocity. Then our IBM-LS-AUM approach is performed on an embedded grid along with mesh adaptation performed on both u-velocity and level-set. The body-fitted and embedded initial grids are presented in Fig. 13, the body-fitted grid contains 44 829 vertices and 87 823 triangles, the embedded grid contains 49 565 vertices and 99 062 triangles. As usual the embedded grid does not contain necessarily points on the zero level. After 3 cycles of mesh adaptation, the two solutions are compared in Fig. 14 where for each sub-figure the body-fitted solution is located on top and IBM-LS-AUM solution on bottom. The adaptation parameters on the u-velocity field for both approaches are taken the same, that is

$$\varepsilon = 5 \cdot 10^{-4}; \qquad h_{\min} = 10^{-4}; \qquad h_{\max} = 2$$

and for the embedded grid we use for the level-set adaptation criteria the following parameters

$$\varepsilon = 10^{-4}; \qquad h_{\min} = 5 \cdot 10^{-4}; \qquad h_{\max} = 2.$$

The adapted meshes contain respectively 84 880 vertices and 165 948 triangles for the body-fitted grid and 101 478 vertices with 202 885 triangles for the penalized grid, meaning that the penalized case adds less than 20% of nodes. The u-velocity component drawn in Fig. 14(a), with a zoom at leading edge (Fig. 14(b)) and at trailing edge Fig. 14(c) shows a perfect agreement of both approaches, validating the overall proposed technique. Special care has to be given to the trailing edge of the NACA 0012 airfoil because this is a singularity for the level-set, indeed the angle is very narrow, the level-set distance should be verify before performing the IBM-LS-AUM technique to avoid convexity default.

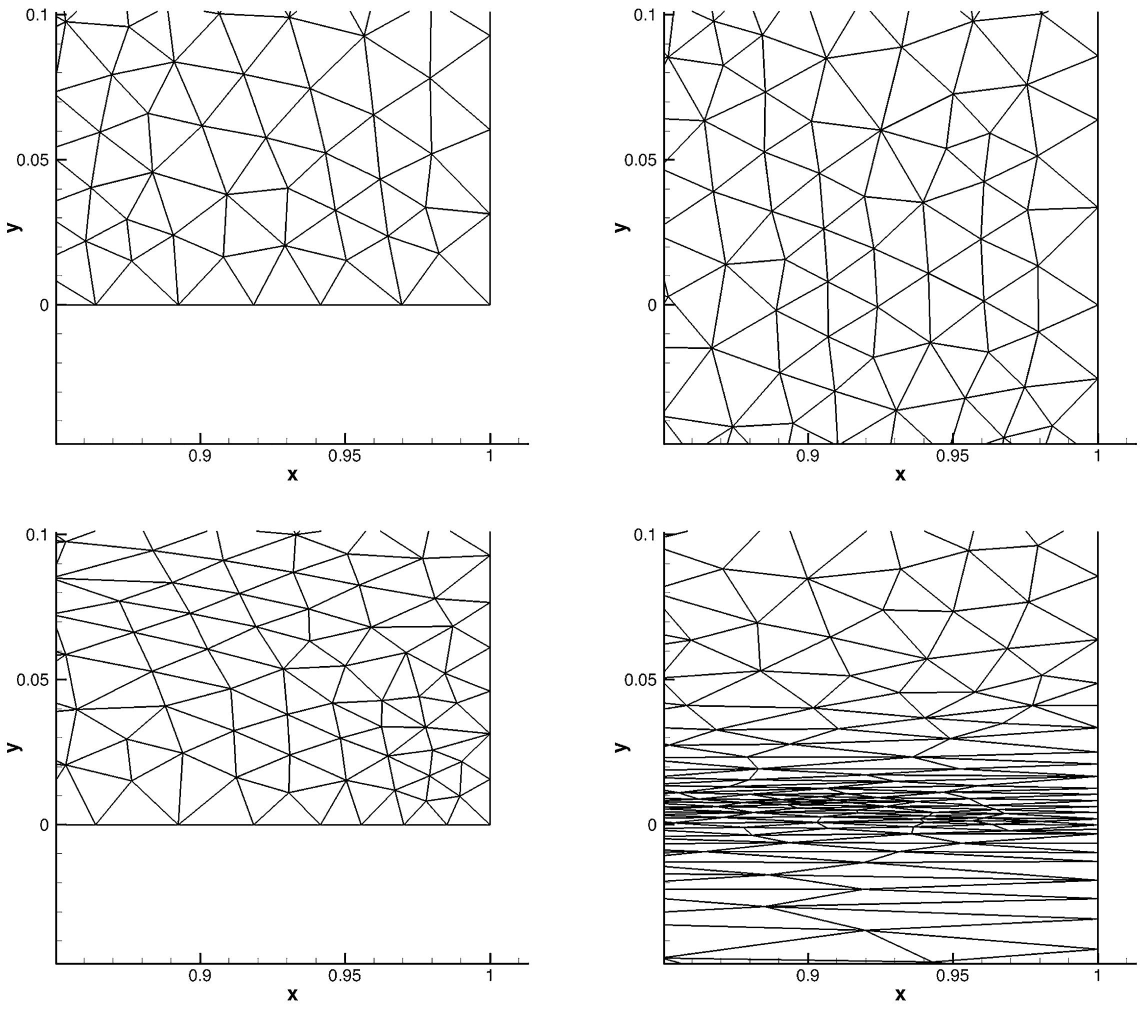


Fig. 7. Meshes for the trailing edge: left fitted, right penalized. We display the initial meshes and after one adaptation.

4.5. 3D test case flow around an ellipse

The method described on this paper can easily be applied on 3D test cases. For this test case we use the mesh generator MMG3D [13,12]. We show in this section, a preliminary test case in three dimensions, a laminar flow around an ellipse. The ellipse is centered at $(0,0,0)$, the axes are aligned with x, y, z and the radius are respectively $(0.5, 0.1, 0.2)$. The complete domain is a sphere of radius 15. The initial mesh is adapted on the isovalue 0 of the level-set function. To obtain the initial mesh we impose

$$\varepsilon = 10^{-4}; \qquad h_{\min} = 5 \cdot 10^{-3}; \qquad h_{\max} = 2.$$

The initial mesh contains 1 051 541 tetrahedrons and 178 909 vertices, see Fig. 15(a). The flow is then computed using the penalized Navier–Stokes equations (2). The Reynolds number is fixed to $Re = 500$ and the Mach number to $M = 0.375$. Once the first solution is computed, we adapt the mesh on two criteria, the x-component of the velocity and the zero isovalue of the level-set. The adaptation parameters for the velocity are

$$\varepsilon = 10^{-3}; \qquad h_{\min} = 5 \cdot 10^{-3}; \qquad h_{\max} = 2$$

and we keep the parameters of the initial mesh for the level-set adaptation. The adapted mesh is shown on Fig. 15(b). It contains 1 117 932 tetrahedrons and 187 977 vertices. Figs. 15(c) and 15(d) show respectively the distribution of x- and y-components of the velocity.

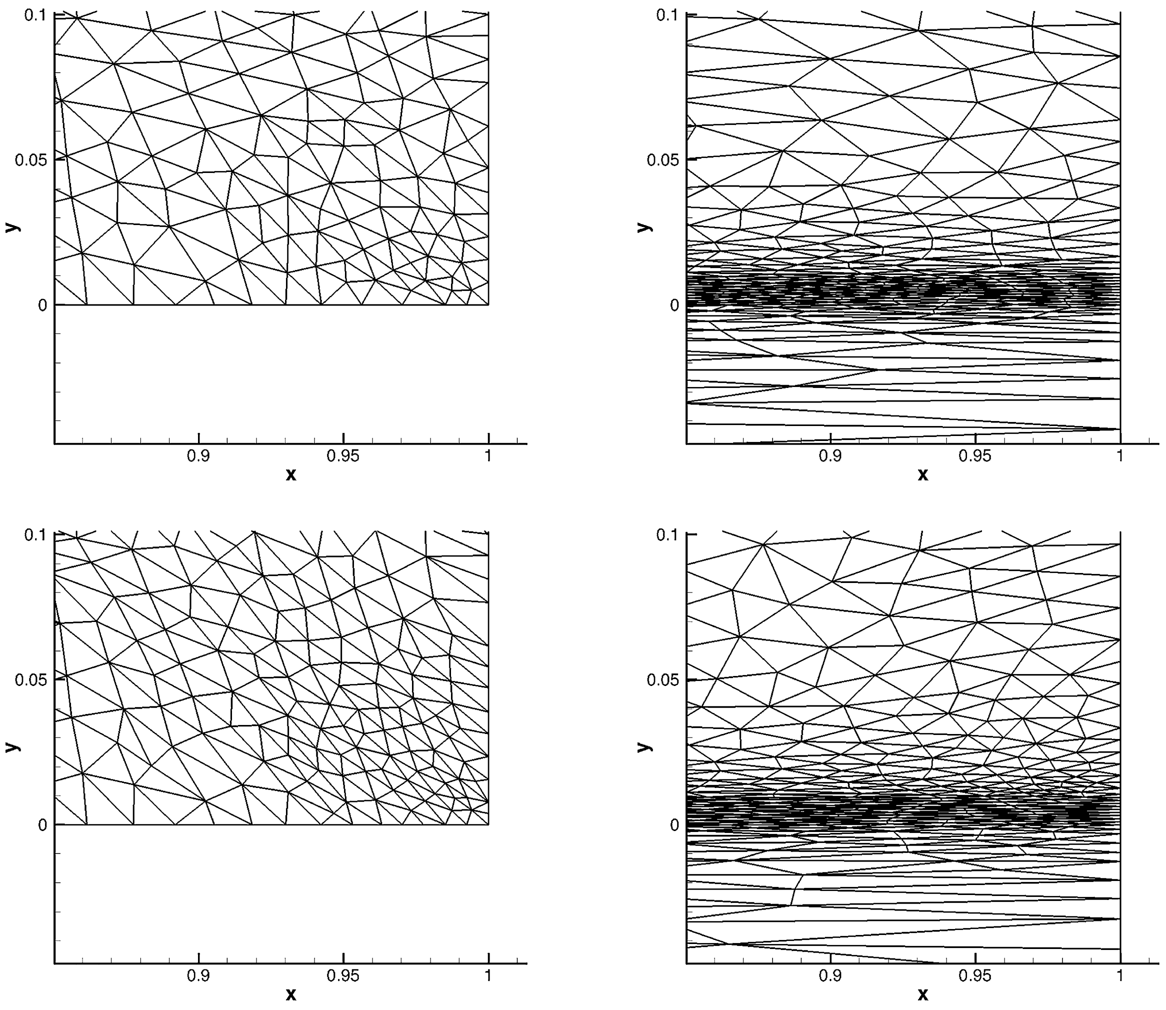


Fig. 8. Meshes for the trailing edge: left fitted, right penalized. We display the second and third meshes.

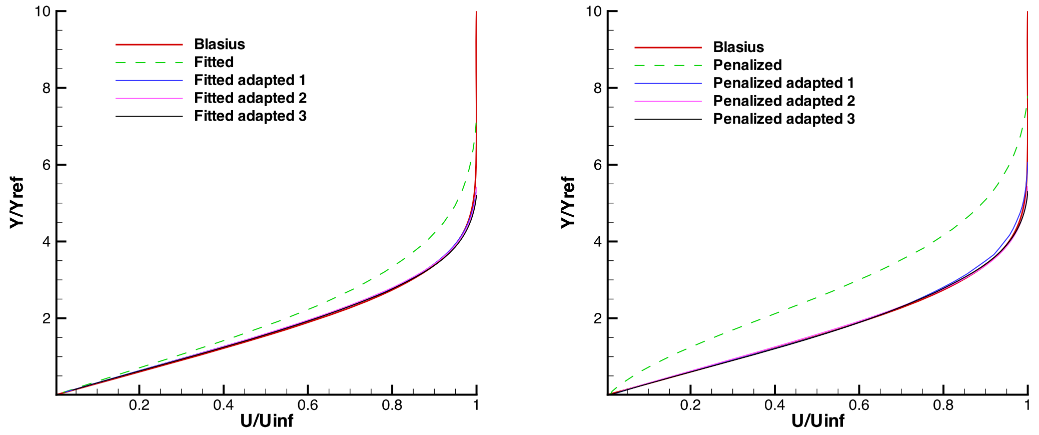
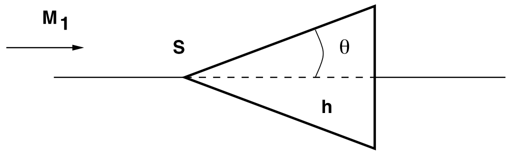
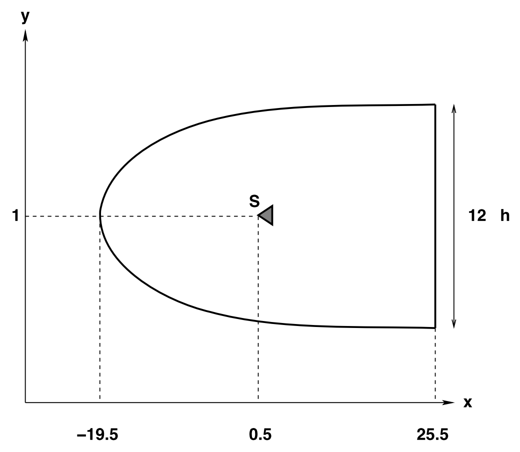


Fig. 9. Velocity distribution for the body-fitted (left) and penalized (right) methods. We display the solutions for the four meshes.

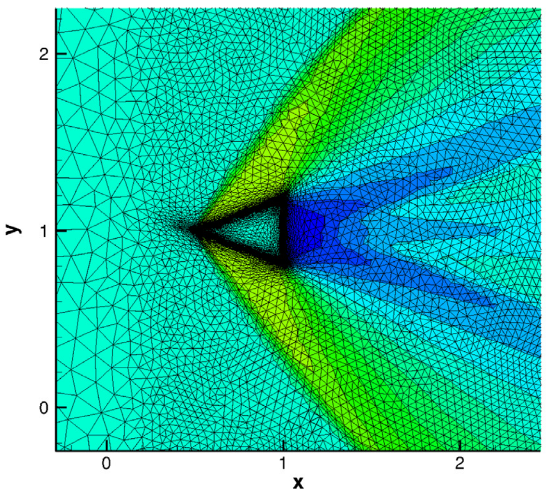


(a) Triangle geometry

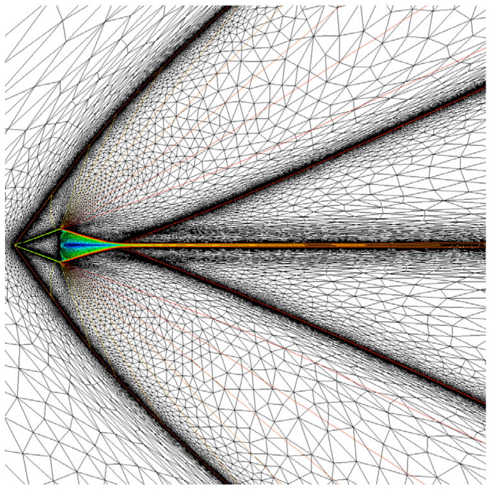


(b) Triangle domain

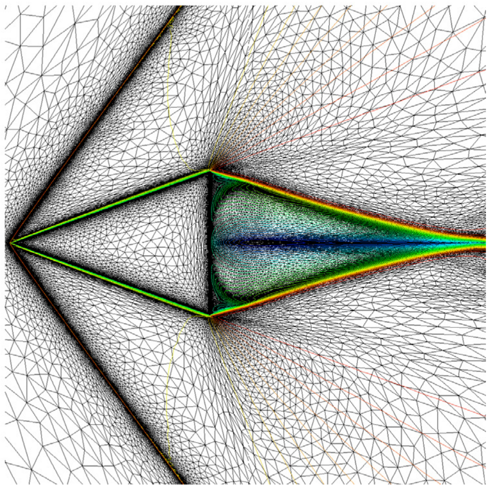
Fig. 10. Triangle test case setup.



(a) Initial Mesh and ρ distribution



(b) Adapted Mesh along with u-velocity



(c) zoom on the Adapted Mesh

Fig. 11. 2D triangle test case, initial mesh and solution, final mesh and solution.

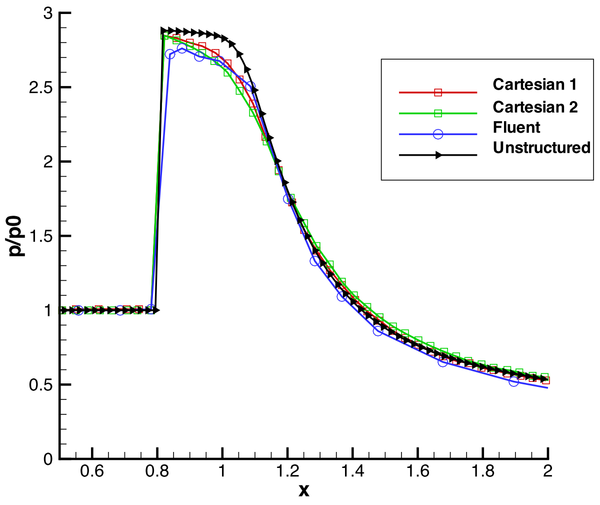


Fig. 12. 1D cut of the pressure along the line $y = 1.44$; red squares Boiron Cartesian grid 1024×1024; green squares Boiron Cartesian grid 512×512; blue circles Boiron fluent solution; black triangles our approach. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

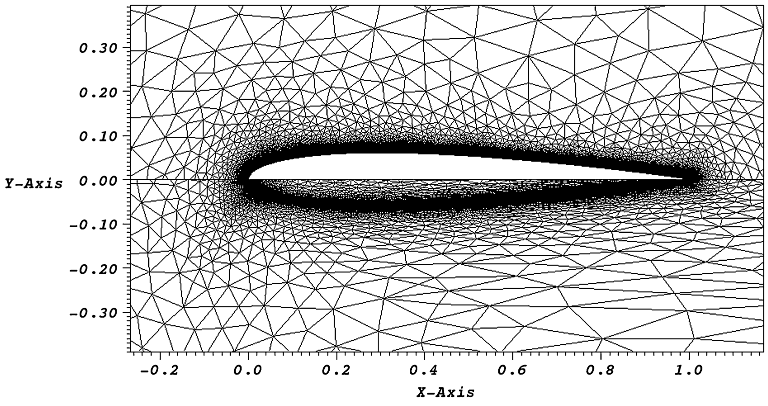


Fig. 13. Initial meshes body-fitted and embedded.

5. Conclusion

In this work, we propose to combine IBM-LS-AUM techniques to mesh adaption. The idea is to conserve the simplicity of the embedded approaches for grid generation process and overcome the difficulty of wall treatments by using mesh adaptation. Mesh adaptions are performed using two criteria, the distance to the level-set 0 and one component of the flow solution (ρ, u, v, etc.). In our case u has been chosen. Other parameters could have been chosen. The four test cases proposed demonstrate the ability of the proposed method to obtain an accurate solution along with an accurate wall treatment even when the initial mesh does not contain any point on the level-set 0. The obtained numerical solutions are compared to analytical solution for Blasius test case, to other numerical solutions for supersonic flow around a triangle, to body-fitted adapted solution for a laminar NACA 0012 airfoil, and the technique is performed on a 3D test case to demonstrate the feasibility of the method even in the case of a more demanding problem. The important conclusion is that, for a given accuracy, the number of points added in the boundary layer by this mesh adaptation technique with level set, though larger to what can be done for body-fitted meshes, is still of the same order: the method is about 20% more expensive but completely avoid the construction of meshes having to match complex geometries. In the near future, we will investigate IBM-LS-AUM methods for moving bodies.

Acknowledgements

R. Abgrall has been funded in part by the EU ERC Advanced grant “ADDECCO” #226616. Experiments presented in this paper were carried out using the PLAFRIM experimental testbed, being developed under the Inria PlaFRIM development action with support from LABRI and IMB and other entities: Conseil Régional d’Aquitaine (http://aquitaine.fr/), FeDER

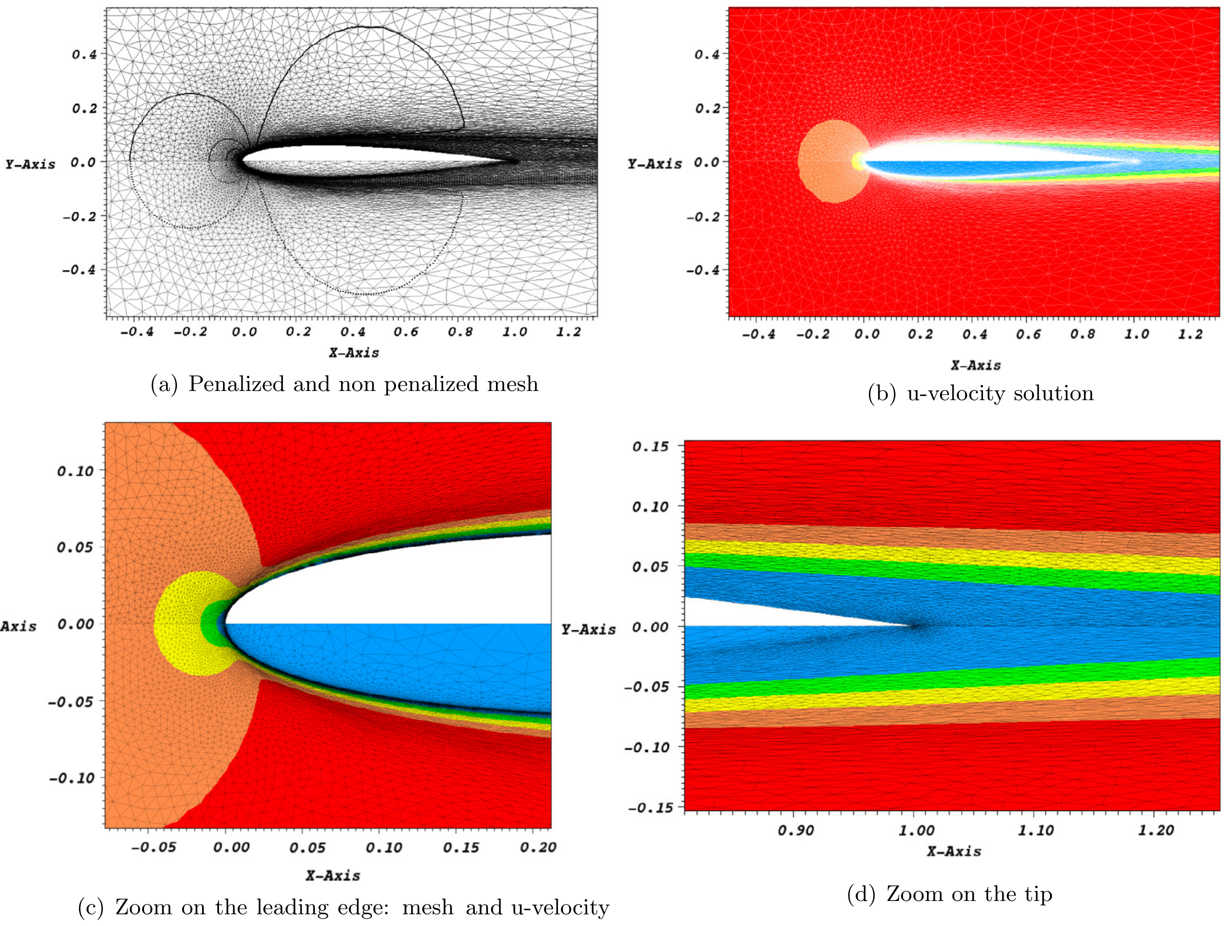


Fig. 14. Laminar flow around a NACA0012 airfoil.

(http://europa.eu/legislation_summaries/employment_and_social_policy/job_creation_measures/l60015_fr.htm), Université de Bordeaux (http://www.univ-bordeaux.fr/) and CNRS (http://www.cnrs.fr/, see https://plafrim.bordeaux.inria.fr/).

Appendix A. Practical issues for the metric construction

In practice we compute the two metrics on each node of the mesh. The algorithms used to compute these metrics are given in the following sections, then we proceed to a metric intersection using a simultaneous reduction.

A.1. Flow features metric

We choose the flow scalar variable on which we decide to adapt the grid, for example the density, and we proceed with the following steps:

1. We compute the gradient on each node using a least square approximation;
2. We compute the Hessian with a least square approximation on the gradient: on each node the Hessian is a $d \times d$ matrix where d is the space dimension;
3. We normalize the Hessian by dividing by the maximum value of the chosen scalar through the domain;
4. We define the metric using Eqs. (11) and (12), i.e. we compute the eigenvalues and eigenvectors of the Hessian.

A.2. Level-set metric

The next step is to compute the level-set metric:

1. On each node we compute the gradient of the level-set function using a least square approximation;
2. We compute the Hessian (using least square);

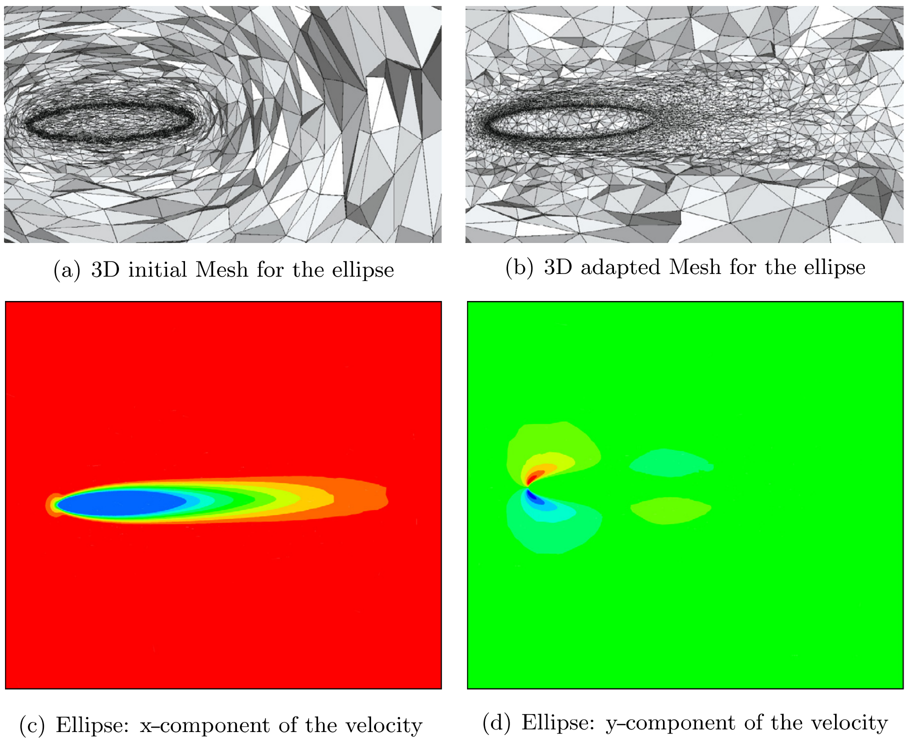

(a) 3D initial Mesh for the ellipse (b) 3D adapted Mesh for the ellipse (c) Ellipse: x–component of the velocity (d) Ellipse: y–component of the velocity

Fig. 15. Cuts through tetrahedral meshes for the 3D ellipse and 2D cuts of the velocity.

3. We check if we are close to the level-set 0 or not:
   3.1. If we are closed to the level-set 0: we compute the curvature of the level-set, using the Hessian, to impose the metric defined by Eq. (10);
   3.2. If not we increase the edge length linearly from $h_{\min}$ to $h_{\max}$ depending on the distance from the level-set 0;
4. We then have our metric for the level-set and we have to intersect it with the previous one.

A.3. Derivation of (2)

Assuming first, Dirichlet boundary conditions on the velocity and the temperature, and using material derivatives, the Navier–Stokes in nonconservative form writes:

$$\begin{aligned}
&\frac{d\rho}{dt}+\rho\frac{\partial}{\partial \boldsymbol{x}}\cdot\boldsymbol{u}=0\\
&\rho\frac{d\boldsymbol{u}}{dt}+\frac{\partial p}{\partial \boldsymbol{x}}=\frac{\partial}{\partial \boldsymbol{x}}\underline{\boldsymbol{\pi}}\\
&\frac{de}{dt}+(p+e)\frac{\partial}{\partial \boldsymbol{x}}\cdot\boldsymbol{u}=\frac{\partial}{\partial \boldsymbol{x}}\cdot(\underline{\boldsymbol{\pi}}\boldsymbol{u}+\boldsymbol{q})-\boldsymbol{u}\cdot\frac{\partial}{\partial \boldsymbol{x}}\underline{\boldsymbol{\pi}}.
\end{aligned}$$

In order to impose $\boldsymbol{u}=\boldsymbol{u}_{s^i}$ and $T=T_{s^i}$, we write

$$\begin{aligned}
&\frac{d\rho}{dt}+\rho\frac{\partial}{\partial \boldsymbol{x}}\cdot\boldsymbol{u}=0\\
&\rho\frac{d\boldsymbol{u}}{dt}+\frac{\partial p}{\partial \boldsymbol{x}}=\frac{\partial}{\partial \boldsymbol{x}}\underline{\boldsymbol{\pi}}+\frac{1}{\eta}\sum_{i=1}^{Ns}\chi_{s^i}(\rho\boldsymbol{u}-\rho\boldsymbol{u}_{s^i})\\
&\frac{de}{dt}+(p+e)\frac{\partial}{\partial \boldsymbol{x}}\cdot\boldsymbol{u}=\frac{\partial}{\partial \boldsymbol{x}}\cdot(\underline{\boldsymbol{\pi}}\boldsymbol{u}+\boldsymbol{q})-\boldsymbol{u}\cdot\frac{\partial}{\partial \boldsymbol{x}}\underline{\boldsymbol{\pi}}+\frac{1}{\eta}\sum_{i=1}^{Ns}\theta_{s^i}\chi_{s^i}\rho\big(\epsilon(T)-\epsilon(T_{s^i})\big)
\end{aligned}$$

with $\theta_{s^i}=1$. If we add the equation on the energy with equation on the momentum dotted against $\boldsymbol{u}$ with $\frac{1}{2}\boldsymbol{u}^2$ the equation on the mass, we get (2). In the case of Neuman boundary conditions, the derivation is similar with $\theta_{s^i}=0$.

The system (2) is Galilean invariant. If we change $\boldsymbol{u}$ in $\boldsymbol{u}+\boldsymbol{u}_0$, the Galilean invariance comes from the combination of the energy equation and the momentum equation that cancels the $\rho\boldsymbol{u}_0(\boldsymbol{u}-\boldsymbol{u}_{s^i})$ term coming out of the energy equation.

The entropy relation is

$$T\frac{ds}{dt}=\frac{d\epsilon}{dt}+p\frac{\partial}{\partial \boldsymbol{x}}\boldsymbol{u}=\frac{1}{\rho}\left(\frac{\partial}{\partial \boldsymbol{x}}\cdot(\underline{\boldsymbol{\pi}}\boldsymbol{u}+\boldsymbol{q})-\boldsymbol{u}\cdot\frac{\partial}{\partial \boldsymbol{x}}\underline{\boldsymbol{\pi}}\right)+\frac{1}{\eta}\sum_{i=1}^{Ns}\theta_{s^i}\,\chi_{s^i}\big(\epsilon(T)-\epsilon(T_{s^i})\big).$$

The last term of this relation has no sign, hence nothing can be said about the entropy dissipation rate.

A way to circumvent this is to modify the term $\sum_{i=1}^{Ns}\theta_{s^i}\,\chi_{s^i}\rho(\epsilon(T)-\epsilon(T_{s^i}))$ into

$$\sum_{i=1}^{Ns}\theta_{s^i}\chi_{s^i}\rho\frac{(\epsilon(T)-\epsilon(T_{s^i}))^2}{\epsilon(T)+\epsilon(T_{s^i})}$$

so that the penalized system becomes

$$\begin{aligned}
&\frac{\partial\rho}{\partial t}+\frac{\partial}{\partial \boldsymbol{x}}\cdot(\rho\boldsymbol{u})=0\\
&\frac{\partial}{\partial t}(\rho\boldsymbol{u})+\frac{\partial}{\partial \boldsymbol{x}}\cdot(\rho\boldsymbol{u}\otimes\boldsymbol{u})+\frac{\partial p}{\partial \boldsymbol{x}}=\frac{1}{\eta}\sum_{i=1}^{Ns}\chi_{s^i}(\rho\boldsymbol{u}-\rho\boldsymbol{u}_{s^i})+\frac{\partial}{\partial \boldsymbol{x}}\cdot\underline{\boldsymbol{\pi}}\\
&\frac{\partial}{\partial t}(\rho e)+\frac{\partial}{\partial \boldsymbol{x}}\cdot\big((\rho e+p)\boldsymbol{u}\big)\quad=\frac{1}{\eta}\sum_{i=1}^{Ns}\theta_{s^i}\,\chi_{s^i}\rho\frac{(\epsilon(T)-\epsilon(T_{s^i}))^2}{\epsilon(T)+\epsilon(T_{s^i})}+\frac{1}{\eta}\sum_{i=1}^{Ns}\chi_{s^i}(\rho\boldsymbol{u}-\rho\boldsymbol{u}_{s^i})\cdot\boldsymbol{u}+\frac{\partial}{\partial \boldsymbol{x}}\cdot(\underline{\boldsymbol{\pi}}\boldsymbol{u}+\boldsymbol{q}).
\end{aligned}\tag{A.1}$$

Note that in (2) or (A.1), η has the dimension of a time.

References

[1] Metis home page, http://glaros.dtc.umn.edu/gkhome/views/metis, 2006.
[2] F. Alauzet, A. Loseille, A. Dervieux, P. Frey, Multi-dimensional continuous metric for mesh adaptation, in: Proc.of 15th Int. Meshing Roundtable, 2006.
[3] P. Angot, C.H. Bruneau, P. Fabrie, A penalization method to take into account obstacles in incompressible viscous flows, Numer. Math. 81 (4) (1999) 497–520.
[4] E.F.D. Azevedo, R.B. Simpson, On optimal triangular meshes for minimizing the gradient error, Numer. Math. 59 (1991) 321–348.
[5] D. Benson, Computational methods in Lagrangian and Eulerian hydrocodes, Comput. Methods Appl. Mech. Eng. 99 (1992) 235–394.
[6] O. Boiron, G. Chiavassa, R. Donat, A high-resolution penalization method for large Mach number flows in the presence of obstacles, Comput. Fluids 38 (3) (2009) 703–714.
[7] F.J. Bossen, P.S. Heckbert, A pliant method for anisotropic mesh generation, in: Proc. of 5th Int. Meshing Roundtable, 1996.
[8] B. Braconnier, B. Nkonga, M. Papin, P. Ramet, M. Ricchiuto, J. Roman, R. Abgrall, Efficient solution technique for low Mach number compressible multiphase problems, in: Proceedings of PMAA 2006, Rennes, September 2006.
[9] C. Bui, C. Dapogny, P. Frey, An accurate anisotropic adaptation method for solving the level set advection equation, Int. J. Numer. Methods Fluids (2011).
[10] J.I. Choi, R.C. Oberoi, J.R. Edwards, J.A. Rosati, An immersed boundary method for complex incompressible flows, J. Comput. Phys. 224 (2007) 757–784.
[11] M. Coquerelle, G.H. Cottet, A vortex level set method for the two-way coupling of an incompressible fluid with colliding rigid bodies, J. Comput. Phys. 227 (21) (2008) 9121–9137.
[12] C. Dobrzynski, P. Frey, MMG3D home page, http://www.math.u-bordeaux1.fr/~cdobrzyn/logiciels/mmg3d.php.
[13] C. Dobrzynski, P. Frey, Anisotropic Delaunay Mesh Adaptation for Unsteady Simulations, in: Proc. of 17th Int. Meshing Roundtable, Pittsburgh, USA, 2008.
[14] V. Dolejsi, Anisotropic mesh adaptation for finite volume and finite element methods on triangular meshes, Comput. Vis. Sci. 1 (1998) 165–178.
[15] J. Dompierre, M.-G. Vallet, Y. Bourgault, M. Fortin, W.G. Habashi, Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent cfd. part iii: unstructured meshes, Int. J. Numer. Methods Fluids 39 (2002) 675–702.
[16] C. Farhat, P. Geuzaine, C. Grandmont, The discrete geometric conservation law and the nonlinear stability of ale schemes for the solution of flow problems on moving grids, J. Comput. Phys. 174 (2001) 669–694.
[17] R.P. Fedkiw, Coupling an eulerian fluid calculation to a lagrangian solid calculation with the ghost fluid method, J. Comput. Phys. 175 (2002) 200–224.
[18] L. Fezoui, B. Stoufflet, A class of implicit upwind schemes for Euler simulations with unstructured meshes, J. Comput. Phys. 84 (1) (1989) 174–206.
[19] L. Formaggia, S. Perotto, New anisotropic a priori error estimates, Numer. Math. 89 (2001) 641–667.
[20] P.J. Frey, F. Alauzet, Anisotropic mesh adaptation for cfd simulations, Comput. Methods Appl. Mech. Eng. 194 (48–49) (2005) 5068–5082.
[21] P.J. Frey, P.-L. George, Mesh Generation. Application to Finite Elements, 2nd edition, Wiley, Paris, 2008.
[22] P.-L. George, H. Borouchaki, P.J. Frey, P. Laug, E. Saltel, Mesh generation and mesh adaptivity: theory, techniques, in: E. Stein, R. de Borst, T.J.R. Hugues (Eds.), Encyclopedia of Computational Mechanics, John Wiley & Sons Ltd., 2004.
[23] P.L. George, F. Hecht, M.G. Vallet, Creation of internal points in Voronoi's type method. Control adaptation, Adv. Eng. Softw. Workstn. 13 (5–6) (1991) 303–312.
[24] R. Ghias Mittal, H. Dong, A sharp interface immersed boundary method for compressible viscous flows, J. Comput. Phys. 225 (2007) 528–553.
[25] A. Glimanov, F. Sotiropoulos, A hybrid Cartesian/immersed boundary method for simulating flows with 3D, geometrically complex, moving bodies, J. Comput. Phys. 207 (2005) 457–492.
[26] C. Gruau, T. Coupez, 3D unstructured ans anisotropic mesh generation with adaptation to natural and multidomain metric, Comput. Methods Appl. Mech. Eng. 194 (48–49) (2005) 4951–4976.
[27] P. Hénon, P. Ramet, J. Roman, PaStiX: a high-performance parallel direct solver for sparse symmetric definite systems, Parallel Comput. 28 (2) (January 2002) 301–321.
[28] X.Y. Hu, B.C. Khoo, N.A. Adams, F.L. Huang, A conservative interface method for compressible flows, J. Comput. Phys. 219 (2006) 553–578.
[29] D. Isola, A. Guardone, G. Quaranta, Arbitrary lagrangian eulerian formulation for two-dimensional flows using dynamic meshes with edge swapping, J. Comput. Phys. 230 (2011) 7706–7722.

[30] G. Kalitzin, G. Iaccarino, Turbulence modeling in an immersed-boundary RANS method, Annual Research Briefs, Center for Turbulence Research, 2002.
[31] C.H. Lepage, W.G. Habashi, Conservative interpolation of aerodynamic loads for aeroelastic computations. AIAA-Paper 2000-1449, 2000.
[32] Claude Y. Lepage, Wagdi G. Habashi, Fluid–structure interactions using the ale formulation, AIAA-Paper 99-0660, 1999.
[33] R. Lohner, J.D. Baum, E.L. Mestreau, D. Sharov, C. Charman, D. Pelessone, Adaptative embbeded unstructured grid methods, AIAA-Paper 03-1116, 2003.
[34] A. Mark, B.G.M. van Wachem, Derivation and validation of a novel implicit second-order accurate immersed boundary method, J. Comput. Phys. 227 (13) (2008) 6660–6680.
[35] R. Mittal, G. Iaccarino, Immersed boundary methods, Annu. Rev. Fluid Mech. 37 (1) (2005) 239–261.
[36] R. Mittal, V. Seshadri, H.S. Udaykumar, Flutter, tumble and vortex induced autorotation, Theor. Comput. Fluid Dyn. 17 (2004) 165–170.
[37] B. Nkonga, H. Guillard, Godunov type method on non-structured meshes for three-dimensional moving boundary problems, Comput. Methods Appl. Mech. Eng. 113 (1–2) (1994) 183–204.
[38] F. Pellegrini, Scotch home page, http://www.labri.fr/perso/~pelegrin/scotch/, 2008.
[39] Alexandre M. Roma, Charles S. Peskin, Marsha J. Berger, An adaptive version of the immersed boundary method, J. Comput. Phys. 153 (2) (1999) 509–534.
[40] J.A. Sethian, Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Cambridge University Press, 1996.
[41] S. Tan, C. Wang, C.-W. Shu, J. Ning, Efficient implementation of high order inverse Lax–Wendroff boundary treatment for conservation laws, J. Comput. Phys. 231 (6) (2012) 2510–2527.
[42] K. Wang, A. Rallu, J.-F. Gerbeau, C. Farhat, Algorithms for interface treatment and load computation in embedded boundary methods for fluids and fluid–structure interaction problems, Int. J. Numer. Methods Fluids 67 (2011) 1175–1206.