$h-$ and $r-$ adaptation on simplicial meshes using MMG tools

Luca Arpaia, Héloïse Beaugendre, Luca Cirrottola, Algiane Froehly, Marco Lorini, Léo Nouveau and Mario Ricchiuto

Abstract We review some recent work on the enhancement and application of both $r-$ and $h-$ adaptation techniques, benefitting of the functionalities of the remeshing platform Mmg: `www.mmgtools.org`. Several contributions revolve around the level-set adaptation capabilities of the platform. These have been used to identify complex surfaces and then to either produce conformal 3D meshes, or to define a metric allowing to perform h-adaptation and control geometrical errors in the context of immersed boundary flow simulations. The performance of the recent distributed memory parallel implementation ParMmg is also discussed. In a similar spirit, we propose some improvements of $r-$adaptation methods to handle embedded fronts, and in domains with curved conformal boundaries.

Acknowledgements Dedicated to our dear friend Cécile Dobrzynski.

1 Introduction

Mesh adaptation has become nowadays a powerful tool to improve the discrete representation of complex solution fields in many applications, and particularly in computational fluid dynamics [51]. Adapting the mesh may lead to a non-negligible

Luca Arpaia
Coastal Risk and Climate Change Unit, French Geological Survey, 3 Av. C. Guillermin 45060 Orléans Cedex 2, France, e-mail: l.arpaia@brgm.fr

Héloïse Beaugendre, Luca Cirrottola, Algiane Froehly, Marco Lorini, and Mario Ricchiuto
INRIA, Université de Bordeaux, CNRS, Bordeaux INP, IMB UMR 5251, 200 Avenue de la Vieille Tour, 33405 Talence cedex, France,
e-mail: heloise.beaugendre/luca.cirrottola/algiane.froehly/marco.lorini/mario.ricchiuto@inria.fr

Léo Nouveau
Univ Rennes, INSA Rennes, CNRS, IRMAR - UMR 6625, F-35000 Rennes, France,
e-mail: leo.nouveau@insa-rennes.fr

computational overhead, as well as to more complex algorithmic and software developments. This motivates the quest for efficient and robust methods.

h-adaptation allows to optimize the discrete representation of a field of interest by inserting and removing mesh entities. This has proven to be a very powerful tool, and it is today quite mature and generic (cf. [51] and references therein). Several aspects of this approach are still quite complex in general. This is the case for conservative high order solution transfer between meshes with different topologies, which is a non-trivial step with a non-negligible cost [27, 6, 33, 40, 52]. *h*-adaptation also entails non foreseeable, non-linear, and dynamic changes in the distribution of the computational workload, which is an inherently weakly parallelizable feature on distributed memory architectures.

Conversely, node relocation without topology change (or $r-$adaptation) has been the first adaptation approach for structured grids, later extended to unstructured ones. It remains still alluring due to the possibility of a minimally intrusive coupling with existing computational solvers, with no modification of the data structures. Moreover, devising conservative projections is quite natural with $r-$adaptation, due to the inherently continuous nature of the process. This allows to exploit space-time or Arbitrary-Lagrangian-Eulerian techniques to build high order conservative remapping [41, 60, 58] compliant with the Geometric Conservation Law (GCL).

The results of this paper benefit of the functionalities of the Mmg platform [1] which implements some well known research on metric-based mesh adaptation [20, 25] in a free and open source software application and library for simplicial mesh adaptation [2]. The platform provides adaptive and implicit domain remeshing through three libraries and applications `Mmg2d`, `Mmgs` and `Mmg3d`, targeting two-dimensional cartesian and surfacic triangulations, and three-dimensional tetrahedral meshes. Recent efforts are devoted to the development of its parallel version `ParMmg` [18, 4] on top of the `Mmg3d` remesher. A moving mesh library `Fmg`, which uses the `Mmg` library for its mesh data structure and geometry representation, is under development. Example of applications in various fields and documentation can be found on the platform website [1], also providing tutorials under the section *Try Mmg!*, and library examples in the respective repositories [2, 4].

The contribution here discussed exploit the functionalities of the platform. We present the work done in the context of the application of *h*-adaptation to control geometrical errors in immersed boundary flow simulations, and we discuss the performance of the parallel implementation in ParMmg. In a similar spirit, we review some improvements of $r-$adaptation methods in domains with both fitted and immersed boundaries. For immersed boundaries we discuss some ideas used to track embedded fronts and level sets. A nonlinear iterative projection method allowing tangling free adaptation of moving fronts in 3D domains with curved manifold boundaries is then presented.

The paper is organized in two main parts. Sect. 2 is devoted to the application and implementation of *h*-adaptation methods, with focus on level-set adaptation, application to immersed boundary methods, and parallel *h*-adaptation. Sect. 3 presents

instead advances and applications of $r-$adaptation methods with focus on the resolution of embedded fronts, and on more general boundary conditions for the moving mesh. The paper is ended by an overlook on ongoing activities.

2 $h-$adaptation: embedded geometries, parallel implementation

Anisotropic $h-$adaptation has largely proved useful in the context of numerical simulations to capture the physical behavior of complex phenomena at a reasonable computational cost [30]. Following a now classical approach, we consider adaptation strategies based on a local error estimation, which is converted in a map for the size and orientation of the mesh edges. This is done via an iterative process based on successive flow field evaluations, error estimations, and a-posteriori local mesh modifications. As described in [31], mesh sizes and edge directions are controlled via metric tensors built starting from error estimations involving the Hessian of the target output. The eigenvalues λ_i of these matrices are directly linked to the sizes h_i of the elements edges in the directions i (where $\lambda_i = 1/h_i^2$), with these directions given by the eigenvectors. Different criteria can be used to evaluate metrics, two examples relevant for the applications discussed later are recalled hereafter.

If several metric fields are available, e.g. the ones of Sects. 2.1 and 2.2, one may seek to combine them. To this end, we use here the simultaneous reduction method described in [31]. Denoting by $\mathcal{M}_1$ and $\mathcal{M}_2$ the two metrics defined at the same node, the resulting metric $\mathcal{M}_{1\cap 2}$ respects both sizes prescribed by the two initial metrics. With a geometrical analogy, the ellipsoid $\mathcal{E}_{1\cap 2}$ associated to $\mathcal{M}_{1\cap 2}$ is the biggest ellipsoid included in both $\mathcal{E}_1$ and $\mathcal{E}_2$, associated to $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively.

2.1 Physical adaptation

If we aim at controlling the error between a certain output field, u, and its linear interpolation, we can exploit the upper bound on a mesh element K [30]:

$$||u - \Pi_h u||_{\infty,K} \leq C_d \max_{\mathbf{e}\in K} \langle \mathbf{e}, \mathcal{M}_1(K)\mathbf{e}\rangle \ . \tag{1}$$

Here, $\mathbf{e}$ denotes the edges of the mesh element and C_d is a constant depending on the dimension. The metric $\mathcal{M}_1(K)$ is a function of the Hessian of u, $\mathcal{H}(u)$:

$$\mathcal{M}_1 = {}^t\mathcal{R}\begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix}\mathcal{R}\,, \tag{2}$$

with $\mathcal{R}$ the matrix of the eigenvectors of $\mathcal{H}(u)$ and with λ_i defined as:

$$\lambda_i = \min\left(\max\left(|h_i|, \frac{1}{h_{max}^2}\right), \frac{1}{h_{min}^2}\right), \tag{3}$$

where h_i is the i-th eigenvalue of the Hessian, h_{min} (resp. h_{max}) are the minimum (resp. maximum) allowed sizes for the mesh edges.

2.2 Level-set adaptation

The principle is here to improve the accuracy in the definition of the zero iso-value of a level-set function, Φ, describing the distance from a given surface. This allows to adapt the volume mesh on which the function is defined. To this end, we can use the metric field proposed in [21], and given by:

$$\mathcal{M}_2 = {}^t\mathcal{R}\begin{pmatrix} \frac{1}{\epsilon^2} & 0 & 0 \\ 0 & \frac{|\lambda_1|}{\epsilon} & 0 \\ 0 & 0 & \frac{|\lambda_2|}{\epsilon} \end{pmatrix}\mathcal{R}, \tag{4}$$

where $\mathcal{R} = (\nabla\Phi \;\; v_1 \;\; v_2)$, with $(v_1, \; v_2)$ a basis of the tangent plane to the local iso-value surface of Φ, λ_i the eigenvalues of its Hessian and ϵ a user-defined target approximation error. As proposed in the above reference, this metric field can be imposed in a user-defined layer of thickness w, close to the zero iso-level. Outside this region, the mesh size is set to grow linearly up to h_{max}, unless other constraints are set.

2.3 An example of implicit domain meshing: La Sagrada Familia

Level-set methods represent the boundary of a closed body as a level-set (typically the zero level-set) of a continuous function. This point of view can be particularly useful when trying to generate the volume mesh of a body starting only from an approximate representation of its boundary, for example a surface triangulation from an STL file, not necessarily displaying the sufficient regularity necessary for volume mesh generation (for example, being intersecting or non-orientable). In this case, this starting triangulation can be embedded in a larger volume mesh and a signed distance function from the surface triangulation can be defined on mesh nodes. The mesh is adapted according to the metrics defined in the previous section. The procedure can be iterated to increase the geometrical accuracy of the model. A new surface triangulation can also be generated in correspondence of the zero level set, allowing to extract the interior body volume mesh.

An application example is given in Fig. 1, where the implicit domain meshing procedure is applied to the Sagrada Familia starting from the STL files provided by

the International Meshing Roundtable for the meshing contest of the 2017 edition[1], using `Mmg` [20, 2] for the implicit domain remeshing and `Mshdist` [21, 3] for the signed distance function computation. Four isotropic remeshing iterations are performed. The initial mesh has 9 million nodes and 51 million tetrahedra, and the final one has 26 million nodes and 153 million tetrahedra (the worst tetrahedron has quality[2] equal to 0.2, while 99% of the tetrahedra have quality above 0.5, and 95% of the edges have unit-lengths between 0.7 and 1.4).

2.4 Level-sets and IBM with $h-$adaptation

Unfitted discretizations are becoming quite popular for the additional geometrical flexibility they offer. In these methods, the accuracy with which boundary conditions are met is directly linked to the accuracy of the implicit description of the solid [45]. Mesh adaptation with respect to the level-set is a useful tool to control this error. We exploit this in the context of a Brinkman-type volumic penalization discretization of the Navier-Stokes equations [37]. This method falls in the category of continuous forcing immersed boundary methods, meaning that the flow equations are solved throughout the whole domain and the enforcement of rigid motion inside the immersed object is done via a continuous volumic forcing term. In the case of the Navier-Stokes equations, this leads to a compact form

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathbf{F}_C(\mathbf{u}) + \nabla \cdot \mathbf{F}_V(\mathbf{u}, \nabla \mathbf{u}) + \mathbf{p}(\chi, \mathbf{u}) = \mathbf{0} \ , \tag{5}$$

where $\mathbf{u} \in \mathbb{R}^m$ denotes the vector of the m conservative variables, $\mathbf{p} \in \mathbb{R}^m$ the penalization term, d the space dimension, $\mathbf{F}_C, \mathbf{F}_V \in \mathbb{R}^m \otimes \mathbb{R}^d$ the inviscid and viscous flux functions. The localization of the immersed object within the computational Navier-Stokes domain is done through a mask function, χ, which is the Heaviside function of the level-set representation of the immersed solid.

We present in the following $h-$adaptation coupled to a discontinuous Galerkin implementation [44] of the immersed boundary method of Eq. 5 for the steady simulation of a laminar flow at high angle of attack around a delta wing with a sharp leading edge. This is a benchmark test case for adaptive methods for vortex dominated external flows. Even if the geometry is not complex as the one in Sect. 2.3,

[1] https://imr.sandia.gov/26imr/MeshingContest.html

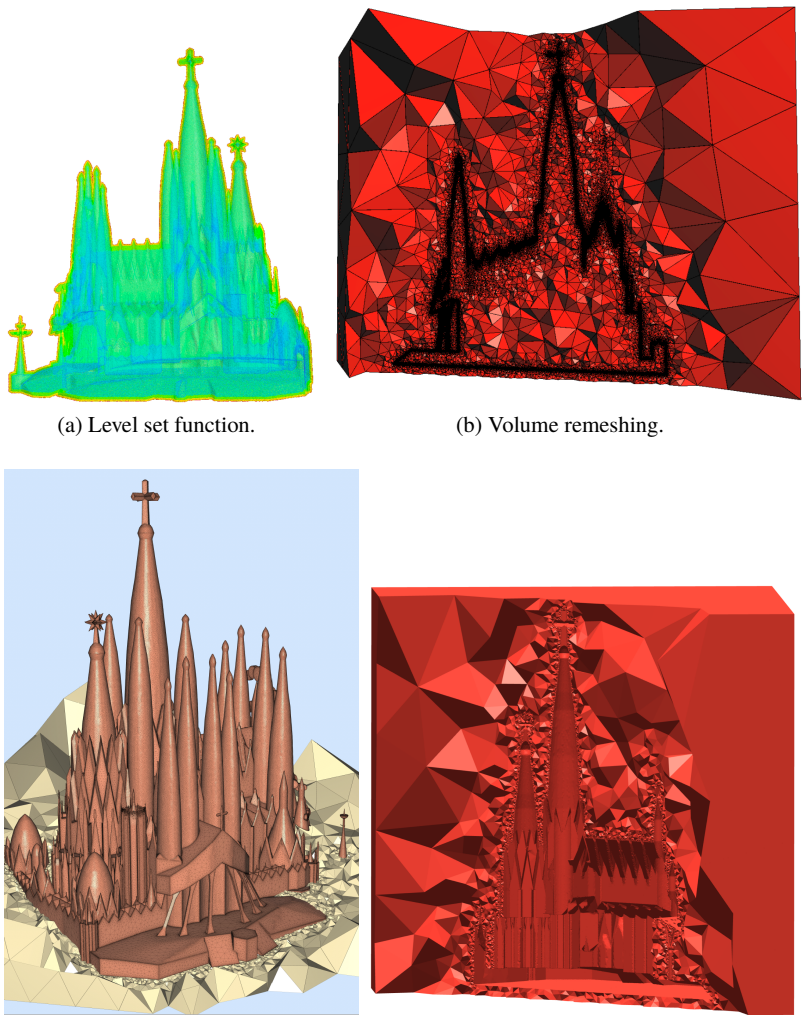
[2] The isotropic length of an edge AB is defined as

$$l_{AB} = \frac{||AB||}{h_A - h_B} \log \frac{h_B}{h_A}$$

while the isotropic tetrahedron quality is defined as

$$Q = \alpha \frac{V}{(\sum_{i=1}^{6} l_j^2)^{1/3}}$$

with α a normalization factor to get quality equal to 1 for the regular tetrahedron with unit edges.



(a) Level set function.

(b) Volume remeshing.

(c) Surface discretization.

(d) Detail of the volume remeshing.

Fig. 1: Implicit domain remeshing and surface discretization of the Sagrada Familia building with `Mmg3d`. From 50 to 150 million tetrahedra on a desktop computer, with 4 remeshing iterations.

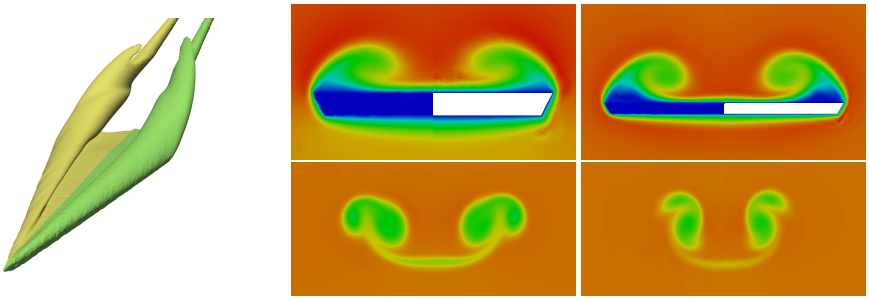


Fig. 2: Laminar delta wing. **left** Roll-up of primary and secondary vortices. **right** $\mathbb{P}^2$ Mach number distribution at $x = 0.5C$, $x = C$, $x = 1.5C$ and $x = 2C$. In each picture on the left half the unfitted computation, on the right the body-fitted one.

$h-$adaptation can be important for the imposition of the immersed boundary condition in order to correctly reproduce the flow topology. See for an example Fig. 2, where the primary and secondary vortices and the DG $\mathbb{P}^2$ Mach number distribution at $x = 0.5C$, $x = C$, $x = 1.5C$ and $x = 2C$ (C being the chord of the wing) are compared between the immersed boundary (left half) and a body-fitted simulation (right half). The mesh for the unfitted simulation consists of 446294 tetrahedra and is adapted to both the Mach number distribution and the level-set (cf. Sects. 2.1 and 2.2).

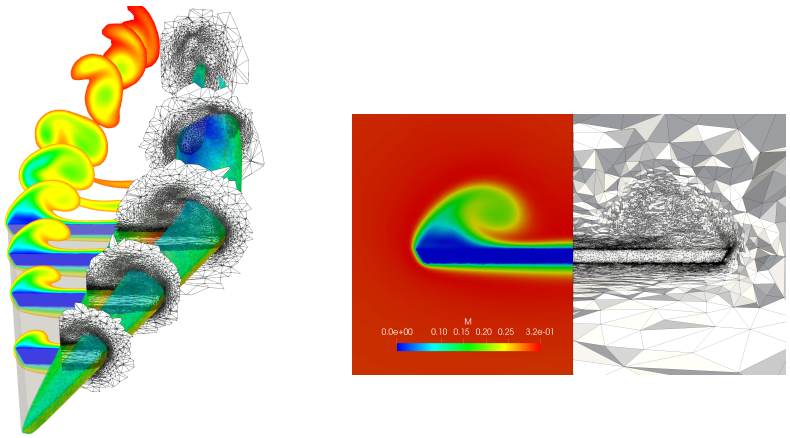


Fig. 3: Laminar delta wing. **left** Slices of the Mach number and adapted mesh. **right** Comparison flow - mesh at the trailing edge.

Then, as a proof of concept, starting from the DG $\mathbb{P}^2$ simulation on the adapted mesh, the full wing mesh has been adapted to fine level of detail (28 milions tetrahedra) with respect to a metric that takes into account both the level-set and the

Table 1: Laminar delta wing, mesh statistics for $h-$adaptation from Original to Final mesh. Percentage of edges $N^{(a,b]}$ whose length in the assigned metrics falls in the interval $l_{\mathcal{M}} \in (a,b]$.

Mesh (# tets)	$N^{(0,0.3]}$	$N^{(0.3,0.6]}$	$N^{(0.6,0.7]}$	$N^{(0.71,0.9]}$	$N^{(0.9,1.3]}$	$N^{(1.3,1.41]}$	$N^{(1.41,2]}$	$N^{>2}$
O (446294)	0.93%	2.81%	1.20%	2.01%	2.99%	0.64%	3.19%	86.23%
F (28083948)	0.01%	0.58%	3.20%	24.36%	63.18%	6.22%	2.45%	0%

computed Mach number distribution. In Fig. 3(left) the left half of the wing presents some slices of the Mach number distribution ($\mathbb{P}^2$ approximation) at different wing locations and in the wake. The surface of the wing is represented only as a reference for the reader. The right half of the wing presents a Mach number isosurface showing the correct representation of the flow topology (with the primary and secondary vortices) and some slices of the adapted mesh. Fig. 3(right) shows a slice of the adapted mesh at the trailing edge of the wing compared with the DG solution. The differences between the imposed metrics, that closely follow the Mach distribution and the level-set description of the wing, can be clearly seen in the different regions of the domain. Finally, in Table 1 we present the mesh statistics in terms of quality of the edges with respect to the imposed metrics from the original to the final mesh.

2.5 Parallel $h-$adaptation

Modern computational mechanics solvers ordinarily exploit parallel, distributed memory computer architectures. This raises the need for generating and adapting meshes whose size, in terms of computer memory, is larger and larger. Even if a parallel solver could in principle use a sequential remesher by gathering the distributed mesh on a single process, it is possible that a large distributed mesh cannot be gathered on a single computing node due to its memory limitation. Beside this primary feasibility consideration, sequential remeshing in a parallel simulation also represents a significant performance bottleneck [50]. Parallel remeshing is thus becoming increasingly necessary in large scale simulations.

The `ParMmg` application and library [4] is built on top of the `Mmg3d` remesher to provide parallel tetrahedral mesh adaptation in a free and open source software. Among the many possible remeshing parallelization methods (for example [13] [23] [49] [17] [28] [14] [24] [10]), a modular approach is adopted by selecting an iterative remeshing-repartitioning scheme that does not modify the adaptation kernel [10]. As described in depth in [18], the sequential remeshing kernel is applied at each iteration on the interior partition of each process while maintaining fixed (non-adapted) parallel interfaces. Then the adapted mesh is repartitioned in order to move the non-adapted frontiers to the interior of the partitions at the next iteration, thus eliminating the presence of non-adapted zones as the iterations progress.

A weak scaling test is shown in Table 2. A sphere of radius 10 is refined while keeping the workload of each process as constant as possible as the number of processes is increased. The test is performed on the `bora` nodes of the `PlaFRIM` cluster[3]. The weak scaling performances are shown on the left in Fig. 4. The slow increase in the time spent in the repartitioning and redistribution phase still leaves space for some implementation optimization.

A strong scaling test is performed with an isotropic sizemap h describing a double Archimedean spiral

$$h(x, y) = \min(1.6 + |\rho - a\theta_1| + 0.005, 1.6 + |\rho + a\theta_2| + 0.0125) \tag{6}$$

with

$$\begin{aligned} \theta_1 &= \phi + \pi \left(1 + \text{floor}\left(\tfrac{\rho}{2\pi a}\right)\right) \\ \theta_2 &= \phi - \pi \left(1 + \text{floor}\left(\tfrac{\rho}{2\pi a}\right)\right) \end{aligned} \tag{7}$$

and $\phi = \text{atan2}(y, x)$, $\rho = s\sqrt{x^2 + y^2}$, $a = 0.6$, $s = 0.5$, into a sphere of radius 10 with uniform unit edge length. The surfacic adaptation and a volumic cut of the adapted meshes on 1 and 1024 processes are shown in Fig. 5. The resulting edge length statistics are shown in Table 3. The scaling performances in log scale are shown on the right in Fig. 4. The speedup S_p over p processes is defined as the ratio between the wall time on 1 process and the wall time on p processes

$$S_p = T_1/T_p \tag{8}$$

except for the speedup of the redistribution part of the program, which is defined with respect to the redistribution time on 2 processes instead of 1 (as there is no redistribution on 1 process). As the number of interfaces increases with the number of processes, it can be noticed both in Table 3 and in Fig. 4 that there is still a trend for a more pronounced propagation of large edge sizes from the fixed interfaces as the number of processes increase, as well as a performance reduction in the redistribution phase. These issues are the focus of ongoing work. Both the weak and strong scaling tests have been performed with the release v1.3.0 of `ParMmg`[4].

3 $r-$adaptation: embedded geometries and curved boundaries

$r-$adaptation techniques are an appealing approach for unsteady simulations of sharp moving fronts. Compared to $h-$refinement, one of their attractive characteristics is the relative algorithmic simplicity, and the ease of defining conservative remaps due to the inherent continuous nature of the process.

Complete reviews of $r-$adaptation can be found for example in [39, 57, 11, 63]. In these methods mesh movement is based on parabolic or elliptic differential maps, often referred to as mesh PDEs. Several interesting formulations allowing PDE based mesh movement have also been proposed for fluid structure interaction where for-

[3] `www.plafrim.fr/`

Table 2: Mesh statistics for the `ParMmg` weak scaling test. Number of vertices n_v and tetrahedra n_e in input and output using p processors.

p	n_v^{in}/p	n_v^{out}/p	n_v^{out}/n_v^{in}	n_v^{out}	n_e^{in}/p	n_e^{out}/p	n_e^{out}/n_e^{in}	n_e^{out}
2	3625	1293637	356.81	2587274	18876	7780974	412.21	15561948
4	3467	1341637	386.88	5366549	18798	8072081	429.39	32288325
8	3346	1380055	412.38	11040444	18666	8306084	444.98	66448675
16	3264	1412516	432.66	22600269	18599	8503695	457.2	136059129
32	3190	1437267	450.46	45992552	18557	8654569	466.36	276946210
64	3214	1431186	445.2	91595935	18625	8619098	462.75	551622317
128	3215	1444674	449.31	184918370	18878	8701524	460.91	1113795077
256	3345	1468905	439.01	376039759	19705	8848464	449.03	2265206835
512	3375	1446532	428.52	740624790	19998	8714450	435.74	4461798709
1024	3335	1449215	434.54	1483996788	19821	8731162	440.49	8940710661

Table 3: Mesh statistics for the `ParMmg` strong scaling test. Percentage of edges $N^{(a,b]}$ whose length in the assigned metrics falls in the interval $l_{\mathcal{M}} \in (a,b]$, for each simulation on p processors.

p	$N^{(0,0.3]}$	$N^{(0.3,0.6]}$	$N^{(0.6,0.7]}$	$N^{(0.71,0.9]}$	$N^{(0.9,1.3]}$	$N^{(1.3,1.41]}$	$N^{(1.41,2]}$	$N^{(2,5]}$	$N^{>5}$
1	1.34 %	35.34 %	16.89 %	25.49 %	20.15 %	0.63 %	0.16 %	0	0
2	1.14 %	34.87 %	16.97 %	25.79 %	20.45 %	0.63 %	0.15 %	0	0
4	1.04 %	34.20 %	17.03 %	26.08 %	20.85 %	0.64 %	0.16 %	0	0
8	0.98 %	33.66 %	17.06 %	26.30 %	21.18 %	0.66 %	0.16 %	0	0
16	1.07 %	32.41 %	17.06 %	26.78 %	21.84 %	0.68 %	0.16 %	0	0
32	0.91 %	30.78 %	17.29 %	27.59 %	22.57 %	0.70 %	0.16 %	0	0
64	0.93 %	30.22 %	17.20 %	27.76 %	23.01 %	0.72 %	0.17 %	0	0
128	0.88 %	29.48 %	17.20 %	28.06 %	23.48 %	0.73 %	0.17 %	< 0.01 %	0
256	0.70 %	27.63 %	17.20 %	28.84 %	24.67 %	0.77 %	0.18 %	< 0.01 %	< 0.01 %
512	0.66 %	27.19 %	17.14 %	28.96 %	25.04 %	0.80 %	0.20 %	< 0.01 %	< 0.01 %
1024	0.69 %	26.14 %	16.91 %	29.03 %	25.99 %	0.92 %	0.30 %	0.02 %	< 0.01 %

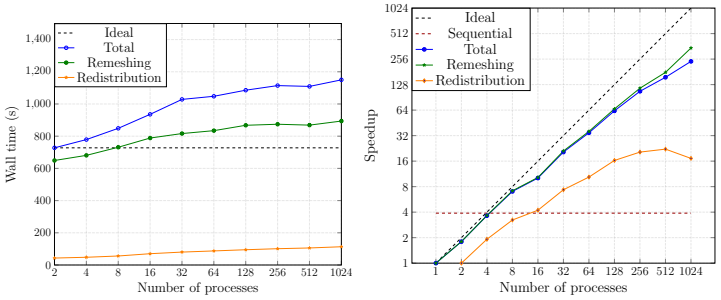
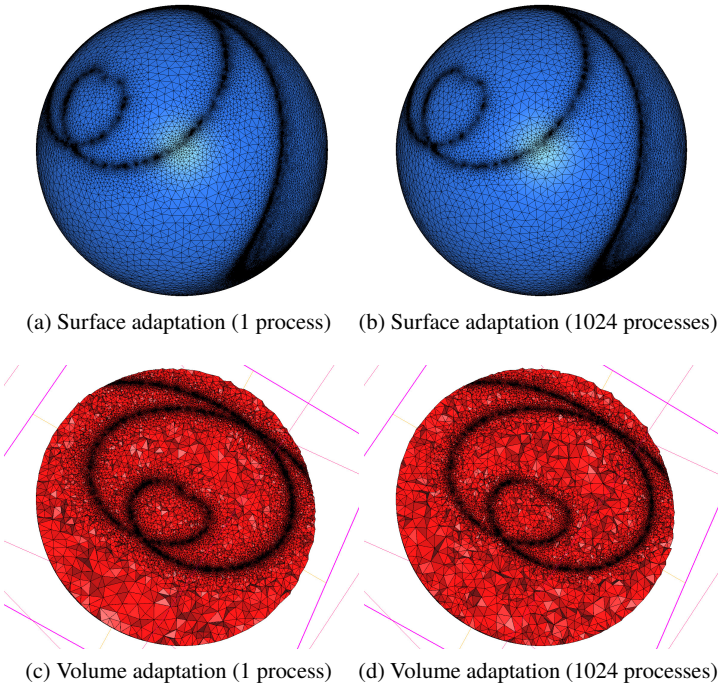


Fig. 4: `ParMmg` scaling test. **left** Weak scaling for the uniform refinement of a sphere. **right** Strong scaling for the adaptation to the double Archimedean spiral in a sphere.



(a) Surface adaptation (1 process) (b) Surface adaptation (1024 processes)

(c) Volume adaptation (1 process) (d) Volume adaptation (1024 processes)

Fig. 5: Adaptation to the double Archimedean spiral on 1 and 1024 processes.

mulations in which the mesh is treated as an elastic continuum are often employed (see e.g. [55] and references therein). Although the problem can be formulated as a mapping $\boldsymbol{\xi} : \Omega_{\mathbf{x}} \rightarrow \Omega_{\boldsymbol{\xi}}$ from the physical domain to a reference one, for computational efficiency [15] the PDE problem is often written for the inverse mapping $\mathbf{x} : \Omega_{\boldsymbol{\xi}} \rightarrow \Omega_{\mathbf{x}}$. Thus, a solution is sought directly for the mesh nodes position $\mathbf{x}$, and the mesh PDE is discretized in the (fixed) reference domain $\Omega_{\boldsymbol{\xi}}$. This framework allows a direct analogy between the mesh PDE and Lagrangian or Arbitrary Lagrangian Eulerian continuum mechanics, and its associated computational methods.

Introducing the displacement $\boldsymbol{\delta} = \mathbf{x} - \boldsymbol{\xi}$, the mesh PDE problem can be written without loss of generality as

$$\tau \partial_t \boldsymbol{\delta} - \boldsymbol{\nabla}_{\boldsymbol{\xi}} \cdot \boldsymbol{\sigma}(\boldsymbol{\delta}) = \boldsymbol{F}(\boldsymbol{\xi}, \boldsymbol{\delta}) \tag{9}$$

where $\boldsymbol{\sigma}(\boldsymbol{\delta})$ can be seen as a Cauchy stress tensor, and the volume force $\boldsymbol{F}(\boldsymbol{\xi}, \boldsymbol{\delta})$ depends on smoothness estimator. Problem (9) is supplemented by the boundary conditions

$$\begin{cases} \boldsymbol{\sigma}(\boldsymbol{\delta}) \cdot \hat{\mathbf{n}} = \boldsymbol{\sigma}_N \ ; & \forall \boldsymbol{\xi} \in \Gamma_{\boldsymbol{\xi}}^N \\ \boldsymbol{\xi} + \boldsymbol{\delta} = \mathbf{x}_D \ ; & \forall \boldsymbol{\xi} \in \Gamma_{\boldsymbol{\xi}}^D \\ \boldsymbol{\xi} + \boldsymbol{\delta} = \boldsymbol{\chi}(\mathbf{w}),\ \mathbf{w} \in \Sigma^S \ ; & \forall \boldsymbol{\xi} \in \Gamma_{\boldsymbol{\xi}}^S \end{cases} \tag{10}$$

where $\Gamma_{\boldsymbol{\xi}}^N$, $\Gamma_{\boldsymbol{\xi}}^D$, and $\Gamma_{\boldsymbol{\xi}}^S$, are subsets of the reference domain boundary defining natural, Dirichlet, and slip boundary conditions. In the last case we have introduced the parametrization of $\Gamma_{\boldsymbol{\xi}}^S$

$$\boldsymbol{\chi} : \Sigma \to \Gamma_{\boldsymbol{\xi}}^S, \quad \Sigma = [0,1] \times [0,1],\ \Gamma_{\boldsymbol{\xi}}^S \subset \mathbb{R}^3 \tag{11}$$

The slip boundary condition, very interesting for practical applications, is in principle non-linear as we require $\boldsymbol{\xi} + \boldsymbol{\delta}$ to be the inverse $\boldsymbol{\chi}$ for some admissible $\mathbf{w} \in \Sigma^S$. We will discuss this condition in more detail in Sect. 3.4.

The parameter/matrix τ is a dimensionless relaxation time that can be used to adjust the time scale of the mesh movement w.r.t. the physics. This parabolic regularisation/relaxation is also a good model for elliptic solvers coupled with some truncated iterative process, as often done in practice especially when explicit time stepping is used in the flow solver [15, 56, 16, 46, 7, 8].

Concerning the closure law, there exists a variety of possibilities in literature. A key role is played by the definition of a monitor function/matrix, which is in general a function of the flow solution in the physical space, and plays the role of the metric field in $h-$adaptation. In the simplest setting, if adaptation is performed w.r.t. the field u, one starts by defining a scalar function $\omega = \omega(u(\mathbf{x}))$. Typically one has

$$\omega = (1 + \alpha u + \beta \|\boldsymbol{\nabla}_{\mathbf{x}} u\| + \gamma \|\mathcal{H}_{\mathbf{x}}(u)\|)^p \tag{12}$$

In this work we consider two cases:

1. the Laplacian mesh movement corresponding to the closure $\boldsymbol{\sigma}(\boldsymbol{\delta}) = \omega(\mathbf{x})\boldsymbol{\nabla}_{\boldsymbol{\xi}}\boldsymbol{\delta}$
2. the elastic closure $\boldsymbol{\sigma}(\boldsymbol{\delta}) = 2\mu\boldsymbol{\epsilon}(\boldsymbol{\delta}) + \lambda\,\mathrm{tr}(\boldsymbol{\epsilon}(\boldsymbol{\delta}))$, with (μ, λ) constant Lamé coefficients, and $\boldsymbol{\epsilon}(\boldsymbol{\delta})$ the (symmetric) deformation tensor.

Note that for the Laplacian mesh movement, the classical elliptic equation

$$\boldsymbol{\nabla}_{\boldsymbol{\xi}} \cdot \left(\omega(\mathbf{x})\boldsymbol{\nabla}_{\boldsymbol{\xi}}\mathbf{x}\right) = \mathbf{0} \qquad \text{in } \Omega_{\boldsymbol{\xi}} \tag{13}$$

is obtained for $\boldsymbol{F}(\boldsymbol{\xi}, \boldsymbol{\delta}) = \boldsymbol{\nabla}_{\boldsymbol{\xi}} \cdot (\omega(\mathbf{x})\mathbf{Id})$. With elasticity, usually used to propagate an imposed boundary displacement, it is less clear what form the forcing should take.

The mesh PDEs are discretized with standard $\mathbb{P}^1$ finite elements on the reference/initial mesh, and new nodal positions are obtained via a small number of nodal Jacobi iterations that can be written as

$$\boldsymbol{\delta}_i^{[k+1]} = \boldsymbol{\delta}_i^{[k]} - (K_{ii}^{[k]})^{-1} \sum_{j \in \mathcal{B}_i} K_{ij}^{[k]} \boldsymbol{\delta}_j^{[k]} + \boldsymbol{F}_i^{[k]} \tag{14}$$

with K denoting the stiffness matrix, and K_{ii} being a block or a diagonal matrix depending on the formulation chosen. Note that these matrices are constant throughout the iterations when using linear elasticity.

We report hereafter some of the work done on this type of methods for different applications.

3.1 Embedded geometries and distance function: Laplace vs elasticity

One of our first aims is to enhance as much as possible the resolution of surfaces implicitly defined by a level-set function Φ. We assume to be able to work just with the Dirichlet and natural boundary conditions in (10). We use this as an exercise to compare the formulations presented in the previous section. Although simple, cost-wise the Laplacian movement presents a considerable non-linearity due to the dependence on the final configuration both of the stress definition, and in the force. As an alternative, we have tried to use linear elasticity as a mean to relax the strain locally generated by an applied volume force. To this end, we have used the same force definition for all systems, namely $\boldsymbol{F}(\boldsymbol{\xi}, \boldsymbol{\delta}) = \boldsymbol{\nabla}_{\boldsymbol{\xi}} \cdot (\omega(\mathbf{x})\mathbf{Id})$.

The definition of the monitor function can be set to mimic a mollified characteristic function corresponding to the domains $\Phi > 0$ (or $\Phi < 0$). A possible choice is

$$\omega_{\Phi}^{\mathrm{GB}} = \sqrt{\alpha_0 + \alpha_{\Phi} e^{-\beta_{\Phi}\Phi^2}} \tag{15}$$

with GB standing for gradient based. The constant term α_0 is defined to offset the stiffness in different regions of the domain. A first experiment consists in comparing Laplace and elasticity based deformation with a similar definition of the sensor. In the experiment we present, the reference domain is meshed with a relatively uniform triangulation, and the Newton-Jacobi update (14) is repeated for a few thousands iterations or until three orders of magnitude convergence are achieved.

We use as a target the level-set corresponding to an implicit curve resembling a 2D four-lobed flower. The values of the constants in (15) are slightly tuned to improve the resolution of the curve but have similar orders of magnitude in the two approaches: $(\alpha_{\Phi}, \beta_{\Phi}) = (40, 300)$ for the Laplace deformation, and $(\alpha_{\Phi}, \beta_{\Phi}) = (30, 200)$ for elasticity. The offset is chose as $\alpha_0 = 1$ for the Laplace deformation, while for elasticity we have set $\alpha_0 = 2.5|\kappa|$ with κ an approximation of the curvature of the level-set contours. The latter is introduced in an attempt to improve the capabilities of capturing singularities of the linear elastic deformation. The two approaches are compared in Fig. 6. While allowing more anisotropy in regions with smaller curvature, the elasticity based deformation tends to provide highly stretched meshes in vicinity of singularities as in the corners. This may be a limiting factor in more complex situations as e.g. when combining level-set with solution based adaptation.

To obtain a high resolution of the zero level-set without impoverishing too much the surrounding regions, the Laplacian movement could be combined with some way to directly control the mesh density at a given distance from the embedded surface. A possibility we use is a piecewise constant definition of the monitor:

$$\omega_\Phi^{\mathrm{PC}} = \begin{cases} \omega_1 & \text{if} \quad |\Phi| \leq \Phi_1 \\ \omega_j & \text{if} \quad \Phi_{j-1} < |\Phi| \leq \Phi_j \quad \forall j \in 2, N_\Phi - 1 \\ \omega_{N_\Phi} & \text{if} \quad \Phi_{N_\Phi - 1} < |\Phi| \end{cases} \tag{16}$$

The number of regions N_Φ is usually taken at least equal to three. This definition allows to set up areas of increasing stiffness, thus with higher density of nodes, as one approaches the zero level-set. The last jump around the usually small value Φ_0 provides the refinement sought around the embedded surface. In this area (15) could also be used, but a higher constant value works fairly well.

To give an example we consider a circular interface, and we compare on the left and central pictures of Fig. 7 the gradient based (or GB) definition with a 4 layers PC with : $(\Phi_1, \Phi_2, \Phi_3) = (0.05, 1, 1.75)$, and $(\omega_1, \omega_2, \omega_3) = (225, 90, 70, 20)$. We see that this definition allows to obtain a denser area of uniform mesh size closer to the zero level-set, thus virtually leaving out nodes for resolving other features, and allows a higher resolution in correspondence of the embedded circle. As a final comparison, in the rightmost picture in Fig. 7 we show the two approaches compared for the 4-lobed flower. This simple piece-wise constant definition of the monitor, combined with Laplacian deformation provides a good degree of control on the node density.

In the next paragraph we focus on applications based on this approach. The reader can refer to the PhD thesis [46] for more comparisons and applications involving the elastic deformation using the above definitions, and to [34, 11, 63] for the derivation of mesh PDEs which attempt to impose a given metric field without topology change.

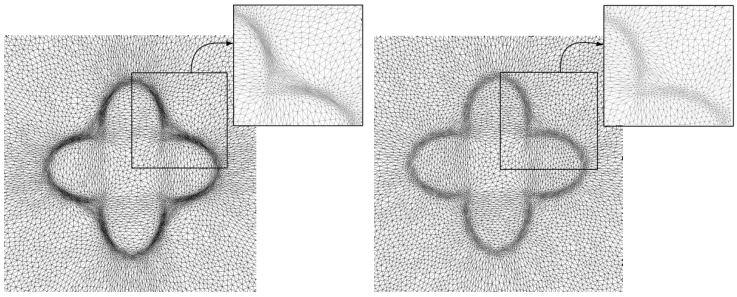
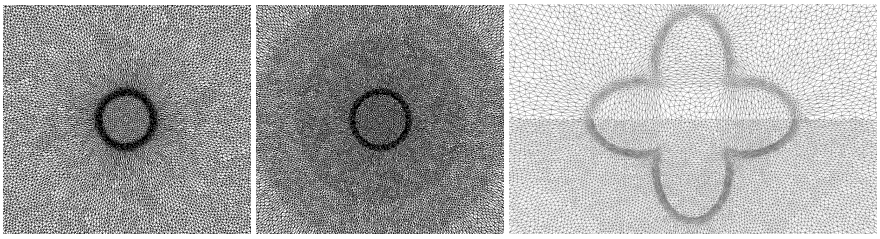


Fig. 6: Flower Adaptation: elasticity (left) versus Laplacian (right) based movement



(a) Circle: Laplacian GB (b) Circle: Laplacian PC (c) Flower: GB (top) - PC (bottom)

Fig. 7: Gradient based (GB) versus piecewise constant (PC) monitor function

3.2 Immersed boundaries and moving bodies

We consider the combination of level-set adaptation, immersed boundary methods, and solution based adaptation. The PDE setting is similar to that of Sect. 2.4, but the numerical results use a flow solver based on a stabilized residual based method in an Arbitrary-Lagrangian-Eulerian (ALE) setting [47, 46, 5]. We focus on forced motion of a solid, whose boundary is implicitly defined via the zero level-set of a signed distance function. To cope with the time dependent nature of the problem in this case we use the explicit knowledge of the exact geometry, and recompute the distance using the method proposed in [22]. As before, we assume to be able to work just with the Dirichlet and natural boundary conditions in (10).

As in Sect. 2.4, we wish to combine level-set and solution based adaptation. To this end, we introduce the smoothness monitor for the solution

$$\omega_u = \sqrt{1 + \alpha_u \widehat{\nabla u}^2} \tag{17}$$

where $\widehat{\nabla u}$ is a capped gradient norm (see e.g. [56, 16])

$$\widehat{\nabla u} = \min(1, \frac{\|\nabla_{\mathbf{x}} u\|}{\beta_u \|\nabla_{\mathbf{x}} u\|_{\text{ref}}}) \tag{18}$$

with the reference value $\|\nabla u\|_{\text{ref}}$ usually chosen as the maximum over the domain. Note that the gradient is computed w.r.t. the actual coordinates $\mathbf{x}$. The combined level-set/solution monitor function is finally defined as

$$\omega = \begin{cases} \omega_\Phi & \text{if} \quad |\Phi| \leq \epsilon \\ \max(\omega_\Phi, \omega_u) & \text{if} \quad |\Phi| > \epsilon \end{cases} \tag{19}$$

The initial mesh is adapted to the embedded surface and initial solution, solving the Newton-Jacobi iterations until a relative convergence of a few orders of magnitude, as discussed in the previous section. However, during the ALE time advancement,

only a few mesh iterations are performed. For the example shown we use 20 mesh iterations, with the last known mesh as initial guess. This gives a negligible overhead compared to the implicit Cranck-Nicholson ALE time stepping used for the flow.

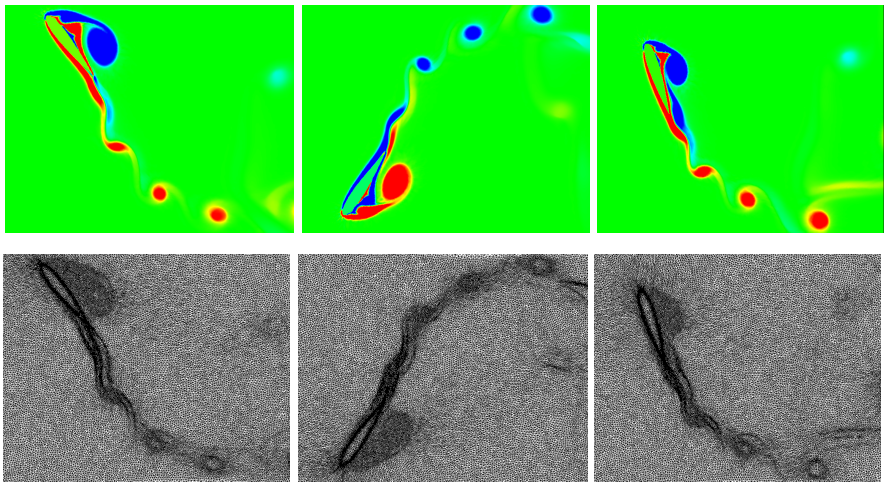


Fig. 8: Oscillation Naca snapshots. **top** Vorticity field. **bottom** Adaptive mesh.

As an example we consider a well known benchmark involving the forced pitching and heaving motion of a NACA 0015 airfoil at Reynolds $\approx$ 1100. We refer to [38, 12, 9] for the set up of the computation. Snapshots of the vorticity are reported in Fig. 8, showing the strong separation and vortex shedding taking place. In the same figure we report the corresponding meshes adapted w.r.t. both the level-set, and vorticity. The efficient capturing of the body movement and vorticity fields in the mesh deformation is quite impressive. To validate our approach we compare the lift and torque aerodynamic coefficients to reference literature data [38, 12, 9] in the top pictures of Fig. 9, observing an excellent agreement. We also compare the results obtained on the adaptive mesh, with those obtained on the initial reference triangulation (finest mesh size $h \approx 0.03$), and on a refined one (finest mesh size $h \approx 0.02$). Mesh data, and computational times are reported in Table 4. We can see that the $r-$adaptive approach allows computational savings of the order of 26.5%, and that without any particular optimization, only 11.7% of the computing time is devoted to the mesh PDE solver.

3.3 Free surface flows and moving shorelines

Another application with very similar issues is the simulation of free surface waves with moving wet/dry fronts. These can be modelled by the shallow water equations

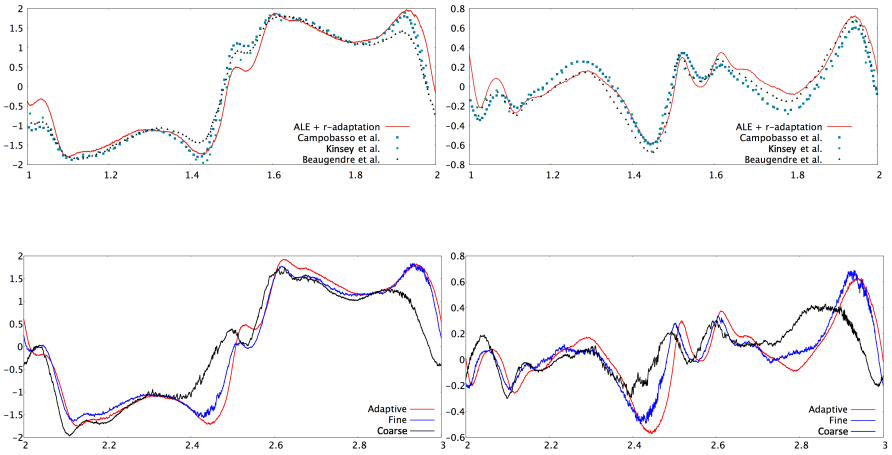


Fig. 9: Oscillating Naca - Aerodynamic coefficients vs time (Left: lift - Right: torque). **top** Comparison with literature. **bottom** Fine/adapted/coarse mesh results.

Table 4: Oscillating Naca: computational times for different simulations

Mesh (size)	Number of nodes	MMPDE CPU time [s]	Total CPU time [s]
Fixed $h_{\min} \approx 0.03$	53495	0	20515
Fixed $h_{\min} \approx 0.02$	33139	0	37716
Adaptive	33139	3241	27738

which can be written as a system of balance equations of the form

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathbf{F}_C(\mathbf{u}) + S(\mathbf{u};\mathbf{x}) = \mathbf{0} \ , \tag{20}$$

where $\mathbf{u}$ is the array of the water depth H, and horizontal mass flux vector, $\mathbf{F}_C$ the conservative fluxes, and $S(\mathbf{u};\mathbf{x})$ a source term modelling several effects such as those of bottom topography, friction, Coriolis forces, and others depending on the application. For applications involving flooding and complex wave propagation ideas similar to those discussed above can be used to adapt the mesh w.r.t shorelines and wave fronts. However, for wave propagation, and in general for hyperbolic problems, a majority of codes are based on an explicit time stepping kernel. This requires limiting somewhat the overhead of the mesh PDE solver for r-adaptation to be interesting. So we do not use the computation of a distance function here, but we proceed as follows:

- generate initial meshes with sizes increasing with the depth, so that a prescribed minimum size is attained at and close to the initial shoreline
- define the regularized Heaviside function

$$\phi_H(\mathbf{x}) = \begin{cases} 1 & \text{if } H(\mathbf{x}) > \epsilon_H \\ \phi(H) & \text{if } 0 < H(\mathbf{x}) \leq \epsilon_H \\ 0 & \text{if } H(\mathbf{x}) = 0 \end{cases}$$

- define a monitor function combining the dry detection function above and a capped gradient (cf. Eq. 18) as

$$\omega = \sqrt{1 + \alpha_\eta \widehat{\nabla \eta}^2 + \alpha_{\text{dry}} \|\nabla_{\mathbf{x}} \phi_H\|^2} \tag{21}$$

where if $b(\mathbf{x})$ denotes the topography, η is the free surface level $H + b$. The regularization $\phi(H) \in [0, 1]$ can be any (at least C^0) function modelling the depth profile in cells cut by the shoreline. Note that, differently than in the case of moving bodies, this curve is in general never known exactly. Even in the initial condition, its definition depends in practice on the resolution of the topographic data. In practice, defining $\phi(H)$ as a linear function is already enough to capture the wet/dry transition.

In the numerical examples shown here, the shallow water equations are solved by means of an explicit stabilized residual based approach in an ad-hoc ALE form. Although the design of numerical schemes is not the focus of this work, we mention that one of the challenges when dealing with the shallow water equations (and in general balance laws) is that the ALE formulation, and the associated remaps, needs to satisfy not only the classical conservation constraints (mass and momentum), but also additional ones related to the preservation of particular steady solutions (well-balanced). As for compressible fluid flows, additional constraints related to the admissibility of the solution (e.g. non-negativity of the depth) should also be embedded. Moreover, in real applications one has to make sure that the remap used for the data of the problem (e.g. topography) is not affected by truncation errors, which is also a constraining requirement. These aspects are covered in detail in [7], in which these constraints are combined within a conservative ALE formulation, with a high order quadrature based projection of the original data on the moving mesh.

Concerning the mesh solver, while for the initial solution (14) is solved until convergence within a few orders of magnitude, within each time step only 5 Jacobi relaxation iterations are performed to move the mesh, using as initial guess the last available one. As an example, we consider the simulation of the 2011 Tohoku-Honsu tsunami, the interested reader can refer to [7, 8] for thorough validation.

The leftmost picture on Fig. 10 shows an overview of the computational domain, including the position of the GPS buoys whose data has been used for validation, and the initial free surface perturbation computed based on the seismic data and with approach by Satake [53, 42]. The middle picture in the same figure shows a close up of the Japanese coast in correspondence of the Myagi prefecture (boxed area) and of the Sendai bay, with the adapted mesh in the initial instants of the propagation of the tsunami. The rightmost picture shows a further close up of the Myagi prefecture with the mesh adapted to the coastline. Note that the resolution of the reference

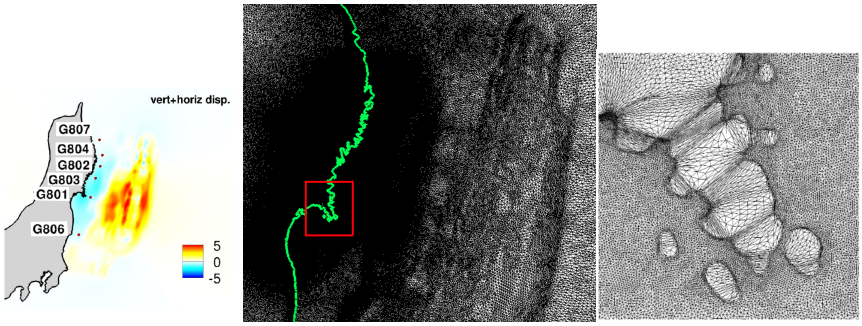


Fig. 10: Tohoku-Honsu tsunami. **left** Problem setup, GPS probes and initial wave height (in meters). **center** Adapted closer to the coastline of the Senady bay and Myagi prefecture (boxed area). **right** Shoreline of the Myagi prefecture.

mesh goes from 360 m close to the initial shoreline to 15 Km offshore. A fine mesh computation has been run for comparison using resolutions down to 120m close to the shore and of 5 Km offshore.

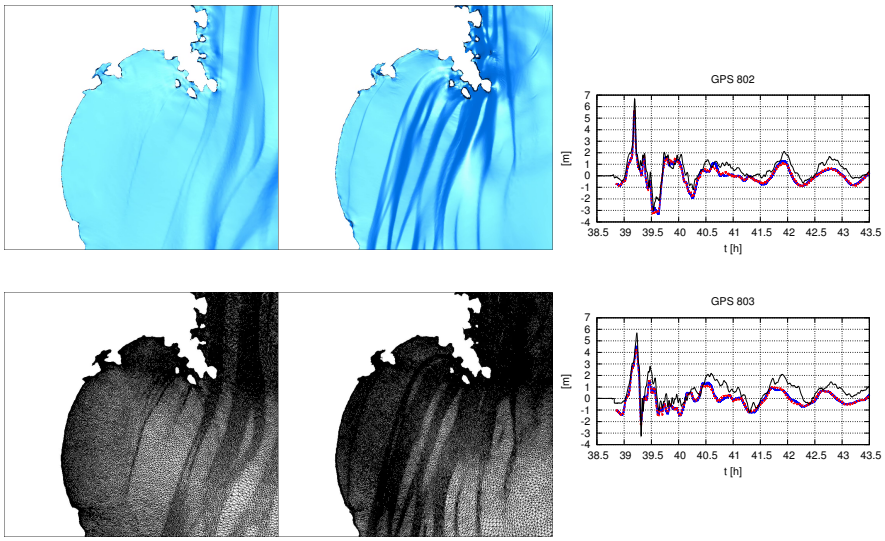


Fig. 11: Tohoku-Honsu tsunami. **left and center** Waves entering the Senday bay. **right** GPS bouys 802 and 803 (red: adaptive - blue: fine mesh - black: GPS data).

The left and middle pictures on Fig. 11 show the waves entering the Sendai bay, and the corresponding adapted meshes, while the rightmost pictures report the GPS time series, comparing the adaptive computations to the fine mesh, and to the

GPS data showing the potential of the approach used in resolving complex wave propagation on coarser grids.

3.4 A-posteriori limiting and curved boundaries in 3D

Mesh folding is a serious concern for mesh deformation with constant connectivity [36, 26, 59, 32, 29, 61]. To the best of our knowledge quite few applications of $r-$adaptation in 3D are available [43, 35]. On the other hand, we have first hand experience that straightforward extensions of nicely working methods as the variable-diffusion Laplacian method [15] to 3D suffer from tangling issues, especially when attempting to properly apply conditions (10) to curved boundaries.

A possible way out is to devise a moving mesh PDE which enjoys a non-negativity principle. This has been done e.g. in [35] using a gradient flux modelling stand-point. Although the method is shown to provide a stable smoother for complex geometries, the reference does not dwell on the possibility of handling a moving front interacting with a curved boundary. In addition, we rarely solve exactly the mesh PDE, but rather a relaxed approximation involving some iterative procedure. So a fully discrete approach to ensure control on volume positivity seems still to be a necessary tool, as much as positivity preserving limiters are when solving flow equations.

We present a solution relying on the combination of two main ideas: an a-posteriori limiter of embedded in the Newton-Jacobi iteration (14); the coupling of (14) with a local projection operator allowing to iteratively impose (10), and in particular the slip condition. A preliminary application to fluid mechanics simulations is shown in [19]. Here we recall the main ideas and we show an analytical example.

3.4.1 Nonlinear iterations and boundary projection

The solution of (9) with boundary conditions (10) is obtained by means of a non-linear iterative procedure. We assume that Dirichlet conditions are homogenous for the displacements, and verified by the initial solution. We then proceed as follows.

Predictor. For nodes $i \notin \Gamma^D_{\boldsymbol{\xi}}$ compute predicted nodal displacements from (14) with natural boundary conditions. Set $\mathbf{d}_i^0 = \boldsymbol{\delta}_i^{[k+1]} + \boldsymbol{\xi}_i - \mathbf{x}_i^{[k]}$.

Volume displacements. For nodes $i \notin \Gamma^S_{\boldsymbol{\xi}} \bigcup \Gamma^D_{\boldsymbol{\xi}}$, apply the non-linear relaxation:

$$\mathbf{d}_i^{s+1} = \begin{cases} \mu_i \mathbf{d}_i^s & \text{if } \min\limits_{K \in \mathcal{B}_i^{[k,k+1]}} |\Omega_K| < \epsilon \\ \mathbf{d}_i^s & \text{otherwise} \end{cases} \qquad \forall s \in [0, s_{\max} - 1] \tag{22}$$

Note that some of the nodes in $\mathcal{B}_i^{[k,k+1]}$ have already been updated, while others have not. In this respect this relax-update step is akin to a *Gauss-Seidel* process, guaranteeing the validity of the tetrahedra in the current ball. Note that slip nodes are kept aside since for these nodes we will introduce a projection of the predicted position on the correct geometry, which cannot be decoupled from the limiting.

Nonlinear boundary projection. For nodes $i \in \Gamma_{\boldsymbol{\xi}}^S$, setting $\mathbf{x}_i^{[k+1,0]} = \mathbf{x}_i^{[k]} + \mathbf{d}_i^0$, project iteratively on the parametrization of the geometry as follows

$$\mathbf{x}_i^{[k+1,s+1]} = \begin{cases} \boldsymbol{\chi}_\tau^{[k,k+1]}\left(\mathbf{w}(\mathbf{x}_i^{[k]} + \mu_i \mathbf{d}_i^s)\right) & \text{if } \min\limits_{K \in \mathcal{B}_i^{[k,k+1]}} |\Omega_K| < \epsilon \\ \boldsymbol{\chi}_\tau^{[k,k+1]}\left(\mathbf{w}(\mathbf{x}_i^{[k]} + \mathbf{d}_i^s)\right) & \text{otherwise} \end{cases} \qquad \forall s \in [0, s_{\max} - 1]$$

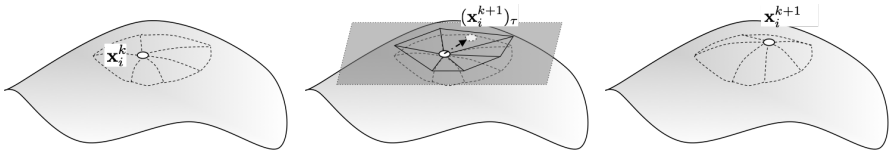
$$\mathbf{d}_i^{s+1} = \mathbf{x}_i^{[k+1,s+1]} - \mathbf{x}_i^{[k]}$$

The operator $\boldsymbol{\chi}_\tau^{[k,k+1]}\left(\mathbf{w}(\mathbf{x}_i^{[k]} + \mathbf{d}_i^s)\right)$ consists of the steps illustrated on Fig. 12:

1. consider the trace of $\mathcal{B}_i^{[k,k+1]}$ on the local tangent plane in $\mathbf{x}_i^{[k]}$ (Fig. 12-(a));
2. evaluate the tangent displacement and the trace triangle containing the displaced node (Fig. 12-(b));
3. use the parametrization within the trace triangle to update the surface normal;
4. project back on the parametrized surface using the updated normal (Fig. 12-(c)).

The limiting of the displacement is embedded via the sub-iterations on s.

Final update. Set $\mathbf{x}_i^{[k+1]} = \mathbf{x}_i^{[k]} + \mathbf{d}_i^{s_{\max}}$.



(a) (b) (c)

Fig. 12: Illustration of the boundary projection procedure.

In practice all these operations rely on piecewise cubic Bézier patches for the boundary position, and quadratic ones for the boundary normal vectors, available in the `Mmg` library, via curved point-normal triangles formalism [62]. This is used to evaluate all the projections from/to the tangent, as well as the normals.

As a final remark, note that *ridges*, i.e. intersection curves of two or more manifold surfaces, are treated in a very similar way, except that the geometrical model is now a piecewise-cubic Bézier curve. On the other hand, *corners*, i.e. intersection points

of two or more ridges, are added to the Dirichlet nodes. These conditions limit somewhat the flexibility of the method.

3.4.2 Moving front passing over a spherical boundary

As an example we consider a regularized Heaviside function moving in a domain of dimensionless length 3, moving at a dimensionless speed 0.2. A C^1 regularization of the discontinuity is defined in a thin layer $\delta = 5 \times 10^{-3}$. The shape of the spatial domain is a quarter cylinder of radius 1.5 along the x-axis with $x \in [-1.5, 1.5]$, surrounding a quarter sphere centered at the origin with radius 0.5. This test case contains at the same time curved surfaces, ridges (the intersection curves of the sphere with each symmetry planes) and corners (the intersection points of the sphere with both the symmetry planes), and a sharp solution moving over the geometry. Starting with an isotropic tetrahedrization of edge size $h = 0.05$, dynamic mesh adaptation is performed on this analytical solution every $\Delta t = 0.25$. Adaptation is performed with Laplace mesh deformation, using the monitor (17)-(18) with $(\alpha, \beta) = (40, 0.1)$. The number of Jacobi iterations is set to 30.

The adapted meshes are shown in Fig. 13. Mesh validity is preserved both when the front is sliding over the sphere surface and, crucially, when it touches and leaves the sphere at the corners. Without the application of the a-posteriori limiter we have been unable to run this example with existing standard formulations of the Laplacian model without obtaining mesh tangling.

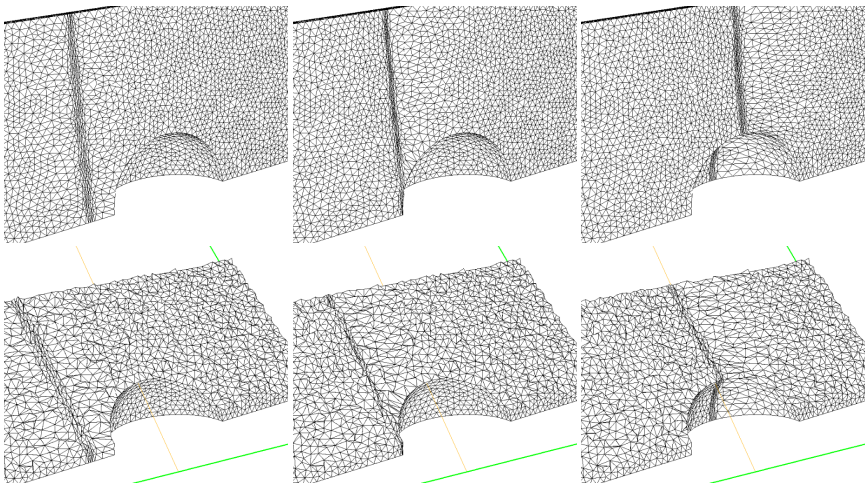


Fig. 13: Moving front test case, adapted mesh and volumic cuts at different times.

4 Summary and perspectives

This paper reviews recent work performed using on the capabilities of the remeshing software platform `Mmg`. As other open source packages for computational mechanics or mesh generation, this platform allows to apply its existing functionalities to problems of interest, as well as developing new meshing/remeshing methods.

Ongoing work related to the results presented are related to the robustification of the 3D mesh deformation method, to the investigation of the impact of adaptation in the framework of higher order embedded methods [54, 48], to the full support of non-manifold surface adaptation and anisotropic remeshing in parallel, to the exploration of hybrid parallelization strategies.

References

1. Mmg platform website. URL https://www.mmgtools.org/
2. `Mmg` version 5.5.2. SWHID: swh:1:rel:fe173a75f45f079d363d5a82204c9737550c5d79; REPOSITORY: https://github.com/MmgTools/mmg
3. `Mshdist`. SWHID: swh:1:dir:ae83c7e306d8a20484e74e5588cdb19ab0ae0d71; REPOSITORY: https://github.com/ISCDtoolbox/Mshdist
4. `ParMmg` version 1.3.0. SWHID:swh:1:rel:bf45d6314a386455d53a6c351be771d6252e0d43; REPOSITORY: https://github.com/MmgTools/ParMmg
5. Abgrall, R., Ricchiuto, M.: High order methods for CFD. In: R.d.B. Erwin Stein, T.J. Hughes (eds.) Encyclopedia of Computational Mechanics, Second Edition. John Wiley and Sons (2017)
6. Alauzet, F.: A parallel matrix-free conservative solution interpolation on unstructured tetrahedral meshes. Comp.Meth.Appl.Mech.Engrg. **299**, 116 – 142 (2016)
7. Arpaia, L., Ricchiuto, M.: r-adaptation for shallow water flows: conservation, well balancedness, efficiency. Computers & Fluids **160**, 175 – 203 (2018)
8. Arpaia, L., Ricchiuto, M.: Well balanced residual distribution for the ALE spherical shallow water equations on moving adaptive meshes. J.Comput.Phys. **405**, 109173 (2020)
9. Beaugendre, H., Morency, F.: Innovative Model for Flow Governed Solid Motion based on Penalization and Aerodynamic Forces and Moments. Inria Research Report RR-8718 (2015)
10. Benard, P., Balarac, G., Moureau, V., Dobrzynski, C., Lartigue, G., D'Angelo, Y.: Mesh adaptation for large-eddy simulations in complex geometries. Int.J.Numer.Meth.Fl. **81**, 719–740 (2016)
11. Budd, C.J., Huang, W., Russell, R.D.: Adaptivity with moving grids. Acta Numerica **18**, 111–241 (2009)
12. Campobasso, M.S., Drofelnik, J.: Compressible navier-stokes analysis of an oscillating wing in a power-extraction regime using efficient low-speed preconditioning. Comput Fluids **67**, 26–40 (2012)
13. Castaños, J.G., Savage, J.E.: The dynamic adaptation of parallel mesh-based computation. In: PPSC (1997)
14. Cavallo, P.A., Sinha, N., Feldman, G.M.: Parallel unstructured mesh adaptation method for moving body applications. AIAA Journal **43**(9), 1937–1945 (2005)
15. Ceniceros, H.D., Hou, T.Y.: An efficient dynamically adaptive mesh for potentially singular solutions. J.Comput.Phys. **172**(2), 609 – 639 (2001)
16. Chen, G., Tang, H., Zhang, P.: Second-order accurate godunov scheme for multicomponent flows on moving triangular meshes. Journal of Scientific Computing **34**(1), 64–86 (2008)
17. Chrisochoides, N., Nave, D.: Parallel delaunay mesh generation kernel. International Journal for Numerical Methods in Engineering **58**(2), 161–176 (2003)

18. Cirrottola, L., Froehly, A.: Parallel unstructured mesh adaptation using iterative remeshing and repartitioning. Inria Research Report RR-9307 (2019). URL https://hal.inria.fr/hal-02386837
19. Cirrottola, L., Froehly, A., Guardone, A., Quaranta, G., Re, B., Ricchiuto, M.: R-adaptation for unsteady compressible flow simulations in three dimensions
20. Dapogny, C., Dobrzynski, C., Frey, P.: Three-dimensional adaptive domain remeshing, implicit domain meshing, and applications to free and moving boundary problems. Journal of Computational Physics **262**, 358 – 378 (2014)
21. Dapogny, C., Frey, P.: Computation of the signed distance function to a discrete contour on adapted triangulation. Rech. Aérospat. (49), 193–219 (2012)
22. Dapogny, C., Frey, P.: Computation of the signed distance function to a discrete contour on adapted triangulation. Calcolo **49**, 193 – 219 (2012)
23. De Cougny, H.L., Shephard, M.S.: Parallel refinement and coarsening of tetrahedral meshes. International Journal for Numerical Methods in Engineering **46**(7), 1101–1125 (1999)
24. Digonnet, H., Coupez, T., Laure, P., Silva, L.: Massively parallel anisotropic mesh adaptation. International Journal of High Performance Computing Applications (2017)
25. Dobrzynski, C., Frey, P.: Anisotropic delaunay mesh adaptation for unsteady simulations. In: R.V. Garimella (ed.) Proceedings of the 17th International Meshing Roundtable, pp. 177–194. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
26. Dwight, R.: Robust mesh deformation using linear elasticity equations. In: H. Deconinck, E. Dick (eds.) Computational Fluid Dynamics 2006. Springer Berlin Heidelberg (2009)
27. Farrell, P., Maddison, J.: Conservative interpolation between volume meshes by local galerkin projection. Comp.Meth.Appl.Mech.Engrg. **200**(1), 89 – 100 (2011)
28. Flaherty, J., Loy, R., Özturan, C., Shephard, M., Szymanski, B., Teresco, J., Ziantz, L.: Parallel structures and dynamic load balancing for adaptive finite element computation. Applied Numerical Mathematics **26**(1), 241 – 263 (1998)
29. Fortunato, M., Persson, P.O.: High-order unstructured curved mesh generation using the winslow equations. J. Comput. Phys. **307**(C), 1–14 (2016)
30. Frey, P., Alauzet, F.: Anisotropic mesh adaptation for cfd computations. Comput. Methods Appl. Mech. Engrg. (194), 5068–5082 (2005)
31. Frey, P., George, P.: Mesh generation. Application to finite elements. Wiley (2008)
32. Hansen, G., Zardecki, A., Greening, D., Bos, R.: A finite element method for unstructured grid smoothing. J.Comput.Phys. **194**(2), 611 – 631 (2004)
33. Hermes, D., Persson, P.O.: High-order solution transfer between curved triangular meshes (2018)
34. Huang, W.: Variational mesh adaptation: Isotropy and equidistribution. J.Comput.Phys. **174**(2), 903 – 924 (2001)
35. Huang, W., Kamenski, L.: On the mesh nonsingularity of the moving mesh pde method. Math.Comp. **87**, 1887–1911
36. Johnson, A., Tezduyar, T.: Mesh update strategies in parallel finite element computations of flow problems with moving boundaries and interfaces. Comp.Meth.Appl.Mech.Engrg. **119**(1), 73 – 94 (1994)
37. K. Khadra P. Angot, S.P., Caltagirone, J.: Fictitious domain approach for numerical modelling of navier–stokes equations. Int.J.Numer. Meth. Fluids (34), 651–684 (2000)
38. Kinsey, T., Dumas, G.: Parametric study of an oscillating airfoil n a power-extraction regime. AIAA Journal **46**(6), 543–561 (2008)
39. Knupp, P., Steinberg, S.: Fundamentals of Grid Generation. The Fundamentals of Grid Generation. Taylor & Francis (1993)
40. Kucharik, M., Shashkov, M.: Extension of efficient, swept-integration-based conservative remapping method for meshes with changing connectivity. International Journal for Numerical Methods in Fluids **56**(8), 1359–1365 (2008)
41. Kucharik, M., Shashkov, M., Wendroff, B.: An efficient linearity-and-bound-preserving remapping method. J.Comput.Phys. **188**(2), 462–471 (2003)
42. LeRoy, S., Lemoine, A., Pedreros, R., Rousseau, M.: Tohoku-oki 2011 tsunami high-resolution modeling and sensitivity to the rupture complexity: Kamaishi and sendai areas. In: French Japanese week on disaster risk reduction. Tokyo (2017)

43. Li, R., Tang, T., Zhang, P.: A moving mesh finite element algorithm for singular problems in two and three space dimensions. J.Comput.Phys. **177**(2), 365 – 393 (2002)
44. Lorini, M., Dobrzynski, C., Perrier, V., Ricchiuto, M.: Preliminary results of a discontinuous galerkin immersed boundary method combining penalization and anisotropic mesh adaptation. In: Proc.s ECCM 6/ECFD 7, pp. 1875–1885 (2020)
45. Mittal, R., Iaccarino, G.: Immersed boundary methods. Ann.Rev.Fl.Mech. (37), 239–261 (2005)
46. Nouveau, L.: Adaptive residual based schemes for solving the penalized Navier Stokes equations with moving bodies : application to ice shedding trajectories. Theses, Université de Bordeaux (2016). URL https://tel.archives-ouvertes.fr/tel-01500093
47. Nouveau, L., Beaugendre, H., Ricchiuto, M., Dobrzynski, C., Abgrall, R.: An adaptive ALE residual based penalization approach for laminar flows with moving bodies. Inria Research Report RR-8936 (2016)
48. Nouveau, L., Ricchiuto, M., Scovazzi, G.: High-order gradients with the shifted boundary method: An embedded enriched mixed formulation for elliptic pdes. J.Comput.Phys. **398**, 108898 (2019)
49. Oliker, L., Biswas, R., Gabow, H.N.: Parallel tetrahedral mesh adaptation with dynamic load balancing. Parallel Computing **26**(12), 1583 – 1608 (2000)
50. Park, M.A., Krakos, J.J., Michal, T.A., Loseille, A., Alonso, J.J.: Unstructured Grid Adaptation: Status, Potential Impacts, and Recommended Investments Towards CFD 2030. In: AIAA Fluid Dynamics Conference, AIAA AVIATION Forum. Washington DC, United States (2016)
51. Park, M.A., Loseille, A., Krakos, J., Michal, T.R., Alonso, J.J.: Unstructured grid adaptation: Status, potential impacts, and recommended investments towards cfd 2030. In: AIAA AVIATION Forum, pp. –. American Institute of Aeronautics and Astronautics (2016)
52. Re, B., Dobrzynski, C., Guardone, A.: An interpolation-free ale scheme for unsteady inviscid flows computations with large boundary displacements over three-dimensional adaptive grids. J.Comput.Phys. **340**, 26 – 54 (2017)
53. Satake, K., Fujii, Y., Harada, T., Namegaya, Y.: Time and space distribution of coseismic slip of the 2011 tohoku earthquake as inferred from tsunami waveform data. Bull. Seismol. Soc. Am. **103**, 1473 – 1492 (2013)
54. Song, T., Main, A., Scovazzi, G., Ricchiuto, M.: The shifted boundary method for hyperbolic systems: Embedded domain computations of linear waves and shallow water flows. J.Comput.Phys. **369**, 45–79 (2018)
55. Stein, K., Tezduyar, T., Benney, R.: Mesh moving techniques for fluid-structure interactions with large displacements. Journal of Applied Mechanics **70**(1), 58–63 (2003)
56. Tang, H., Tang, T.: Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws. SIAM Journal on Numerical Analysis **41**(2), 487–515 (2003)
57. Tang, T.: Moving mesh methods for computational fluid dynamics. Contemporary mathematics **383**, 141–174 (2005)
58. Thomas, P., Lombard, C.: Geometric conservation law and its application to flow computations on moving grids. AIAA journal **17**(10), 1030–1037 (1979)
59. Toulorge, T., Geuzaine, C., Remacle, J.F., Lambrechts, J.: Robust untangling of curvilinear meshes. J.Comput.Phys. **254**, 8 – 26 (2013)
60. Trulio, J.G., Trigger, K.R.: Numerical solution of the one-dimensional lagrangian hydrodynamic equations. Tech. rep., California. Univ., Livermore, CA (United States). Lawrence Radiation Lab. (1961)
61. Turner, M., Peiró, J., Moxey, D.: Curvilinear mesh generation using a variational framework. Computer-Aided Design **103**, 73 – 91 (2018). 25th International Meshing Roundtable Special Issue: Advances in Mesh Generation
62. Vlachos, A., Peters, J., Boyd, C., Mitchell, J.L.: Curved pn triangles. In: Proceedings of the 2001 Symposium on Interactive 3D Graphics, I3D '01, p. 159–166. Association for Computing Machinery, New York, NY, USA (2001)
63. Weizhang, H., Russell, R.: Adaptive Moving Mesh Methods, *Applied Mathematical Sciences*, vol. 174. Springer (2011)