VI International Conference on Adaptive Modeling and Simulation
ADMOS 2013
J. P. Moitinho de Almeida, P. Díez, C. Tiago and N. Parés (Eds)

FULLY SPACE-TIME METRIC BASED ANISOTROPIC MESH ADAPTATION FOR UNSTEADY PROBLEMS

G. JANNOUN, E. HACHEM, J. VEYSSET, J-F. ZARAGOCI, AND T. COUPEZ

MINES ParisTech, Center for Materials Forming (CEMEF),
UMR CNRS 7635, BP 207, 06904 Sophia-Antipolis, France
e-mail: ghina.el_jannoun@mines-paristech.fr, www.cemef.mines-paristech.fr/

Key words: Anisotropic Mesh Adaptation, Adaptive Time-Stepping, Paradoxical meshing technique, Time-dependent problems

Abstract. This paper presents a novel method for building unstructured meshes for time-dependent problems. We start by introducing the classical anisotropic mesh adaptation technique proposed in [1, 2]. The latter is developed based on the length distribution tensor approach and the associated a posteriori edge based error analysis. Then we extend the mesh adaptation technique to contain adaptive time advancing. A newly developed time error estimator is constructed and intends to homogenize the global error over space and time. The main purpose of this work is the development of a novel meshing algorithm, the paradoxical meshing, that provides optimal space and time meshes suitable for several simulation time subintervals. The advantage of the proposed method relies in its conceptual and computational simplicity as it only requires from the user a number of nodes and a frequency of adaptation according to which the mesh and the time-steps are automatically adapted. Numerical solutions on time-dependent problems demonstrate the accuracy and efficiency of the proposed space-time error estimator.

1 INTRODUCTION

Despite the increasing computer performances and the progress of computational fluid dynamics in modelling and simulating time dependent PDEs, numerical restrictions are still present and caused by the complexity of the numerical simulations.

Anisotropic mesh adaptation has proved to be a powerful strategy to improve the accuracy and efficiency of finite element methods. It enables the capture of multi-scaled physical or mechanical phenomena. The method, as developed in [1], allows the creation of highly stretched and highly directional elements leading to very good capture of the gradients of the solution and the internal and boundary layers. Moreover, it provides a good level of accuracy within a reasonable degree of freedom. Another extension was proposed in [2], and accounts for time-step adaptation. Based on the derived error estimator in space and the solutions at the previous times, the proposed algorithm automatically computes an appropriate time-step for the following computations.

The above described mesh adaptation is optimal for steady problems. It is theoretically valid for transient CFD applications only when applied, together with the time adaptation method, at every solver iteration. Nevertheless this would excessively increase the computational cost and lead to an accumulation of interpolation errors. Fixing a low frequency of adaptation can help solving these issues but this might lead to a time lag between the mesh and the solution. To overcome these problems, a new fully adaptive method is proposed in this paper: the paradoxical meshing. It intends to predict the solution's evolution over a period of time and to automatically generate corresponding mesh and set of time-step sizes.

This paper is structured as follows: we start section 2 with a brief description of the classical anisotropic mesh adaptation. Section 3 is dedicated to the time adaptive technique. The good performance of these methods is evaluated on a 3D example with complex geometry. The extension of these two methods into the paradoxical meshing algorithm is described in Section 4. Finally, in section 5, we test the efficiency and accuracy of the space-time adaptive algorithm on time-dependent problems.

2 Construction of an anisotropic mesh for stationary problems

In [1], we have developed an a posteriori edge based spatial error estimator relying on the length distribution tensor approach. Working on a nodal based metric, an anisotropic mesh adaptation procedure is obtained under the constraint of a fixed number of nodes.

2.1 Edge based error estimation

We consider $u \in \mathcal{C}^2(\Omega) = \mathcal{V}$ and $\mathcal{V}_h$ a simple P^1 finite element approximation space:

$$\mathcal{V}_h = \left\{ w_h \in \mathcal{C}^0(\Omega), w_h|_K \in P^1(K), K \in \mathcal{K} \right\}$$

where $\Omega = \bigcup_{K \in \mathcal{K}} K$ and K is a simplex (segment, triangle, tetrahedron, ...).

We define $\mathbf{X} = \left\{ \mathbf{X}^i \in \mathbb{R}^d,\, i = 1, \cdots, N \right\}$ as the set of nodes of the mesh and we denote by U^i the nodal value of u at $\mathbf{X}^i$ and we let Π_h be the Lagrange interpolation operator from $\mathcal{V}$ to $\mathcal{V}_h$ such that:

$$\Pi_h u(\mathbf{X}^i) = u(\mathbf{X}^i) = U^i\,,\ \forall i = 1, \cdots, N$$

As shown in figure 1, we define the set of nodes connected to node i by

$$\Gamma(i) = \left\{ j\,,\ \exists^i K \in \mathcal{K}\,,\ \mathbf{X}^i, \mathbf{X}^j \text{ are nodes of } K \right\}$$

By introducing the following notation: $\mathbf{X}^{ij} = \mathbf{X}^j - \mathbf{X}^i$ and using the analysis carried in [1], we can set the following results:

$$\nabla u_h \cdot \mathbf{X}^{ij} = U^{ij}\,, \tag{1}$$

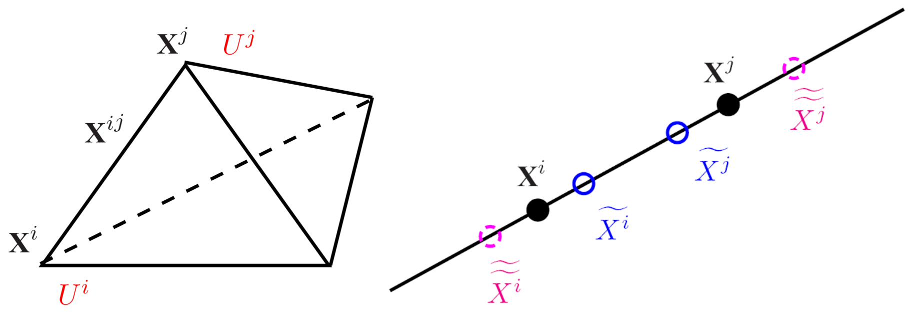

Figure 1: length X^{ij} of the edge joining nodes i and j (left). Varying the edge in its own direction (right).

$$|| \underbrace{\nabla u_h \cdot \mathbf{X}^{ij}}_{U^{ij}} - \nabla u(X^i) \cdot \mathbf{X}^{ij} || \leq \max_{Y \in [X^i, X^j]} |\mathbb{H}(u)(Y)\mathbf{X}^{ij} \cdot \mathbf{X}^{ij}| \,, \tag{2}$$

where $\mathbb{H}(u) = \nabla^{(2)} u$ is the associated Hessian of u. A definition of the projected second derivative of u is obtained using (1) and the interpolation operator on ∇u :

$$\nabla g_h \mathbf{X^{ij}} \cdot \mathbf{X^{ij}} = g^{ij} \cdot \mathbf{X^{ij}} \tag{3}$$

where $\nabla g_h = \Pi_h \nabla u$, $g^i = \nabla u(\mathbf{X^i})$ and $g^{ij} = g^j - g^i$.
Using a mean value argument, we set that:

$$\exists y \in [x^i, x^j] | g^{ij} \cdot \mathbf{X^{ij}} = \mathbb{H}(u)(Y)\mathbf{X}^{ij} \cdot \mathbf{X}^{ij} \,.$$

This projection is considered as an expression of the error along the edge:

$$e_{ij} = g^{ij} \cdot \mathbf{X^{ij}} \tag{4}$$

However a gradient recovery procedure is needed as the gradient of u is not known and is not necessarily continuous at the nodes of the mesh.

2.2 Gradient Recovery

Based on an optimization analysis, the author in [1, 2] proposes a recovery gradient operator defined by:

$$G^i = (\mathbb{X}^i)^{-1} \sum_{j \in \Gamma(i)} U^{ij} \mathbf{X}^{ij} \tag{5}$$

where $\mathbb{X}^i = \frac{d}{|\Gamma(i)|} \sum_{j \in \Gamma(i)} \mathbf{X}^{ij} \otimes \mathbf{X}^{ij}$ is the length distribution tensor at node $\mathbf{X^i}$. Note that this construction preserves the second order:

$$\left| \left(G^i - g^i \right) \cdot \mathbf{X^{ij}} \right| \sim \left(\mathbb{H}(u) \mathbf{X}^{ij} \cdot \mathbf{X}^{ij} \right)$$

where G^i is the recovery gradient at node i (given by (5)) and g^i being the exact value of the gradient at node i. The error is evaluated by substituting g by G in (4):

$$e_{ij} = G^{ij} \cdot \mathbf{X^{ij}}$$

2.3 Metric construction from the edge distribution tensor

Taking into account this error analysis, we construct the metric for the unit mesh as follows:

$$\mathbb{M}^i = \left(\frac{d}{|\Gamma(i)|} \sum_{j \in \Gamma(i)} \mathbf{X}^{ij} \otimes \mathbf{X}^{ij} \right)^{-1}$$

For a complete justification of this result, the reader is referred to [1, 2].

2.4 Error behavior due to varying the edge length

We examine now how the error behaves when we change the length of the edges by stretching coefficients

$$\mathcal{S} = \left\{ s_{ij} \in \mathbb{R}^+ \,,\, i = 1, \cdots, N \,,\, j = 1, \cdots, N \,,\, \Gamma(i) \cap \Gamma(j) \neq \phi \right\}$$

Stretching factors $s \in \mathbb{R}$ are employed to link the error variations to the changes in edge lengths:

$$\left\{ \begin{array}{l} \widetilde{\mathbf{X}_{ij}} = s\mathbf{X}_{ij} \\ ||\widetilde{e_{ij}}|| = s^2 ||e_{ij}|| = s^2 ||G^{ij} \cdot \mathbf{X}_{ij}|| \end{array} \right. \tag{6}$$

where $\widetilde{e_{ij}}$ and $\widetilde{\mathbf{X}_{ij}}$ are the target error at edge ij and its associated edge length.
The metric associated with $\mathcal{S}$ can be redefined as:

$$\widetilde{\mathbb{M}^i} = \frac{|\Gamma(i)|}{d} \left(\widetilde{\mathbb{X}^i} \right)^{-1} \quad \text{with} \quad \widetilde{\mathbb{X}^i} = \frac{d}{|\Gamma(i)|} \sum_{j \in \Gamma(i)} s_{ij}^2 \mathbf{X}^{ij} \otimes \mathbf{X}^{ij} \tag{7}$$

being is the length distribution tensor. Let n_{ij} be the number of created nodes in relation with the stretching factor s_{ij} and along the edge ij. When scaling the edges by a factor s_{ij}, the error changes quadratically so that the number of created nodes along the edge ij is given by:

$$n_{ij} = \left(\frac{\widetilde{e_{ij}}}{e_{ij}} \right)^{-\frac{1}{2}} = s_{ij}^{-1}$$

Recall that $\widetilde{e_{ij}}$ denotes the induced error for edge $\widetilde{X^{ij}}$. As per node i, the number of created nodes along the different edges' directions is given by the following tensor:

$$N^i = \left(\mathbb{X}^i \right)^{-1} \left(\frac{d}{|\Gamma(i)|} \sum_{j \in \Gamma(i)} {n_{ij}}^2 \mathbf{X}^{ij} \otimes \mathbf{X}^{ij} \right)$$

So that the total number of created nodes per node i is:

$$n^i = \sqrt{ \det \left((\mathbb{X}^i)^{-1} \left(\frac{d}{|\Gamma(i)|} \sum_{j \in \Gamma(i)} {n_{ij}}^2 \mathbf{X}^{ij} \otimes \mathbf{X}^{ij} \right) \right) }$$

Assuming a uniform totally balanced error along the edge $\widetilde{e_{ij}} = e = \text{cst}$, we get a direct relation between N and e as follows:

$$n^{ij}(e) = s_{ij}^{-1}(e) = \left(\frac{e}{e_{ij}}\right)^{-\frac{1}{2}}$$

Hence for a node i we have

$$n^i(e) = \sqrt{\det\left((\mathbb{X}^i)^{-1}\left(\frac{d}{|\Gamma(i)|}\sum_{j\in\Gamma(i)} n_{ij}(e)^2 \mathbf{X}^{ij}\otimes\mathbf{X}^{ij}\right)\right)}$$

Replacing $n_{ij}(e)$ by its expression,

$$n^i(e) = e^{-\frac{d}{2}}\sqrt{\det\left((\mathbb{X}^i)^{-1}\left(\frac{d}{|\Gamma(i)|}\sum_{j\in\Gamma(i)} e_{ij}\mathbf{X}^{ij}\otimes\mathbf{X}^{ij}\right)\right)}$$

and this is equivalent to:

$$n^i(e) = e^{-\frac{d}{2}} n^i(1)$$

so that the total number of nodes in the adapted mesh is: $N = e^{-\frac{d}{2}}\sum\limits_i n^i(1)$.

Hence, the global induced error for N nodes can be determined by:

$$e(N) = \left(\frac{N}{\sum\limits_i n^i(1)}\right)^{-\frac{2}{d}}$$

Therefore the corresponding stretching factors under the constraint of a fixed number of nodes N are given by:

$$s_{ij} = \left(\frac{e}{e(N)}\right)^{-\frac{1}{2}} = \left(\frac{\sum\limits_i n^i(1)}{N}\right)^{\frac{2}{d}} e_{ij}^{-1/2}$$

Note that the mesh does not change during time advancing but at a certain time level t^n. Hence, an optimal mesh at a time level t^n need not be an optimal one at t^{n+1} which is the case when propagating a discontinuity. This raises the question about the frequency of remeshing.

As time-dependent problems exhibit arbitrary progression with time, the duration of applicability of a mesh cannot be known apriori. When the time-step size is greater than the length of the mesh's time interval, the solution may propagate into a non pre-adapted region of the domain resulting in a mesh/solution lag. Adapting the mesh at every solver iteration guarantees that the spatial error remains bounded. Nevertheless, this approach increases significantly the computational cost and leads to the accumulation of interpolation errors polluting the solution.

3 Time adaptation procedure

The focus now can be waived to the choice of the time-step size computed at each solver iteration. The main objective is to produce a time-step that preserves the accuracy of the mesh adapted solution and avoids unnecessarily small time-step sizes. This method was first introduced and validated in [2]. In this work, we revisit the developed method with the intention of extending it to a fully space-time adaptivity.

We apply the above described analysis in 1D where the only variable is time. Denote by $\mathcal{T} = \{t^0, \cdots, t^{n-1}, t^n, t^{n+1}, \cdots\}$ and let $t^{nk} = |t^n - t^k|$, n, $k \in \mathcal{T}$ be the temporal nodes and $\Delta t^n = t^{nn+1}$ the time increments as shown in figure 2. Assume that the solution is already computed on the whole domain up to time t^n. The aim is to choose an appropriate time-step Δt^n.

Without loss of generality, the analysis will be carried on an arbitrary spatial node i. Note that at a spatial node i, we only have one time edge to be determined $(t^n t^{n+1})$.

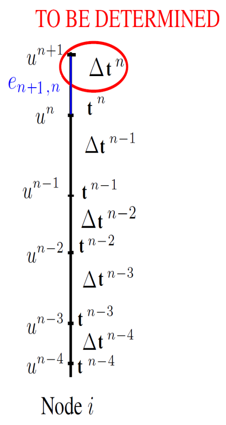

Figure 2: Temporal discretization at the spatial node i.

Define $\{\tau_{nn+1}\}$ to be the temporal edge scaling (stretching) factor such that:

$$\begin{aligned} \tilde{e}_{n+1,n} &= \tau^2_{n+1,n} e_{n+1,n} \\ \left|\tilde{t}^{n+1,n}\right| &= \tau_{n+1,n} \left|t^{n+1,n}\right| \end{aligned} \tag{8}$$

where $e_{n+1,n}$ is an approximation of the interpolation error from t^n to t^{n+1}, $\tilde{e}$ and $\left|\tilde{t}\right|$ are the target error at the temporal edge $t^n t^{n+1}$ and its associated edge length.

Let u^i_{n-1}, u^i_n and u^i_{n+1} be the solutions at node i and times $n-1$, n, and $n+1$, respectively. Using a forward difference approximation, we have that $u_{n+1} - u_n = \dot{u}_n \Delta t_n$ and $u_{n-1} - u_n = -\dot{u}_{n-1}\Delta t_{n-1}$. Then applying the recovery gradient in $1D$, we get:

$$\dot{u}_n = \frac{u^i_{n,n+1}\Delta t_n + u^i_{n,n-1}\Delta t_{n-1}}{\Delta t_n^2 + \Delta t_{n-1}^2}$$

and the quadratic interpolation error:

$$e^i_{n,n-1} = \dot{u}^i_{n,n-1}\Delta t_{n-1}$$

and

$$\widetilde{e}^i_{n,n-1} = {\tau^i_{n-1}}^2 \dot{u}^i_{n,n-1}\Delta t_{n-1}$$

where $\dot{u}^i_{n,n-1} = \dot{u}^i_n - \dot{u}^i_{n-1}$. Now using the equidistribution error argument, we write

$$\tilde{e}^i_{n,n-1} = e_n(N)$$

with $e_n(N)$ being the maximal error in space for a total number of nodes N. Hence the stretching factor of the time-step size is given by:

$$\tau^i_{n-1} = \left(\frac{e(N,t_n)}{e^i_{n,n-1}}\right)^{\frac{1}{2}}$$

and the optimal time-step is determined by:

$$\widetilde{\Delta t_n} = \min_i \tau^i_{n-1}\Delta t_n \qquad (9)$$

Looking closely at this formula, we notice that it requires the solution at time t^{n+1} which is not yet computed. Therefore instead of computing the optimal time-step $\widetilde{\Delta t^n}$ we calculate $\widetilde{\Delta t^{n-1}}$ and we let $\Delta t^n = \widetilde{\Delta t^{n-1}}$.

3.1 Application to 3D heat transfer and turbulent flow inside an industrial furnace

In this section, we will apply the classical space and time-adaptive methods to simulate the heat transfer and fluid flows inside an industrial furnace with complex geometry. The objective of this test case is to show the applicability and the potential of the developed algorithm in simulating long time heating inside large scale furnaces.

The furnace is modelled as a hexagonal section duct of $2.7 \times 8.1 \times 5.3m^3$ forming one heat transfer zone. All computations have been conducted by starting with a gas at rest with a constant temperature of $1463°C$. Adiabatic temperature is considered at all other boundaries for sake of simplicity. The heated air is pumped into the furnace at a velocity $14.3m/s$ by a circular burner with $6m$ diameter and located at the left vertical wall. The air is vented out of the furnace through two outlets positioned at the bottom of the left vertical wall. The 3D computations aim at simulating an hour of heating and have been conducted in parallel on 16, $2.4Ghz$ Opteron cores.

Figure 3 (top) shows the isothermal distribution at different time-steps. When the hot fluid spreads along the volume of the furnace, it induces a turbulent motion within the geometry. This forced convection is caused by the interaction of the moving stream and the

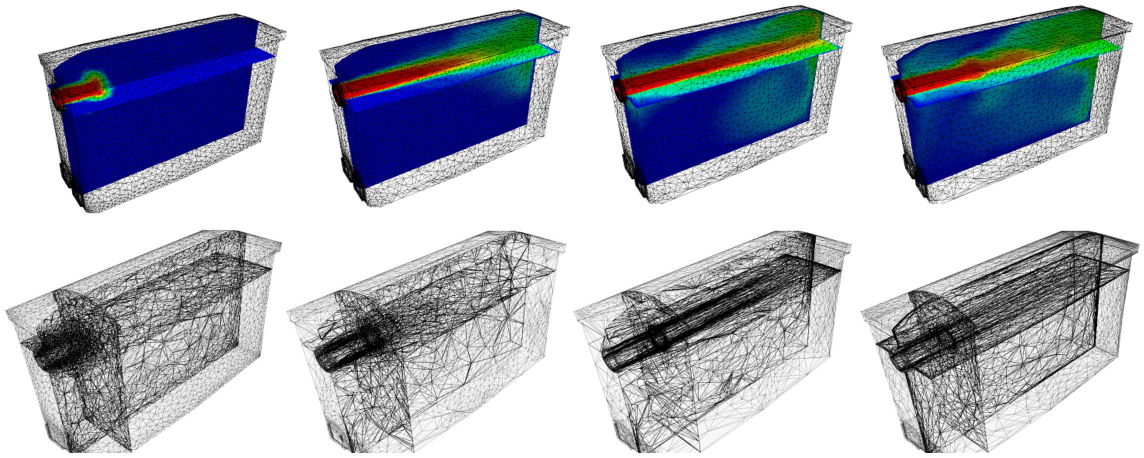

Figure 3: Isotherms (Top) and corresponding adapted meshes (bottom) at three different time-steps.

	CPU Time (s)
Non-Adaptive $\Delta t = 0.005s$	8,640,000
Adaptive Δt	172,800
Ratio	50

Table 1: CPU time for computing the solution with non-adaptive and a space/time adaptive methods

stationary fluid inside the furnace. The numerically obtained temperature distribution (fig. 3) clearly reflects the expected flow pattern. A number of small vortices inside different buffer zones can be observed. The latter are due to the turbulence dissipation and mixing of the hot and cold air. The numerical results obtained with our space and time adaptive algorithm reflect well the efficiency and potential of the methods. Figure 3 (bottom) highlights how well the mesh is adapted to sharply capture, with highly stretched elements, the gradients of the solution, the boundary layers and the emerging vortices. The algorithm builds up the mesh in a way to maximize the accuracy of the numerical solution with a fixed number of nodes ($100,000$). Note that the mesh is adapted according to the velocity components and its norm as in this test case it is the motor of the induced airflow and spread of the temperature. The results describing only one hour of the heating process required 100 days of computations with a fixed time-step equal to $0.005s$. Significant CPU time and computational cost were saved by applying our time adaptive procedure as it required only 2 days of calculations (see table 1). Figure 4 shows the evolution of the time-step sizes allowing at the same time a certain level of accuracy and an acceleration of the computations.

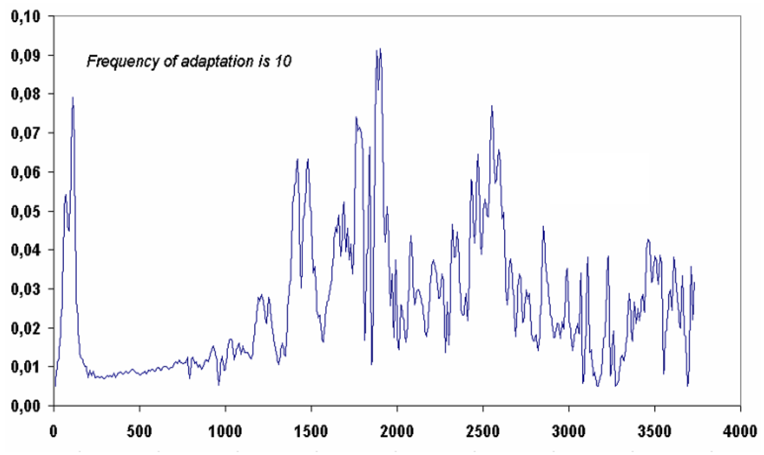

Figure 4: Time-step evolution for the simulation of an hour of heating inside an industrial furnace.

4 paradoxical meshing: Full adaptivity algorithm

When dealing with steady state problems, the classical mesh adaptation technique, presented in section 2, performs pretty well for converging the mesh-solution couple. Nevertheless, this method is no more optimal when applied to unsteady problems as the physical solution evolves in time. Together with the time-adaptive method introduced in section 3, the classical mesh adaptation can be efficiently adapted to time dependent problems.

In this paper we are interested in developing a space and time fully adaptive algorithm. The latter aims at anticipating the solution progress over a period of time and generating the optimal mesh that is adequately adapted, for a fixed number of nodes, to the evolving solution along that time interval. The analysis is carried out on a (3D+1D) mesh, i.e. the computations are performed synchronously on a 3D spatial mesh and a 1D temporal mesh. We aim at generating a mesh that holds for several solver iterations together with the corresponding optimal set of time-step sizes. Note that the user can assign a frequency of adaptation and the algorithm will accordingly adapt the meshes.

The principle consists of dividing the simulation time $[0, T]$ into n_{SI} subintervals:

$$[0,T] = [0,T^1] \cup [T^1,T^2] \cup \cdots [T^{k-1},T^k] \cup [T^k,T^{k+1}] \cup \cdots \cup [T^{n_{\mathrm{SI}}-2},T^{n_{\mathrm{SI}}-1}]$$

that will in turn be divided into $n_{\mathrm{freq}} - 1$ subintervals where n_{freq} is the frequency of adaptation assigned by the user. We call the adaptation method a paradoxical meshing as the resulting mesh is being adapted to n_{freq} time-steps while adapting every n_{freq} steps.

The mesh and the set of time-step sizes are computed through an iterative process along which we try to converge both meshes (the spatial and the temporal one) to the optimal configurations that give the most accurate solution for the corresponding interval of time. At every iteration, we consider each of the n_{SI} intervals at a time and divide it into subintervals

$$[T^k, T^{k+1}] = [T^k = t_0^k, t_1^k] \cup [t_1^k, t_2^k] \cup \cdots \cup [t_l^k, t_{l+1}^k] \cup \cdots \cup [t_{n_{\text{freq}}-2}^k, t_{n_{\text{freq}}-1}^k = T^{k+1}]$$

such that $t_{l+1}^k = t_l^k + \Delta_{k,l} = T^k + l \times \Delta_{k,l}$ and $\Delta_{k,l} = \frac{\delta_k}{n_{\text{freq}}}$.

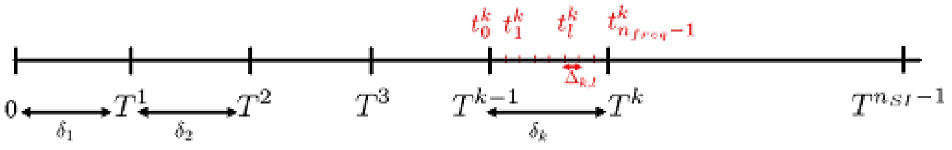

Figure 5: Temporal subintervals $[T^{k-1}, T^k]$.

The solution is predicted at each of the $\{t_l^k\}$ temporal nodes using the numerical scheme. From these solutions we construct a vector field

$$V = \{U^{k,0},\, U^{k,1},\, \cdots\,,\, U^{k,l},\, U^{k,n_{\text{freq}}-1}\}$$

and we estimate the edge based spatial errors:

$$e_{ij} = \max_{0 \le k \le n_{\text{SI}}} G_{ij}^k \cdot \mathbf{X_{ij}}$$

We compute a global error $e(N, T_k)$, as in the classical approach, for equidistributing the error on the edges of the discrete domain. An optimal metric is deduced controlling the spatial error over $[T^k, T^{k+1}]$. We also compute the temporal error at the nodes t_l^k as in section 3. Once this is done we optimize the time-step sizes $\Delta t_{k,l}$ by equidistributing the error in space and time:

$$\Delta t_{k,l} = \left(\frac{e(N, T_k)}{\max_i e_i^{k,l}} \right)^{\frac{1}{2}} \times \Delta t_{k,l}$$

An optimal size $\tilde{\delta t}_k$ of the time interval $[T^k, T^{k+1}]$ is recomputed as follows:

$$\widetilde{\delta t_k} = \sum_l \Delta t_{k,l}$$

These $\widetilde{\delta t_k}$ are given to the $1D$ mesher that will generate a new discretization of the interval $[0, T]$ as well as its corresponding temporal nodes T^k. Notice that the number of subintervals n_{SI} will be automatically changed due to this remeshing. The above described algorithm is repeated iteratively until convergence of the metrics and the set of time-step sizes. Therefore, for each interval $[T^k, T^{k+1}]$ a metric is computed accounting for the solution's transient evolution. At convergence, a mesh is generated from this metric field. Computations are then resumed on the predicted optimal set of meshes with the corresponding set of time-step sizes.

The novel method that we presented herein is perceived not only as a fully adaptive technique but also as a space and time accurate way of solving time-dependent problems within reasonable computational costs.

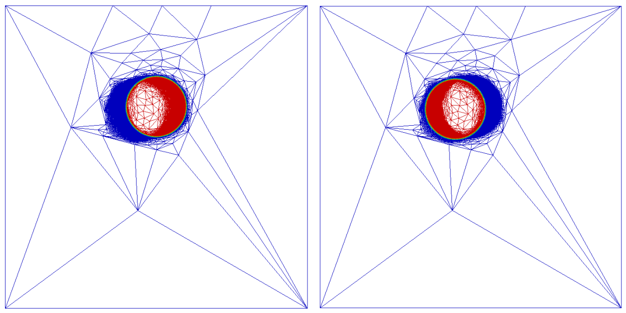

Figure 6: Paradoxical meshing algorithm applied to a rotating circle for the interval $[t^n, t^{n+1}]$.

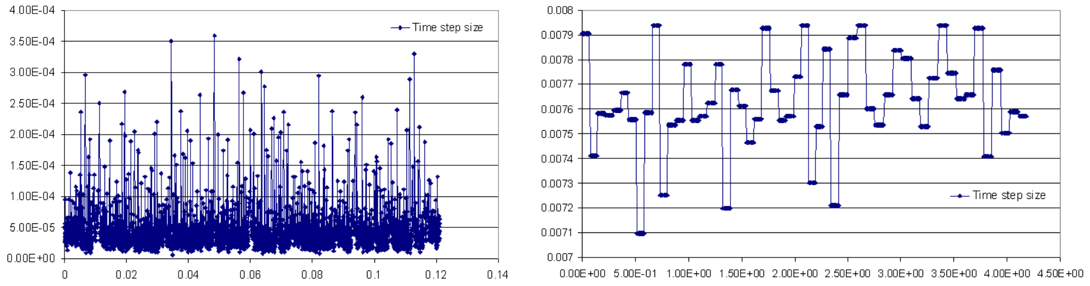

Figure 7: Time-steps generated by the classical (left) and the paradoxical (right) meshing algorithm.

5 Numerical example: Rotating Circle

In this section, we assess the performance of the newly developed fully-adaptive algorithm on a numerical test case and compare the result with the classical space and time adaptive techniques. We consider a 2D circle of radius 0.3 located at $(0.5, 0.5)$ in the computational domain $[-1.5, 1.5] \times [-1.5, 1.5]$. The simulation consists of rotating the circle in the counterclockwise direction at the rate $\theta = 1rad/s$. The objective is to test the capability of the anisotropic paradoxical meshing technique to accurately capture the dynamically evolving interface. Figure 6 shows the adapted mesh, made up of $10,000$ nodes, for the time interval $[t^n, t^{n+1}]$ made up of 10 sub-intervals of time. We can clearly see how refined the mesh is at the location of high gradients of the solution and how accurate is the capture of the interface as it rotates from t^n(left) to time t^{n+1}(right). The elements all along the interface are isotropic yielding a well respected curvature. The time intervals' lengths $[t^n, t^{n+1}] = [t^n_0, t^n_1, \cdots, t^n_i, \cdots t^{n+1}_{10}]$ are automatically generated to guarantee the validity of the mesh for 10 consecutive time-steps. Figure 7 presents the time-step sizes for the first few iterations of the algorithm. The periodic variation of the time-steps is in good agreement with the nature of the problem, as the circle rotates at a constant rate and maintains the same behavior all over computations.

Using the classical mesh adaptation with the time adaptive technique and adapting the mesh every 10 iterations, the generated time-step sizes will be too small, as seen in figure

7(left), in order to reduce the temporal error, preventing the progress of the solution in time so that it remains in phase with the adapted mesh. Recall that the method aims at giving a better efficiency than the classical algorithm. This is exactly what we notice when comparing figures 7(left) and 7(right); the paradoxical meshing technique produces time-step sizes that are almost 200 times larger than those generated by the classical algorithm. Hence, the resulting computations will be about 200 times faster reflecting the high efficiency of the novel method. Note that the inner loop of the algorithm is repeated only two times to get the optimal meshes and time-step size for which the solution remains bounded.

6 CONCLUSIONS

In this paper, we have presented a classical anisotropic mesh adaptation that showed very good performance when applied together with the new time adaptive technique for resolving time dependent problems. An extension of these algorithms lead to a novel and very powerful method for full adaptation known as the paradoxical meshing. This method demonstrated its efficiency in generating meshes and time-step sizes that guarantee the convergence of the solution all over computations for a limited number of nodes and a fixed frequency of adaptation.

Acknowledgement: The authors gratefully acknowledge the support from the “ANR: Agence nationale de la recherche”.

REFERENCES

[1] T. Coupez, *Metric construction by length distribution tensor and edge based error for anisotropic adaptive meshing*, Journal of Computational Physics, vol. 230, (2011), 2391-2405.

[2] T. Coupez and G. Jannoun and N. Nassif and H.C. Nguyen and H. Digonnet and E. Hachem,*Adaptive Time-step with Anisotropic Meshing for Incompressible Flows*, Submitted to Journal of Computational Physics, (2012).

[3] E. Hachem and B. Rivaux and T. Klozcko and H. Digonnet and T. Coupez, *Stabilized finite element method for incompressible flows with high Reynolds number*, Journal of Computational Physics, Vol. 229, (2010), *23*: 8643-8665.