View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by UPCommons. Portal del coneixement obert de la UPC

Congreso de Métodos Numéricos en Ingeniería
25-28 junio 2013, Bilbao, España
© SEMNI, 2013

SIZE PRESERVING MESH GENERATION IN ADAPTIVITY PROCESSES

Eloi Ruiz-Gironés[1], Xevi Roca[2] and Josep Sarrate[1]

1: Laboratori de Càlcul Numèric (LaCàN),
Departament de Matemàtica Aplicada III (MA III),
Universitat Politècnica de Catalunya (UPC),
Campus Norte UPC, 08034 Barcelona, Spain.
e-mail: {eloi.ruiz, jose.sarrate}@upc.edu

2: Aerospace Computational Design Laboratory,
Department of Aeronautics and Astronautics,
Massachusetts Institute of Technology, Cambridge, MA 02139, USA.
e-mail: xeviroca@mit.edu

Keywords: Mesh size function, background mesh, adaptive process, quadrilateral mesh, gradient-limiting

Abstract. *It is well known that the variations of the element size have to be controlled in order to generate a high-quality mesh. Hence, several techniques have been developed to limit the gradient of the element size. Although these methods allow generating high-quality meshes, the obtained discretizations do not always reproduce the prescribed size function. Specifically, small elements may not be generated in a region where small element size is prescribed. This is critical for many practical simulations, where small elements are needed to reduce the error of the numerical simulation.*

To solve this issue, we present the novel size-preserving technique to control the mesh size function prescribed at the vertices of a background mesh. The result is a new size function that ensures a high-quality mesh with all the elements smaller or equal to the prescribed element size. That is, we ensure that the new mesh handles at least one element of the correct size at each local minima of the size function. In addition, the gradient of the size function is limited to obtain a high-quality mesh.

Two direct applications are presented. First, we show that we can reduce the number of iterations to converge an adaptive process, since we do not need additional iterations to generate a valid mesh. Second, the size-preserving approach allows to generate quadrilateral meshes that correctly preserves the prescribed element size.

1 INTRODUCTION

In several numerical simulations, such as adaptive processes, it is of the major importance to generate a mesh that correctly preserves the prescribed element size. On the one hand, the element size is directly related to the error of the final results. On the other hand, the element size influences on the computational costs to obtain the final solution. For this reason, the final mesh has to correctly reproduce the prescribed element size. One of the most used techniques to prescribe the element size consists on assigning scalar values at the nodes of a mesh and then, interpolate these values are over the whole domain. For instance, this technique is used in adaptive simulations, where starting with an initial mesh, a size function is deduced from the computed solution via an error estimate. Then, this mesh is used as a background mesh to generate a new spatial discretization.

The size function has to verify certain requirements in order to generate a high-quality mesh that reproduces the size function. Therefore, special effort has been focused on generating element sizes fields that facilitates the generation of the desired mesh, [1, 2, 3]. Current techniques, such as the gradient-limiting algorithms [4, 5, 6, 7], modify a given size function by limiting its gradient. Thus, ratio of neighbouring element sizes is bounded and it is easier for any mesh generator to provide a high-quality mesh with smooth variation of the element size. However, the final mesh may still not correctly reproduce the initial size function. Thus, it is still necessary to develop new techniques that, given a prescribed element size field, modify it in such a way that a mesh generation algorithm can provide a discretization that verifies the initial size field.

The main contribution of this work is to introduce the new concept of size-preserving size function. The main idea behind this concept is to compute a new size function by solving an implicit and non-linear equation such that: 1. it ensures that all the elements are smaller or equal to the prescribed element size; and 2. the maximum gradient of the new size function is limited. In addition, we propose an implementation to solve the implicit and non-linear equation that states a size-preserving size function.

The new size-preserving size function has several advantages. For instance, quadrilateral and hexahedral mesh generators benefit from this new size function since these types of meshes are more difficult to refine or coarse. In addition, the new size function can reduce the number of iterations needed to converge an adaptive process, since we do not need additional iterations to properly reproduce the size field.

The outline of this paper is the following. First, in Section 2, using a one-dimensional example we motivate this work and introduce the basic definitions. In Section 3, we deduce the size-preserving method and analyze its properties. Finally, in Section 4, several examples are presented to illustrate the capabilities of the size-preserving size function.

2 MOTIVATION

To illustrate the most common problems arising from the use of a size function in a mesh generation process, we consider the following one-dimensional size function defined

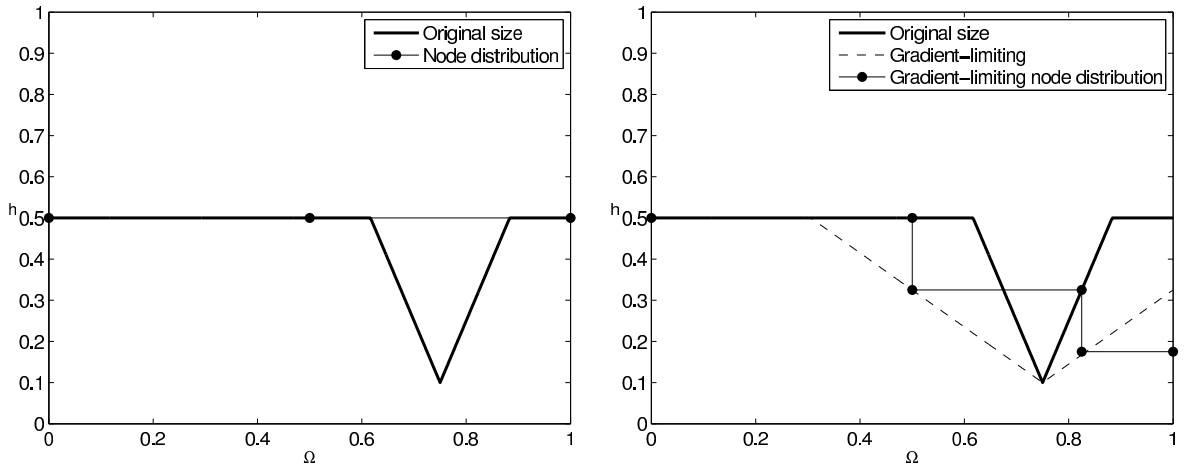


Figure 1. Node distribution using: (a) the original size function, and (b) the gradient-limiting size function.

in the $[0, 1]$ interval, see Figure 1:

$$h(x) = \min\{0.5, 0.1 + 3|x - 0.75|)\}, \tag{1}$$

Function (1) is constant almost everywhere, with a valley at $x = 0.75$. The minimum size is 0.1 and the maximum size is 0.5. Figure 1(a) presents the node distribution generated using an advancing front method where the element size is obtained using the size function (1). Since the size prescribed at $x = 0$ is 0.5, only two elements are generated and the discretization does not preserve the prescribed element size. The main reason of this shortcoming is that the size function contains high gradients that the meshing algorithm cannot reproduce.

To obtain a better discretization, a gradient-limiting technique [4, 5, 6, 7] can be applied. Figure 1(b) presents the gradient-limiting size function obtained from (1), and the node distribution obtained using the same meshing algorithm. The maximum gradient of the new size function is imposed to be $\varepsilon = 1$. Note that this node distribution does not correctly reproduce the prescribed element size. Although the element size is smaller around the valley of the size function, the minimum element size is not captured in the final mesh. To compare these discretizations we have to measure how accurately a mesh reproduces a prescribed size function. To this end, we introduce the following definitions.

Definition 1. *An element e reproduces a prescribed size function $h(x)$ if it verifies*

$$\mu(e) \leq \beta \min_{x \in e} h(x), \tag{2}$$

where $\mu(e)$ is the size of element e, and β is a scaling factor.

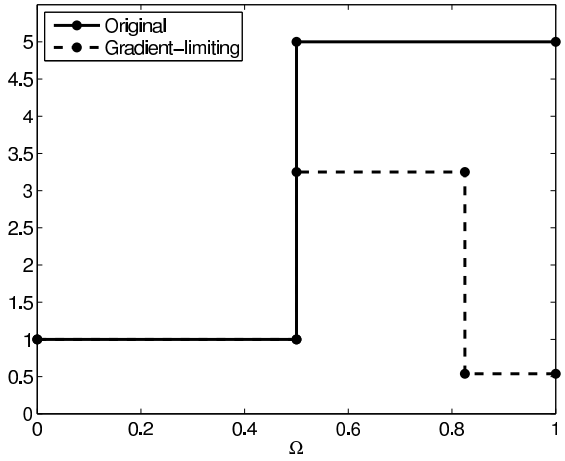


Figure 2. Ratio $R(e)$ for mesh computed using: the original size function (solid line) and the gradient-limiting size function (dashed line).

Definition 2. *A mesh $\mathcal{M}$ reproduces a prescribed size function $h(x)$ if it verifies*

$$\mu(e) \leq \beta \min_{x \in e} h(x), \qquad \forall e \in \mathcal{M}. \tag{3}$$

By introducing the ratio

$$R(e) = \frac{\mu(e)}{\beta \min_{x \in e} h(x)}, \tag{4}$$

a valid mesh has to verify

$$R(e) \leq 1 \qquad \forall e \in \mathcal{M}. \tag{5}$$

Figure 2 plots function (4) for the discretizations generated using the original size function and the gradient-limiting size function with $\beta = 1$. Since $R(e) \geq 1$ for all the elements of the mesh, these meshes do not reproduce the size function.

3 SIZE-PRESERVING SIZE FUNCTION

We have shown that a mesh may not reproduce the prescribed element size function. That is, it may not verify Equation (3). To overcome this drawback, in this section we introduce the new concept of *size-preserving size function*. Given the original size function, $h(x)$, we will deduce an alternative size function, called size-preserving size function and denoted by $h^*(x)$, such that it allows reproducing the size function according to Equation (3).

In fact, the new size function, $h^*(x)$, can be written in terms of the original one, $h(x)$. To this end, we consider the one-dimensional example presented in Figure 3(a). To obtain a mesh that correctly reproduces the size function, we assume that the new element size around a point $x \in \Omega$ has to be $h^*(x)$. Then, the new node is created at position $x \pm h^*(x)$,

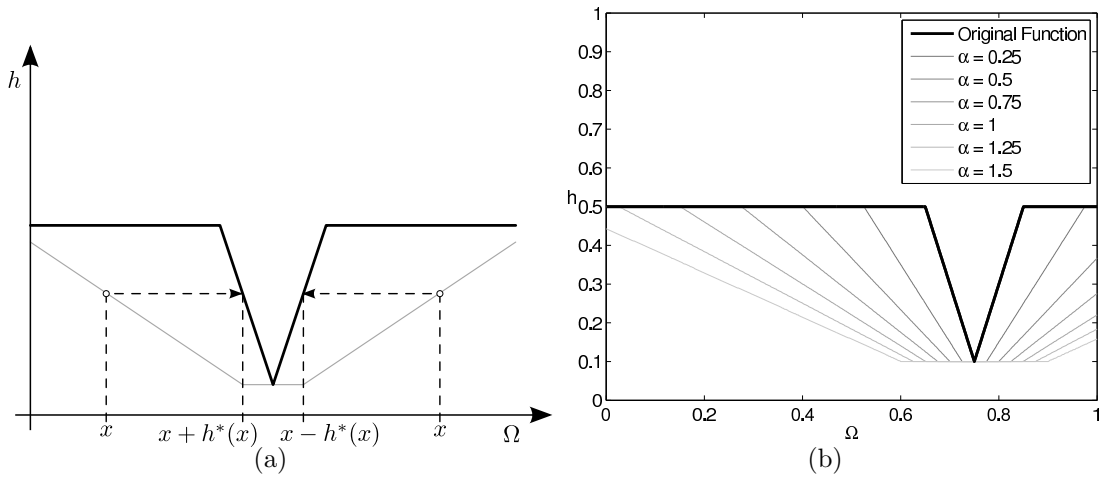


Figure 3. (a) Original size function (thick black line) and size-preserving size function (grey line), and (b) one-dimensional size function defined in the $[0, 1]$ interval (thick black line) and several size-preserving size functions for different values of α (grey lines).

depending on the advancing direction of the meshing algorithm, see Figure 3(a). For this reason, the size of the new element, e, is $\mu(e) = h^*(x)$. Taking into account condition (2), we deduce that the following equation has to be verified:

$$h^*(x) = \mu(e) \leq \beta \min_{y \in [x-h^*(x), x+h^*(x)]} h(y).$$

If we want $h^*(x)$ as big as possible to generate the minimum amount of elements, we have that

$$h^*(x) = \beta \min_{y \in [x-h^*(x), x+h^*(x)]} h(y).$$

To add more flexibility to our method, we include a parameter α that determines the trial interval (i.e. the interval in which the minimum of the original size function is computed):

$$h^*(x) = \beta \min_{y \in [x-\alpha h^*(x), x+\alpha h^*(x)]} h(y).$$

Note that taking $\alpha > 1$ we enlarge the trial interval. Thus, the size-preserving size function can achieve smaller values. On the contrary, taking $\alpha > 1$ we reduce the trial interval and the size-preserving size function can achieve larger values. The previous equation can be expressed in any dimension as:

$$h^*(\mathbf{x}) = \beta \min_{\mathbf{y} \in \mathrm{B}_{\alpha h^*(\mathbf{x})}(\mathbf{x})} h(\mathbf{y}), \tag{6}$$

where $\mathrm{B}_r(\mathbf{x})$ is the set of points at distance at most r from $\mathbf{x}$. Note that β can be interpreted as a parameter that controls the maximum ratio $R(e)$ accepted in the final

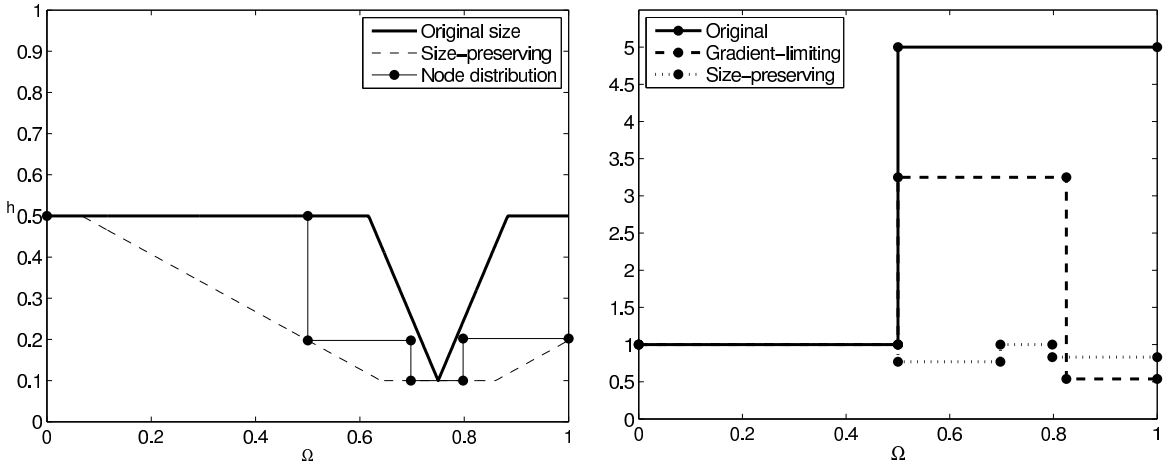


Figure 4. (a) Node distribution using the size-preserving function. (b) Ratio $R(e)$ for mesh computed using: the original size function (solid line), the gradient-limiting size function (dashed line) and the size-preserving size function (dotted line).

mesh. It is important to point out that Equation (6) is an implicit and non-linear definition of the size-preserving size function. Figure 3(b) plots several size-preserving functions for different values of parameter α. This parameter plays an important role in Equation (6). In fact, it controls:

(i) **The measure of local minima in the element size function**. This ensures that small element sizes prescribed at local minima can be correctly reproduced. That is, an element can be held at each local minima.

(ii) **The maximum gradient allowed**. This ensures that a high-quality mesh can be generated. Although we do not have a formal proof, experience has shown that the maximum gradient of the size-preserving size function is limited to β/α.

Figure 4(a) presents the original size function stated in Equation (1), the corresponding size-preserving size function, and the node distribution of the generated 1D mesh. The size-preserving size function is computed according to Equation (6) with $\alpha = 1$. Note that the region around the minimum is automatically enlarged in order to held an element of the requested size. In addition, the gradient of the size-preserving size function has been automatically limited.

Figure 4(b) presents the ratio $R(e)$, see Equation 4, for the meshes generated using: the original, the gradient-limiting and the size-preserving size functions. Note that for the size-preserving size function the distribution of nodes correctly preserves the original size function.

In order to obtain a size-preserving size function, we propose a node-by-node process in which we solve the non-linear equation defined in (6). That is, for each node in the

background mesh, we use an iterative solver to compute the value of the new size function at that node.

4 EXAMPLES

This section presents four examples in order to illustrate the behavior of the size-preserving size function. This function has been successfully implemented in the ez4u meshing environment [8, 9, 10]. The first example shows the advantages of using the size-preserving size function in an adaptive process. The second one illustrates the applicability of the proposed method to generate a one-dimensional mesh using an advancing front technique. The remaining examples deal with multi-dimensional mesh generation problems. The third example is a 2D applications where quadrilateral meshes are generated using the algorithm presented in [11]. The fourth example shows the mesh generated for a complex 2D size function defined using an MRI image.

4.1 Accelerating an adaptive process

The objective of this example is to show that the proposed size-preserving approach can decrease the number of iterations of an adaptive process. To this end, we present three different executions of the adaptive process presented in Algorithm 1. We define, for all the executions of the adaptive process, the same analytical element size function:

$$h = \min\left\{0.25, |x - 0.575| + 0.25 \cdot 10^{-15}\right\}.$$

Note that minimum element size is $0.25 \cdot 10^{-15}$. From this analytical size function, we will compute a background mesh for each iteration of the adaptive process. The new background mesh is constructed from the mesh of the previous iteration. When the new background mesh is obtained, we generate the next mesh. This process is iterated until the mesh reproduces the analytical size function with a relative error below 0.05. That is, we accept all the elements whose length is, at most, 5% above the prescribed size of the analytical size function.

Algorithm 1 Adaptive process

Ensure: Mesh $\mathcal{M}$

```
1: function AdaptiveProcess
2:     Mesh M ← createUniformMesh
3:     BackgroundMesh bm ← estimateNewElementSize(M,sizeFunction)
4:     Boolean converged ← checkConvergence(M,bm)
5:     while not converged do
6:         M ← createNewMesh(M,bm)
7:         BackgroundMesh bm ← estimateNewElementSize(M,sizeFunction)
8:         Boolean converged ← checkConvergence(M,bm)
9:     end while
10: end function
```

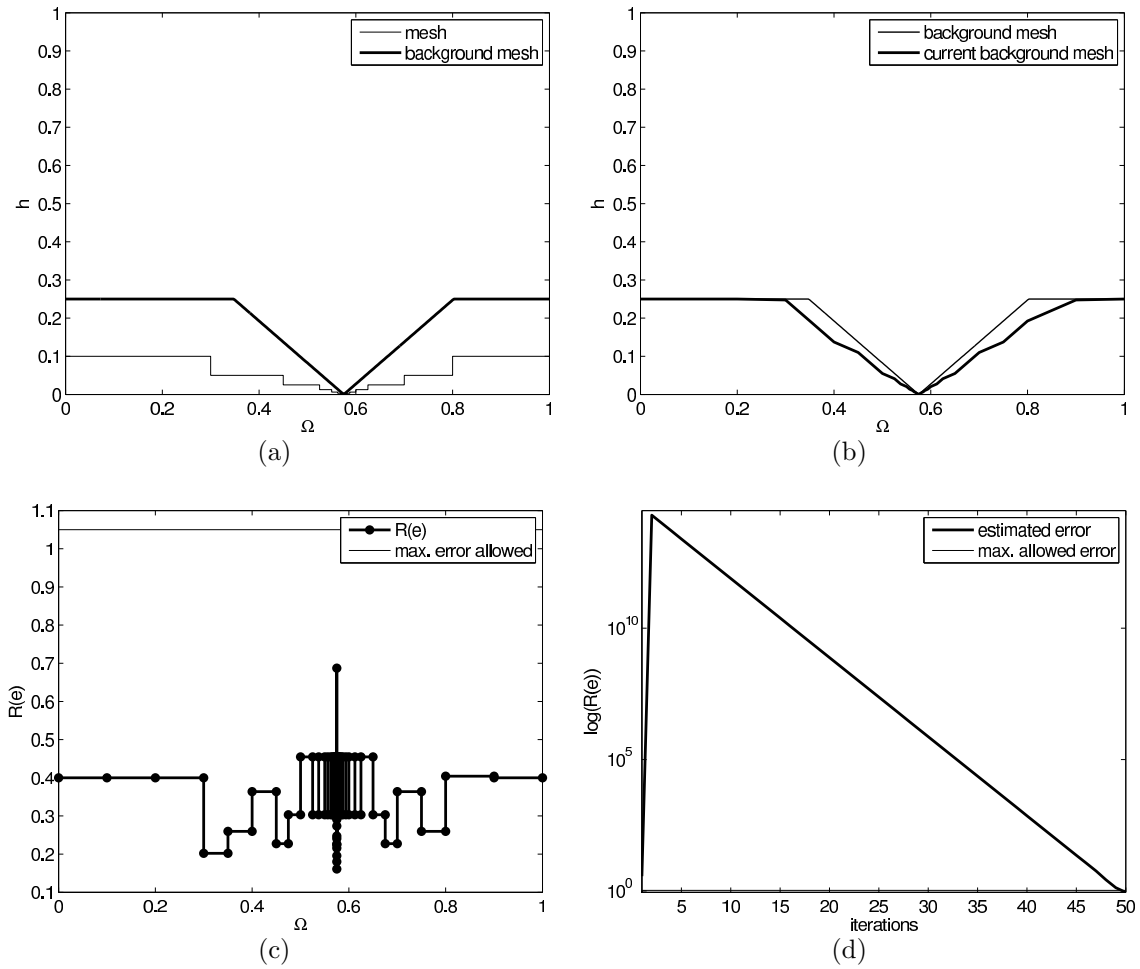


Figure 5. Adaptive process, converged in 50 iterations, without processing the background mesh: (a) final mesh; (b) background mesh for the last iteration; (c) ratio $R(e)$ for each element of the final mesh, and (d) estimated error for each iteration.

In the first execution, we do not process the size field with any method. In addition, the mesh generation function is defined as a mid-point subdivision of the elements that do not satisfy the prescribed element size. The process takes 50 iterations to generate the final mesh, see Figure 5(a). Figure 5(b) shows the background mesh of the last iteration. Note that the final mesh correctly reproduces the prescribed size function. In Figure 5(c) we present the ratio $R(e)$. Since the mesh reproduces the size function, all the elements have a ratio below the desired relative error. Finally, Figure 5(d) presents the logarithm of the maximum ratio $R(e)$ for each iteration. Note that the algorithm divides each element in half and, for this reason, the logarithm of $R(e)$ decreases linearly. Thus, the number of iterations is proportional to the ratio between the element size of the initial mesh and the desired element size.

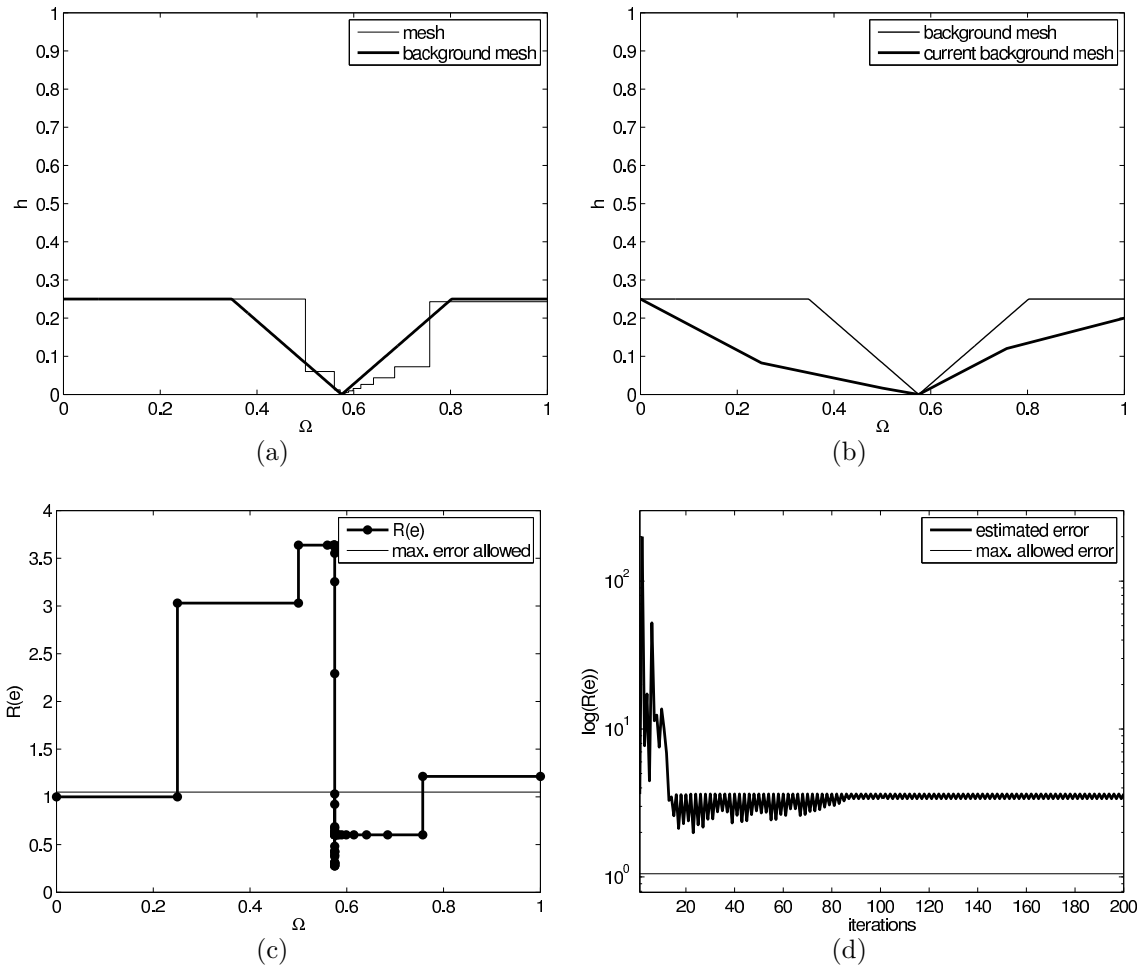


Figure 6. Adaptive process, not converged in 200 iterations, using the gradient-limiting technique: (a) final mesh; (b) background mesh for the last iteration; (c) ratio $R(e)$ for each element of the final mesh, and (d) estimated error for each iteration.

In the second execution, we process the background mesh using a gradient limiting technique, $\varepsilon = 0.8$. The mesh for each iteration is generated using an advancing front technique. In this example, the adaptive process has not converged in 200 iterations. Figures 6(a) and 6(b), show the mesh and the background mesh for the las iteration, respectively. Figure 6(c) presents the $R(e)$ ratio for each element. Note that in the region where a small size has been prescribed, the elements are almost four times bigger than the requested size. Finally, Figure 6(d) shows the logarithm of the $R(e)$ ratio for each iteration. Note that the process is not able to converge to the required mesh. The main reason is that it is not possible to guarantee that, at each iteration, the size field on the background mesh is correctly preserved.

In the last execution, we have processed the background mesh using the proposed size-

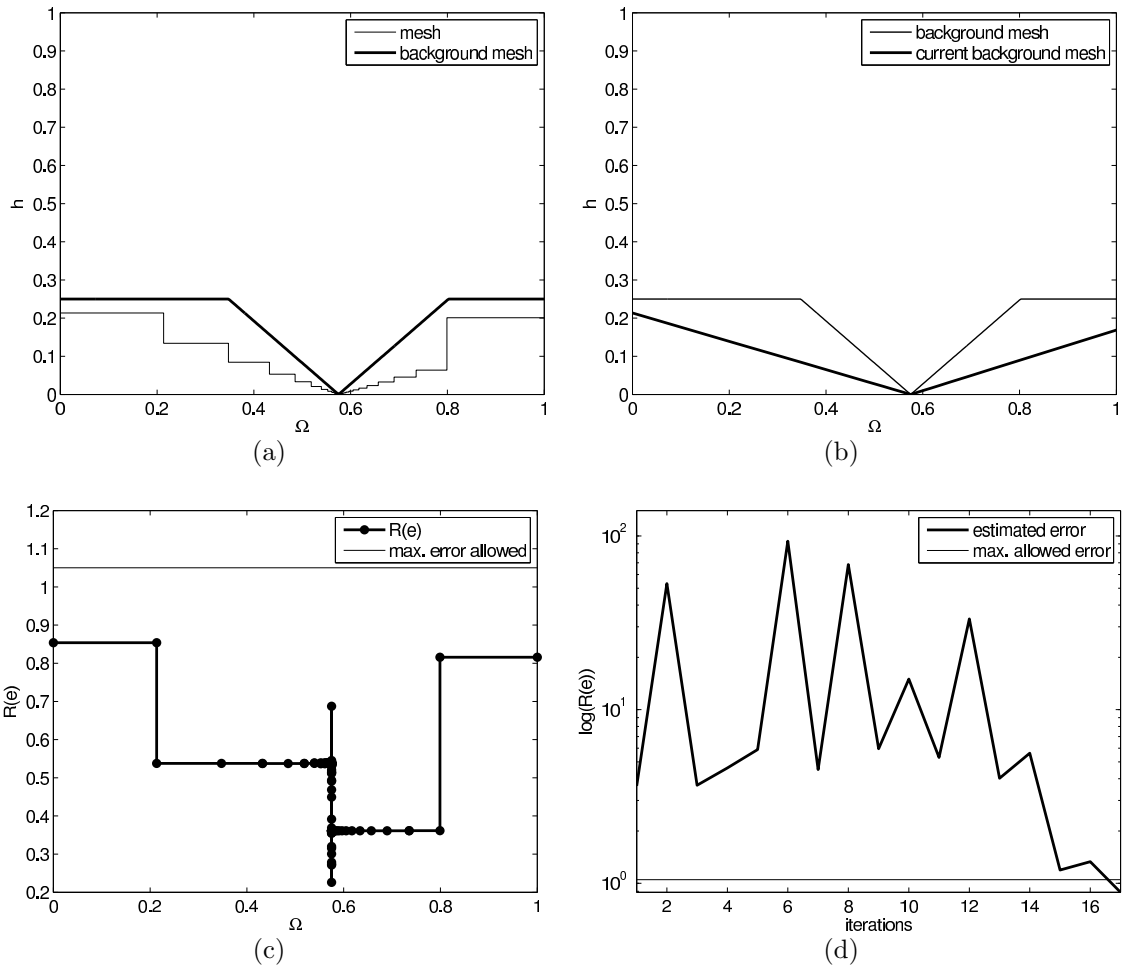


Figure 7. Adaptive process, converged in 17 iterations, using the size-preserving technique: (a) final mesh; (b) background mesh for the last iteration; (c) ratio $R(e)$ for each element of the final mesh, and (d) estimated error for each iteration.

preserving algorithm with $\alpha = 1.25$. In each iteration, the whole mesh is re-generated using an advancing front method. Figures 7(a) and 7(b) show the generated mesh and the used background mesh at the last iteration, respectively. Figure 7(c) shows the $R(e)$ ratio for each element of the final mesh. Note that the mesh correctly reproduces the analytical size function. Finally, Figure 7(d) shows the logarithm of the $R(e)$ ratio for each iteration. Note that the adaptive process only takes 17 iterations to converge because at each iteration, the used background mesh is correctly reproduced.

4.2 Preserving a 1-D size function in a mesh generation process

In this example we present a sinusoidal size function defined in the $[0, 1]$ interval. The minimum element size is 0.1 and maximum element size is 1.0. In Figure 8 we represent

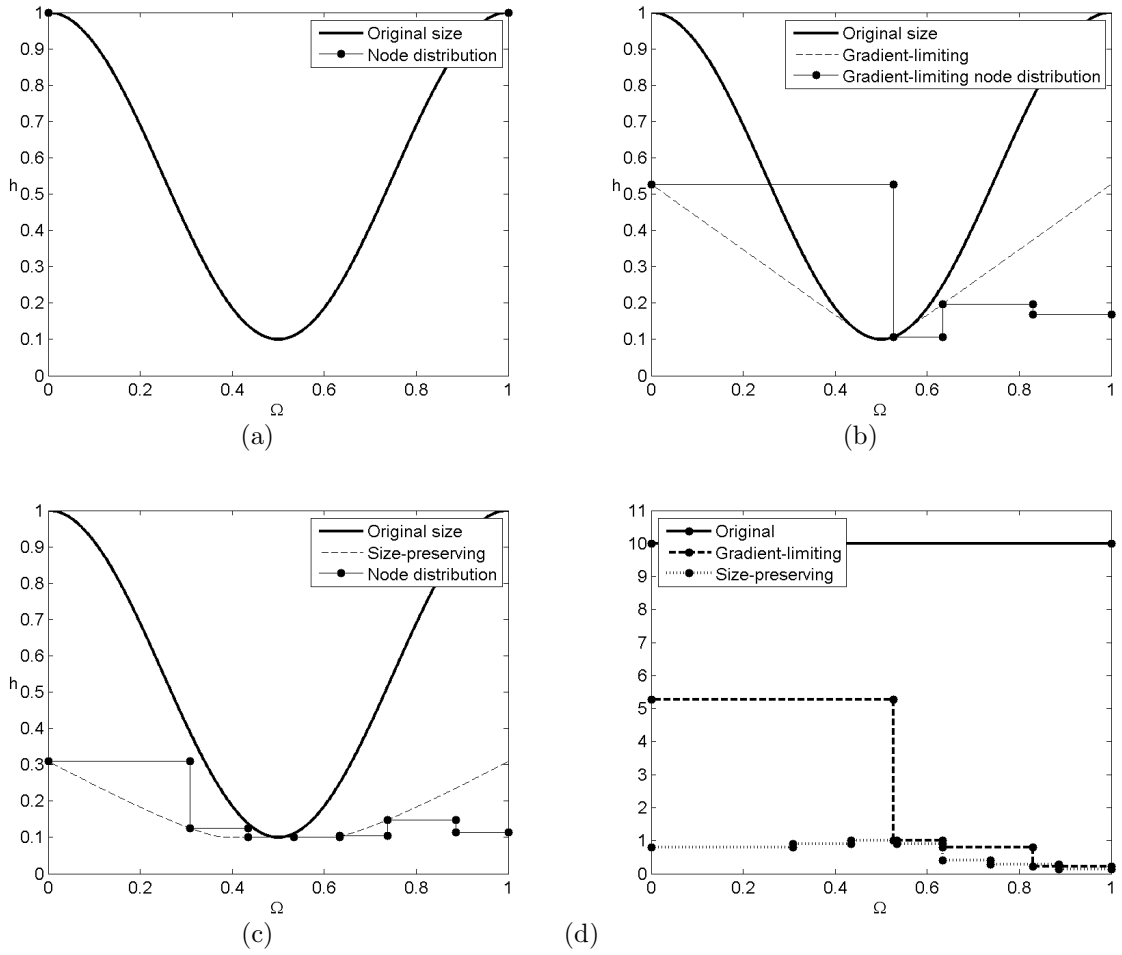


Figure 8. Node distributions for the first example using: (a) the original size function; (b) the gradient-limiting size function; (c) the size-preserving size function; and (d) ratio $R(e)$ for mesh computed using: the original size function (solid line), the gradient-limiting size function (dashed line), and the size-preserving size function (dotted line).

the original size function with a thick solid line, the modified size functions with dashed lines and the corresponding node distributions with thin solid lines.

Figure 8(a) plots the original size function and the node distribution obtained using it. Note that the final mesh does not reproduce the prescribed element size. Figure 8(b) displays the gradient-limiting size function (using $\varepsilon = 1$), and the corresponding node distribution. Although the mesh better reproduces the element size function, it does not generate sufficiently small elements around the minima points of the size function. To generate a mesh that correctly preserves the element size, we compute a size-preserving function with $\alpha = 1$), Figure 8(c). Note that the size-preserving function presents two features. First, it increases the width of the areas around small element sizes. Second,

the gradient of the new size function is limited. These two features allow the generation of a high-quality mesh that correctly reproduces the original size function and captures the minimum element sizes.

In Figure 8(d), we present the ratio $R(e)$ for the mesh generated using: the original, the gradient-limiting, the size-preserving, and the approximated size-preserving size functions. Note that only the mesh generated using the size-preserving size function verifies Equation (5) and correctly preserves the element size. The meshes obtained using the original and the gradient-limiting techniques do not preserve the element size.

4.3 Preserving a size function in quadrilateral mesh generation

2-D sinusoidal size function.

In this example we show the meshes generated using a two-dimensional background mesh defined on the $[0,1] \times [0,1]$ square. The element size is defined as a sinusoidal function with 4 periods in which the minimum and maximum element sizes are 0.01 and 1.0 respectively, see Figure 9(a). Note that in all the generated meshes, we used comparable values for the parameters of the different methods where it was possible. That is, $\varepsilon = 1/\alpha$ and $\delta = 2\alpha$.

Figure 9(b) shows the mesh generated using a gradient limiting technique ($\varepsilon = 0.5$). The colors of each element are the ratio of the actual element size and the prescribed element size of the original size function. Note that there are elements that are 57% larger than the prescribed element size. From a numerical point of view, these elements may not be acceptable.

In order to obtain a mesh that preserves the element size, we use a size-preserving size function, with $\alpha = 2$. Figure 9(c) shows the mesh generated using it. In this case, the final mesh correctly captures all the features of the initial size function. That is, the areas where a small element size is prescribed are correctly captured in the final mesh.

Preserving a field defined by MRI image. In this example, we present a quadrilateral mesh generated using a size field derived from a MRI image, courtesy of the OASIS project [12], see Figure 10(a). The size field is defined in terms of the mean curvature of the MRI field, in order to generate more elements where the variation of the gradient of the MRI field is higher, see Figure 10(b). With this background mesh, we have generated two meshes. In each mesh, we have computed the interpolation error of the initial MRI field on the corresponding mesh, and the ratio $R(e)$ for the elements. The first mesh is obtained with the gradient-limiting technique. Figure 11(a) shows the interpolation error of the MRI field on the mesh, and Figure 11(b) presents the ratio $R(e)$ for the elements. Note that there are elements that are almost 75% bigger that the requested size. The second mesh is obtained using the size-preserving approach. In this case, the results present a reduced interpolation error (Figure 12(a)), and ratio $R(e)$ (Figure 12(b)).

Table 1 summarizes the statistics for the meshes generated with the different size functions. The behavior of the size functions is similar to the ones reported in the previous example. The gradient-limiting technique, although fast is not able to reproduce the initial

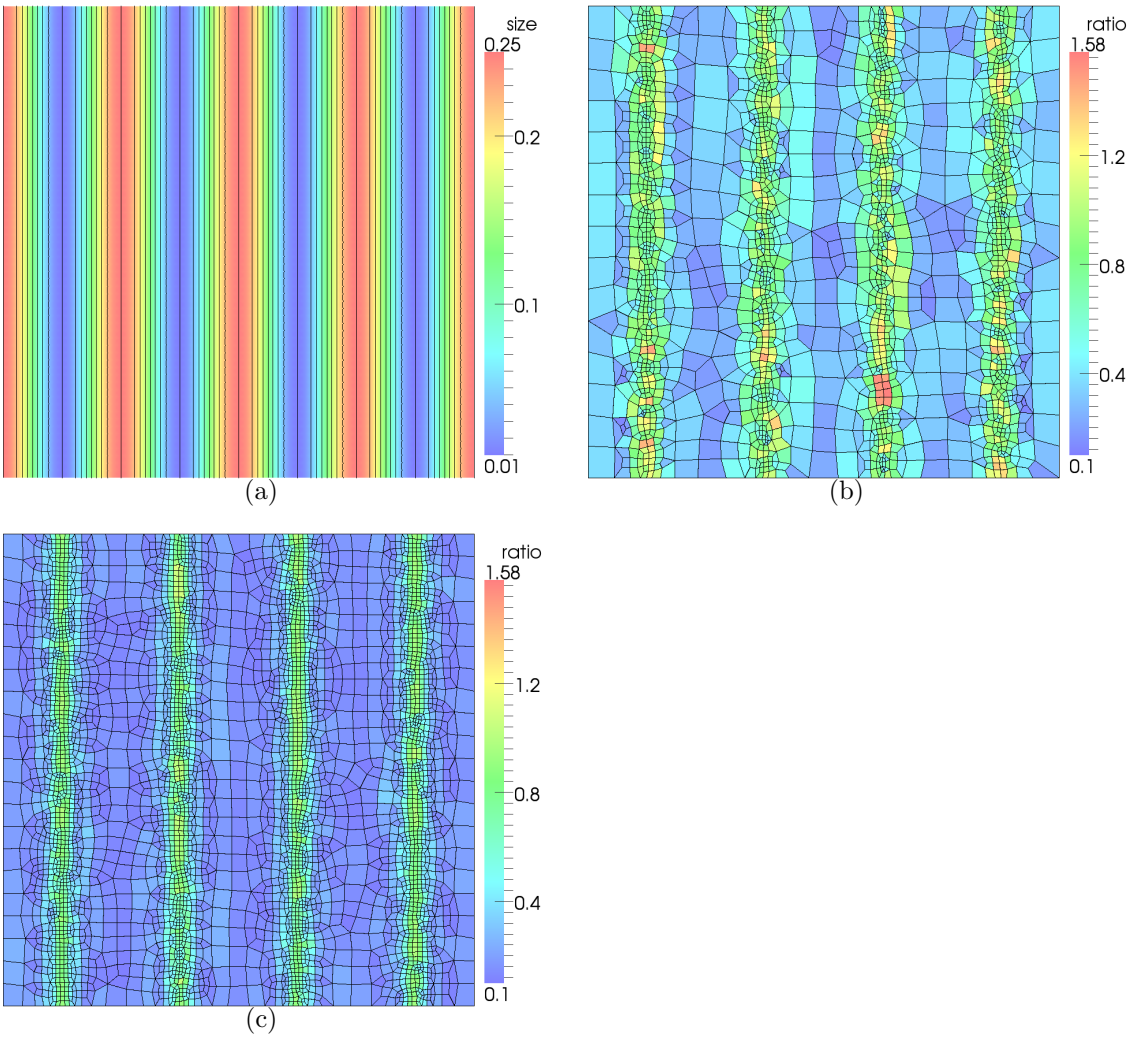
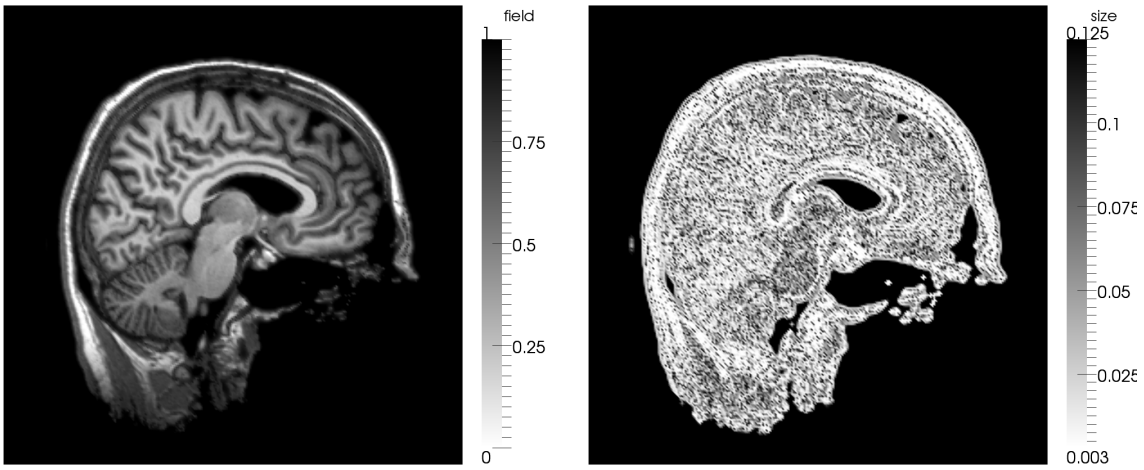


Figure 9. (a) Original size function of the second example. Meshes generated using: (b) a gradient limiting technique; and (c) a size-preserving function.

size function, obtaining the highest ratio $R(e)$ and the lowest percentage of correct faces. Specifically, there are elements that are almost 75% percent bigger than the requested size. The mesh obtained with the size-preserving approach presents the lowest ratio $R(e)$ and the maximum percentage of correct faces. In this case, 99.99% percent of the faces are correct, and the maximum ratio $R(e)$ is around 1.03.

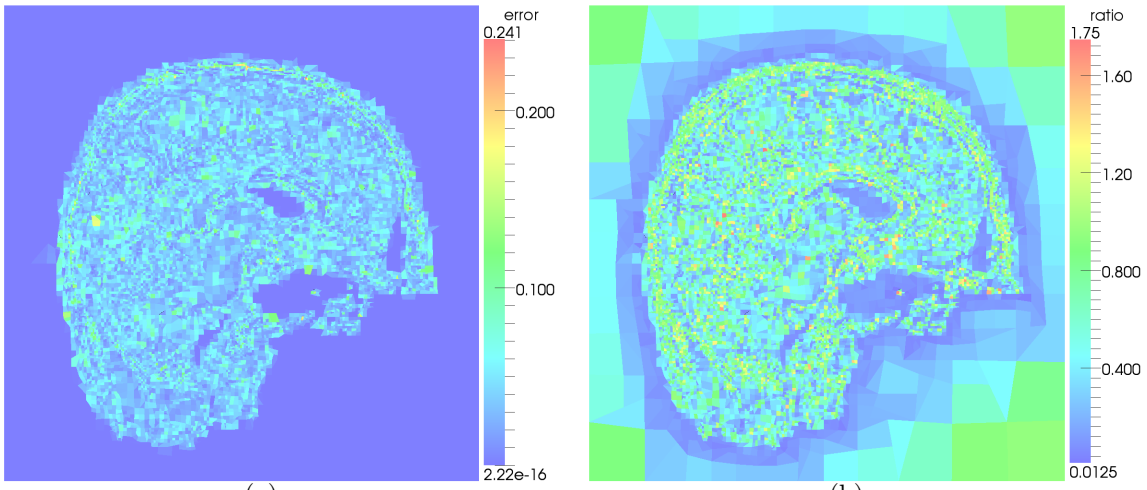
5 CONCLUSIONS

In this paper, we have presented the new size-preserving method for computing good element size functions. It modifies the original size function, and allow any mesh generation algorithm to generate meshes that preserve the original size function. For this



(a) (b)

Figure 10. (a) MRI field defined on a square and (b) its associated size function.



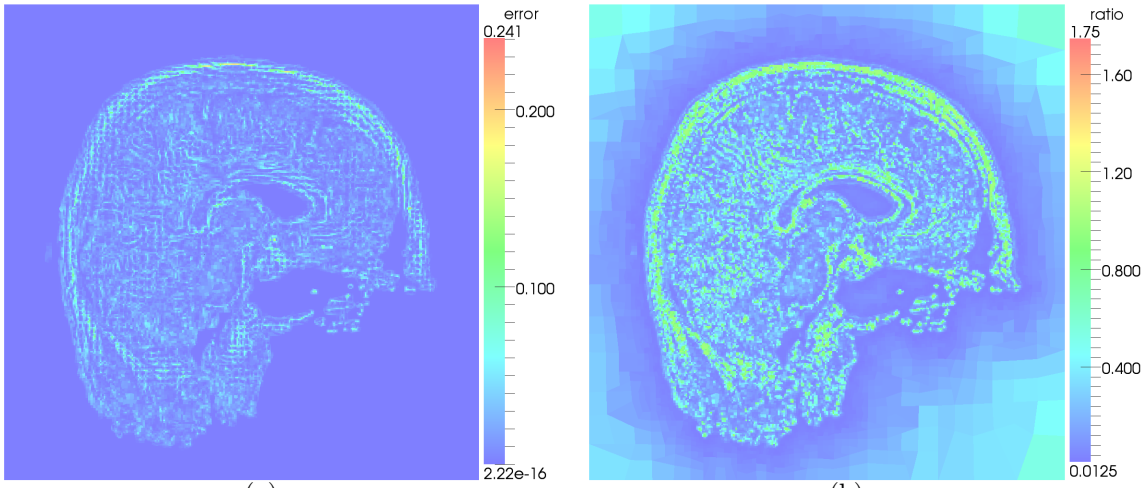
(a) (b)

Figure 11. Mesh generated using the gradient-limiting technique: (a) interpolation error and; (b) ratio $R(e)$ of the elements.

Table 1. Statistics for the meshes generated for MRI field.

method	total faces	correct faces	correct faces (%)	maximum ratio $R(e)$	Time compared to gradient-limiting
gradient-limiting	14089	10402	73.83%	1.746	1
size-preserving	42932	42543	99.09%	1.049	53.2

reason, it facilitates the generation of quadrilateral and hexahedral meshes, since it is not



(a) (b)

Figure 12. Mesh generated using the size-preserving technique: (a) interpolation error and; (b) ratio $R(e)$ of the elements.

required to refine them to reproduce the initial size function. Moreover, this method can potentially reduce the number of iterations of an adaptive process, since the prescribed size field is correctly reproduced in each iteration.

The size-preserving method computes computes the new size function by solving a non-linear equation. The proposed algorithm modifies the size function at each node of a background mesh ensuring that an element can be held in that area and, at the same time, limits the maximum gradient of the new size function. We have shown that to generate a mesh that preserves the original size function, it is recommended to use $\beta \leq 1$ and $\alpha \geq \beta$.

Several aspects of the proposed method can be extended in the near future. For instance, we use an edge-based solver to compute the new size functions. Although more expensive from the computational point of view, it could be interesting to analyze the solutions provided by Hamilton-Jacobi solvers. In addition, the ideas presented in this work can be extended to anisotropic size fields. This implies to work with an anisotropic metric, and modify some aspects of the proposed algorithms.

REFERENCES

[1] W.R. Quadros, K. Shimada, and S.J. Owen. Skeleton-based computational method for the generation of a 3d finite element mesh sizing function. *Engineering with Computers*, 20(3):249–264, 2004.

[2] W.R. Quadros, V. Vyas, M. Brewer, S.J. Owen, and K. Shimada. A computational framework for automating generation of sizing function in assembly meshing via disconnected skeletons. *Engineering with Computers*, 26(3):231–247, 2010.

[3] J. Zhu, T.D. Blacker, and R. Smith. Background overlay grid size functions. In *Proceedings of the 11th International Meshing Roundtable*, pages 65–74, 2002.

[4] M. Yerry. Modified quad-tree approach to finite element mesh generation. *IEEE Computer Graphics & Applications*, 3(1):39–46, 1983.

[5] H. Borouchaki, F. Hecht, and P.J. Frey. Mesh gradation control. *International Journal for Numerical Methods in Engineering*, 43(6):1143–1165, 1998.

[6] P.J. Frey and L. Marechal. Fast adaptive quadtree mesh generation. In *Proceedings of the 7th International Meshing Roundtable*, 1998.

[7] P.O. Persson. Mesh Size Functions for Implicit Geometries and PDE-Based Gradient Limiting. *Engineering with Computers*, 22(2):95–109, 2006.

[8] X. Roca, J. Sarrate, and E. Ruiz-Gironés. A graphical modeling and mesh generation environment for simulations based on boundary representation data. In *Congresso de Métodos Numéricos em Engenharia*, 2007.

[9] X. Roca. *Paving the path towards automatic hexahedral mesh generation.* PhD thesis, Universitat Politècnica de Catalunya, 2009.

[10] X. Roca, E. Ruiz-Gironés, and J. Sarrate. ez4u. mesh generation environment. `http://www-lacan.upc.edu/ez4u.htm`, 2010.

[11] J. Sarrate and A. Huerta. Efficient unstructured quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering*, 49:1327–1350, 2000.

[12] OASIS project. `http://www.oasis-brains.org`, 2013.