Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 203 (2017) 310–322

Procedia Engineering

www.elsevier.com/locate/procedia

26th International Meshing Roundtable, IMR26, 18-21 September 2017, Barcelona, Spain

Subdividing triangular and quadrilateral meshes in parallel to approximate curved geometries

A. Gargallo-Peiró, G. Houzeaux, X. Roca*

Computer Applications in Science and Engineering Dpt., Barcelona Supercomputing Center-Centro Nacional de Supercomputación, C/ Jordi Girona 29, Barcelona, 08034, Spain.

Abstract

A parallel distributed approach to refine a mesh while preserving the curvature of a target geometry is presented. Our approach starts by generating a coarse linear mesh of the computational domain. Second, the former coarse mesh is curved to match the curvature of the target geometry. Then, the curved mesh is partitioned and the subdomain meshes are sent to the slaves. Finally, the curved elements are uniformly subdivided in parallel targeting the geometry approximated by the curved mesh. The result is a distributed finer linear mesh featuring improved geometric accuracy. The key ingredient of our implementation is to approximate the target geometry as a linear mesh equipped with an elemental field corresponding to an element-wise polynomial geometry representation. Thus, the distribution of the curved geometry is equivalent to partitioning the linear mesh and sending the subdomain meshes and the elemental fields to the slaves. The main application of the obtained finer linear mesh is to compute in parallel steady state flow solutions on real topographies. The qualitative results show that for 2D and 3D steady state flow solutions, on real and synthetic topographies, our parallel subdivision approach mitigates the artificial artifacts that might appear with standard straight-sided subdivision methods. We also check the parallel performance of the implementation by performing a weak scalability test in 2D.

Peer-review under responsibility of the scientific committee of the 26th International Meshing Roundtable.

Keywords: Mesh subdivision, Mesh multiplication, Parallel mesh generation, Parallel computing, High-order mesh

1. Introduction

Standard commercial linear mesh generators are targeted to approximate geometries for simulation. To this end, they feature boundary layer meshing, and local adaptivity. However, they are designed to provide piecewise linear (straight-sided tetrahedra) or trilinear (ruled-sided hexahedra) approximations of the computational domain and therefore, subdivided meshes are not targeted to approximate curved boundaries. Furthermore, they are not ready to generate in parallel meshes suitable for petascale runs, mainly due to computational costs and memory constraints.

To overcome these issues, two main strategies have been proposed. On the one hand, we find the possibility of directly generating the mesh in parallel using distributed memory. However, this implies modifying each sequential

*Corresponding author.
E-mail address: xevi.roca@bsc.es

1877-7058
Peer-review under responsibility of the scientific committee of the 26th International Meshing Roundtable.
10.1016/j.proeng.2017.09.814

meshing algorithm so that it can be parallelized. Furthermore, it demands developing the parallel distributed implementation for the modified meshing scheme. Following this direction, we can find parallel distributed implementations for the Advancing Front [1–3] and Delaunay [4–6] meshing algorithms.

On the other hand, an alternative approach consists of generating a coarse mesh using standard sequential meshers, distributing the mesh and finally subdividing the elements creating new degrees of freedom in parallel in each partitioned sub-domain [7]. However, an inherent problem of subdiving the mesh in parallel is to recover the geometry in the new nodes on the boundaries, and, in addition, obtain a valid final mesh. To overcome this drawback several strategies have been developed. In [8,9] the new mid-nodes are relocated to the closest node in a fine surface mesh, correcting the location of the interior nodes using a spring-analogy technique. Similarly, in [10] the new geometry nodes are projected to an STL representation of the geometry, and the interior nodes are relocated using a local mesh quality optimization process.

Regarding other coarse-to-fine paradigms, although they can not directly be considered mesh subdivision approaches, we find different sequential works to subdivide, refine or add new degrees of freedom to a given mesh. For all-hexahedral mesh scaling analysis, the work presented in [11] allows to increase the number of elements in meshes by any positive multiplier (and not only powers of 2) in domains that can be decomposed in structured or sweeping blocks. We also find approaches to accelerate the mesh curving of moving domains [12] by first generating a coarse high-order mesh and following subdividing it. Alternatively, we find a posteriori approaches to insert boundary layers in curved meshes [13], by first generating the isotropic mesh, and following inserting the boundary layer elements. Boundary layers can also be generated a posteriori by subdividing boundary prismatic elements into anisotropic tetrahedra [14,15].

In this work, we have implemented a parallel curved mesh subdivision method, by extending a parallel linear uniform subdivision method [7,16] build on top of the Alya code [17,18]. Our approach starts by generating a coarse linear mesh. Then the former coarse mesh is curved to match the boundaries of the computational domain [19,20]. Each element of the linear mesh is equipped with an elemental field that corresponds to an element-wise representation of the curved geometry approximated by the curved mesh. Following, the linear mesh and its elemental field are partitioned and the resulting subdomain meshes are sent to the slaves. The subdomain meshes are subdivided targeting the elemental geometry, and the result is a finer linear mesh obtained in parallel with improved geometric accuracy. Then, a steady state flow solution can be obtained in parallel on the finer and already distributed linear mesh.

The advantage of the proposed and implemented strategy is that the subdivision is performed and stored in parallel and therefore, there are no memory constraints. Furthermore, the finer linear mesh targets the curvature information described by the curved mesh. Finally, if necessary, the post-processing can be performed on the coarse mesh. From the implementation point of view, we find an additional advantage of considering the geometry element-wise. A code featuring linear (or bi-linear elements), only needs to be modified in the subdivision process, since the geometry is from the implementation point of view an elemental field. Hence, all the data structures of the existent code can be re-used to partition and send to the subdomains the additional created data to reproduce the geometry. There is no need to generate new code to partition or distribute the geometry or to project nodes to recover the geometry. Each element has its own geometry, which is known a priori, and each elemental geometry is partitioned and distributed together with its corresponding linear element.

The rest of the paper is organized as follows. First, in Sec. 2 we present the proposed curved mesh subdivision scheme. Following, we detail the parallel implementation of the proposed approach, Sec. 3. Finally, in Sec. 4 we present the application of the mesh subdivision process to a 2D topographical scenario, analyzing the differences on the velocity of an Atmospheric Boundary Layer flow simulation, and comparing the results to the standard straight-sided mesh subdivision process. In addition, we study the mesh scaling of the curved subdivision process and we present some preliminary results of the extension of the technique to 3D meshes.

2. Mesh subdivision with curvature

In this section, we present the main steps and implementation details of our new method to refine a mesh in parallel while preserving the curvature of a target geometry. Our approach consists of two main steps. First, in Sec. 2.1, we generate an initial coarse linear mesh and a high-order curved mesh that will be used as an element-wise approximation of the geometry to recover when subdividing the linear mesh. Following, in Sec. 2.2, we present the

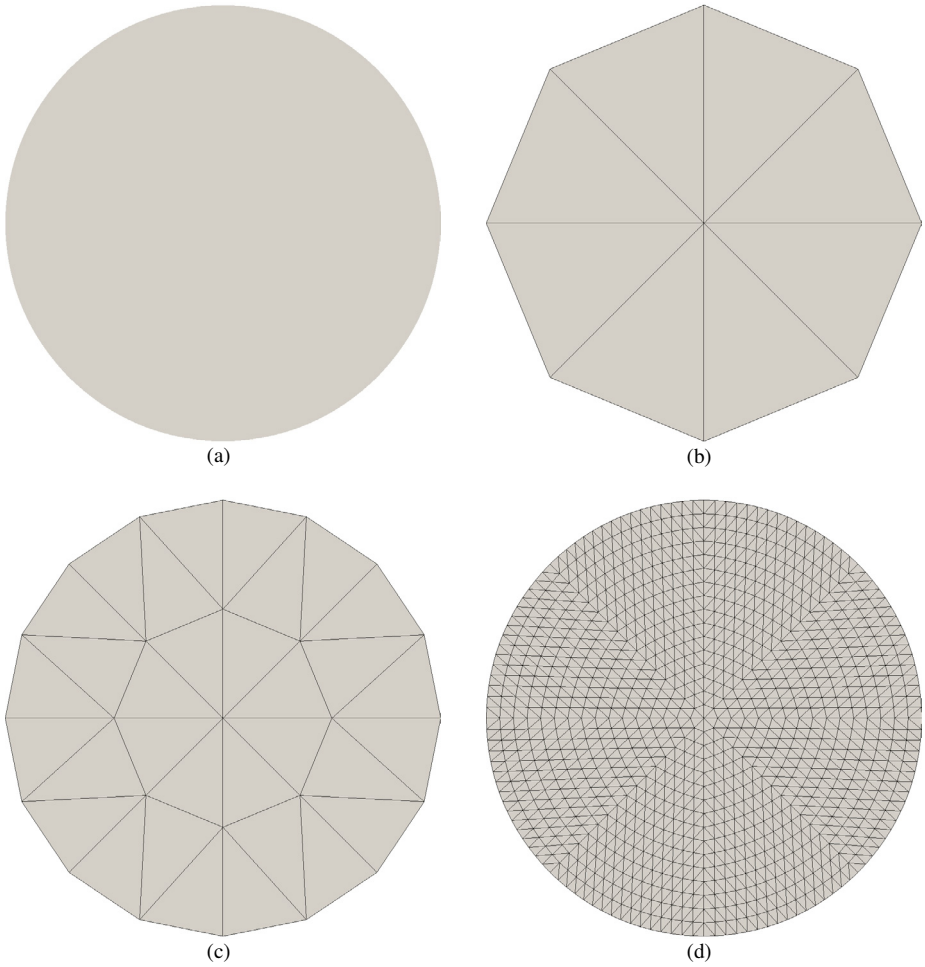

Figure 1. The subdivision approach applied to (a) a circle as the target curved geometry. The (b) initial coarse linear mesh is subdivided to obtain the meshes corresponding to (c) one and (d) four levels of refinement.

main details of the subdivision process, featuring the geometry represented by the high-order geometry associated to each coarse linear element.

2.1. Generate an initial linear mesh and a curved approximation of the target geometry

First, given the target geometry, Figure 1(a), we generate a coarse linear mesh, Figure 1(b). We assume that the mesh features the required resolution for the current problem (CFD) such as the desired anisotropy or boundary layer elements near the walls (see results in Sec. 4).

Following, we generate the curved approximation to the target geometry for each element, which will be used to subdivide each element while transferring the curved approximation of the target geometry. The geometry proxy for each element will be a high-order element with the same vertices than the linear one, but with the high-order nodes interpolating the geometry. To consistently assign a curved high-order geometry for each element, we generate a high-order curved mesh using an *a posteriori* process [21,22] that starts from the linear coarse mesh. The elements of the former coarse mesh are curved to match the boundaries of the computational domain. In this manner, the inherent curvature of the domain can be reproduced with higher accuracy. The method uses piece-wise polynomial representations of the elements that can be curved to match the domain boundaries by performing an optimization process [19,20] based on minimizing the mesh distortion [22,23]. Note that, to obtain a valid geometry representation,

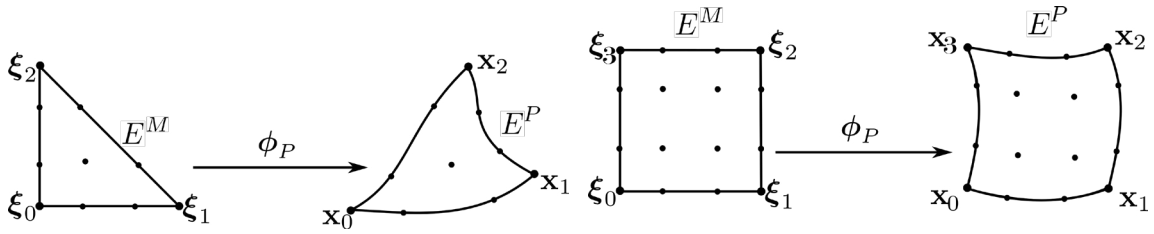

Figure 2. High-order representation mapping (polynomial degree 3) between the master and physical elements: (a) triangles, and (b) quadrilaterals.

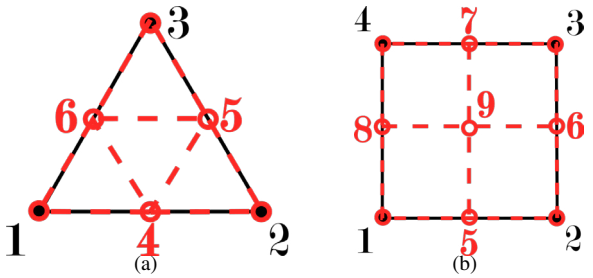

Figure 3. Element subdivision and local node re-numbering.

it may be necessary to modify the location of both the new high-order nodes but also of the vertices of the elements. Therefore, the location of the linear mesh nodes must be updated once the geometry has been generated to consistently match the elemental geometry with the coarse linear mesh.

In particular, we consider nodal high-order elements to represent the elemental geometry. Let E^P be an element of polynomial degree p determined by n_p nodes ($n_p = (p+1)\cdot(p+2)/2$ for triangles, and $n_p = (p+1)\cdot(p+1)$ for quadrilaterals) with coordinates $\mathbf{x}_i$ in $\mathbb{R}^d$, for $i = 1,\dots,n_p$. Given a master element E^M with nodes $\boldsymbol{\xi}_i$ in $\mathbb{R}^d$, being $i = 1,\dots,n_p$, we consider the basis $\{N_i\}_{i=1,\dots,n_p}$ of nodal shape functions (Lagrange interpolation) of degree p. Then, the high-order representation mapping from E^M to E^P, see Figure 2, can be expressed as:

$$\begin{aligned} \boldsymbol{\phi}_P : E^M \subset \mathbb{R}^d &\longrightarrow E^P \subset \mathbb{R}^d \\ \boldsymbol{\xi} &\longmapsto \mathbf{x} = \boldsymbol{\phi}_P(\boldsymbol{\xi}) = \sum_{i=1}^{n_p} N_i(\boldsymbol{\xi})\,\mathbf{x}_i. \end{aligned} \tag{1}$$

We highlight that once we have a valid curved high-order mesh, each high-order element is assigned to one linear element as a geometry proxy of the target geometry. This proxy is used to compute the coordinates of the new nodes obtained by element subdivision. Therefore, this allows to store the geometry representation as element-wise polynomial representations of degree greater than one only for those elements that have been strictly curved during the *a posteriori* procedure, *e.g.* those close to the curved boundaries. All the high-order elements that have not been curved in the high-order mesh generation process can be stored as straight-sided triangles (quadrilaterals) with linear (or bi-linear) shape functions. This reduces the geometrical data that has to be stored and distributed for each element.

2.2. Subdivide mesh preserving the curved geometry

The mesh subdivision process is composed by two main steps: the generation of the topology of the new elements, and the location of the new nodes in the curved geometry:

- **Topological subdivision.** Given an element, we perform its subdivision using the template presented in Figure 3. For each triangle we create 3 new nodes and 4 new elements. Analogously, for each quadrilateral we create 5 new nodes and 4 new elements. In the implementation of the topological operations, there is no difference between a straight-sided subdivision and the method proposed in this work, since the same number of nodes and the same sub-elements are created in each subdivision step.

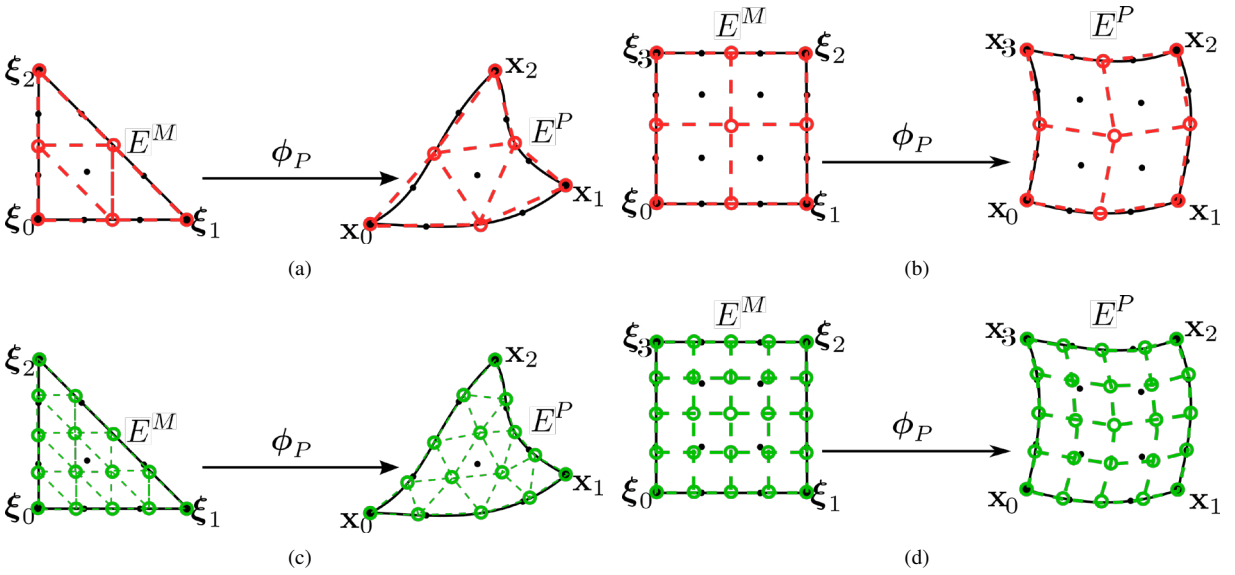

Figure 4. High-order master subdivision and mapping of the new nodes to the curved high-order element. In black we display the high-order master and physical elements and nodes, and in color the new nodes and elements resulting from: (a,b) one level of subdivision (red), and (c,d) two levels of subdivision (green).

- **New node location on curved geometry.** In a standard straight-sided subdivision process, the subdivided nodes can be directly computed on the physical mesh. Herein, first the location of the each new node in the master element (E^M) is computed, and following, this master coordinate is mapped to the corresponding geometry proxy (E^P) through the element-wise polynomial mapping presented in Eq. (1).
  In particular, given an edge [i,j], we can compute the master coordinate of the new mid-edge node as

$$\boldsymbol{\xi}_m = \frac{\boldsymbol{\xi}_i + \boldsymbol{\xi}_j}{2},$$

  being $\boldsymbol{\xi}_i$ and $\boldsymbol{\xi}_j$ the master coordinates of the edge nodes. Following, the new physical coordinate is computed as $\mathbf{x_m} = \boldsymbol{\phi}_P(\boldsymbol{\xi}_m)$. For the center node in the quadrilateral case, given $\boldsymbol{\xi}_i$ the master coordinates of the nodes ($i = 1, \ldots, 4$), we compute the coordinate in the master element as

$$\boldsymbol{\xi}_b = \frac{1}{4} \sum_{i=1,\ldots,4} \boldsymbol{\xi}_i,$$

  and the physical center node using the representation mapping as $\mathbf{x_b} = \boldsymbol{\phi}_P(\boldsymbol{\xi}_b)$.
  Figure 4 illustrates this process for triangles and quadrilaterals, with one and two levels of refinement. We can observe that the computation of the location of the new nodes is performed in the master element, and that the new physical coordinate is computed using the high-order mapping. In addition, we can observe that the curved mesh subdivision improves the geometrical representation from the first level of refinement (Figures 4(a) and 4(b)) to the second level of refinement (Figures 4(c) and 4(d)).

Figure 1 illustrates several levels of refinement of a given initial coarse mesh for a circle geometry. Starting from the initial coarse mesh presented in Figure 1(b) (8 elements), in Figure 1(c) we observe the result after one level of refinement ($8 \cdot 4^1 = 32$ elements). Figure 1(d) presents the result after 4 levels of refinement ($8 \cdot 4^4 = 2048$ elements). We observe that after each level of refinement there is an improvement of the accuracy of the geometry representation, since new nodes are always mapped to the curved elemental geometry representation. The level of geometrical accuracy is determined by the chosen polynomial degree to represent the geometry.

3. Parallel mesh subdivision

In this section, we first detail the parallel implementation of the proposed subdivision process, Sec. 3.1. Next, in Sec. 3.2 we highlight which are the main changes to perform in any straight-sided mesh subdivision code to enable the use of the proposed curved subdivision scheme.

3.1. Parallel implementation

The parallel implementation in Alya [17,18] of the mesh subdivision algorithm, presented for straight-sided mesh subdivision in [16], is based on a unique global node numbering and element based partition. Each subdomain data structure has access to the global numbering of the owned nodes, stored as a local array that maps the local numbering of the nodes to the global numbering. Moreover, the data structure provides access to the nodes in the interfaces between different subdomains. Specifically, each subdomain data structure stores an array with three types of nodes: interior nodes (not participating in MPI exchanges), boundary nodes that are owned by the actual subdomain, and boundary nodes owned by adjacent subdomains.

The mesh subdivision process presented in Sec. 2.2 is performed independently by all the subdomains. Once the meshes of each subdomain have been subdivided, the communication arrays are updated. In particular, the new created nodes are added to the communication array, taking into account if they are new interior nodes, new boundary nodes of the actual subdomain, and new nodes from other boundaries.

To update the node communication arrays between subdomains, neighboring subdomains exchange the list of boundary edges, since one new node has been created for each edge. The edges are ordered using the global numbering of the vertex nodes. In this manner, two adjacent subdomains assign the same numbering to the new interface nodes to be communicated. Then, the two given neighboring subdomains exchange their new boundary edges. Following, the common edges with the adjacent neighbor are identified and sorted. Finally, the communication node array is reallocated and the new edge nodes are added following the edge ordering. The new node is assigned to the subdomain with a smaller number of nodes.

To finalize, since new nodes have been created, the global vertices have to be renumbered to provide a new global numbering that leads to a sparse matrix with reduced filling. To this end, we use the `METIS_NodeND` [24] function. Each subdomain k numbers its interior and owned boundary nodes consecutively as $l_j = j + \sum_{i=1}^{k-1} n_i,\ j = 1, \dots, n_k$, being l_j the global numbering of the local jth node. Once all the interior and owned boundary nodes are labeled, the results are exchanged. In our implementation we use the `MPI_Sendrecv` routine. Thus, the global numbering of the boundary nodes that are owned by a different subdomain can be labeled with the received data from the neighboring subdomains.

3.2. Extending a linear mesh subdivision scheme to approximate a curved geometry

In this section, we highlight the main steps that must be performed to inherit the geometry curvature to an already implemented and existent straight-sided mesh subdivision procedure:

1. Implementing high-order shape functions in the code to allow iso-parametric mappings of the desired polynomial degree to be represented. For instance, geometry elements of polynomial degree 2 feature an additional node in the middle of: the edges, the quadrilateral faces, and the hexahedral elements. The coordinates of these new nodes allow capturing the curvature of the target geometry, and shape functions of degree 2 must be implemented to compute the required representation mapping, see Eq. (1).

2. Representing the curved geometry proxy as a vectorial elemental field associated to the standard linear mesh. On the one hand, the linear mesh determines the topology of the curved mesh. On the other hand, the components of the vectorial element field are the coordinates of the nodes of the high-order elements and therefore, determine the geometry of the curved mesh. This approach allows to re-use part of the existing code to refine in parallel linear meshes. Specifically, it allows to re-use the part of the implementation that deals with the topology of the mesh.

3. Incorporating the capability to read a curved mesh as an elemental field. We highlight that no curved mesh data structure is required, since each curved element is read and stored as a field associated to the corresponding linear element. This field will only be queried in the mesh subdivision process, using the element-wise polynomial representation mapping to compute the physical coordinates of the new nodes.

4. Changing the computation of the coordinates of the new nodes in the middle of edges and faces to allow preserving the curvature of a target geometry, see Sec. 2. Accordingly, we have modified the code in such a manner that the coordinates of the new nodes interpolate the target geometry. To this end, the target geometry is represented element-wise by iso-parametric high-order elements, and new code to compute the master and physical coordinates of the new nodes must be implemented.

Enabling the previous capabilities in the mesh subdivision code will allow the preservation of the geometric accuracy of a piece-wise polynomial representation of the domain. Note that to incorporate curvature, the polynomial degree of the representation has to be greater or equal than two. Herein, the coarse curved mesh is given to the Alya code, which is based on a master-slave strategy. The master reads the linear mesh and an elemental geometry field (high-order element associated to the coarse linear one), partitions them, and sends the subdomain meshes to the slaves. The slaves then refine their corresponding subdomains in parallel, by performing the subdivision of the coarse elements in finer linear elements. During the process we have to ensure that the boundary nodes of the new elements match with the curved elements of the neighbouring subdomains. The result is a finer linear mesh obtained in parallel with improved geometric accuracy. In a practical example, we can expect that the coarse curved mesh that approximates the target geometry features significantly less elements than the obtained finer mesh. For instance, the computational mesh could have between 10M and 100M of elements whereas the curved mesh to be partitioned and distributed could have only between 10K and 100K of elements.

4. Results

In this section, we present the different tests to analyze the performance and the features of the parallel curved mesh subdivision algorithm. First, we qualitatively analyze the ability of the proposed approach to recover the curvature of the target geometry. Second, we study the weak scaling of the subdivision process. Finally, we present some preliminary results in 3D, where we also analyze the qualitative improvement of the flow solution when recovering the target curvature with the subdivided mesh.

All the results have been obtained on the super-computer MareNostrum3, with computing nodes interconnected through Infiniband FDR10 and equipped with the GPFS filesystem via ethernet of 1 GBit/s. Each computing node features 128GB of memory (8 GB/core) and two eight-core E52670 SandyBridge-EP CPUs (16 cores/node), each CPU with a clock frequency of 2.6GHz and 20MB of cache. The code has been fully developed in Fortran, implemented inside Alya, an in-house finite elements multi-physics parallel solver [17,18].

We assume that a coarse linear mesh has been obtained and that the mesh features the required resolution for the current problem (CFD) such as boundary layer elements near the wall. Specifically, we generate the coarse linear mesh of the topography and the corresponding elemental geometry using WindMesh [25], an in-house mesh generator for Atmospheric Boundary Layer flows in real topographies. The methods to generate a valid curved high-order mesh that approximates the target geometry are presented in [19,20,22,26].

4.1. Application to simulate Atmospheric Boundary Layer flows over terrains

In this section, we study the capabilities of the mesh subdivision method with the simulation of the Atmospheric Boundary Layer flow over a 2D section of the topography of a wind farm placed in Mexico, see Figure 5(a). The meshes are used to solve the Reynolds Averaged Navier-Stokes (RANS) equations with a $k - \varepsilon$ turbulence model ready to consider the Atmospheric Boundary Layer (ABL).

We generate an initial mesh composed by 68391 elements, Fig. 5(b), and we perform four levels of parallel subdivision with 256 cores, obtaining a final mesh composed by 17.5 Million elements. The subdivision is performed both with the standard straight-sided subdivision approach, Fig. 5(c), and the new curved one, Fig. 5(d). In Figure

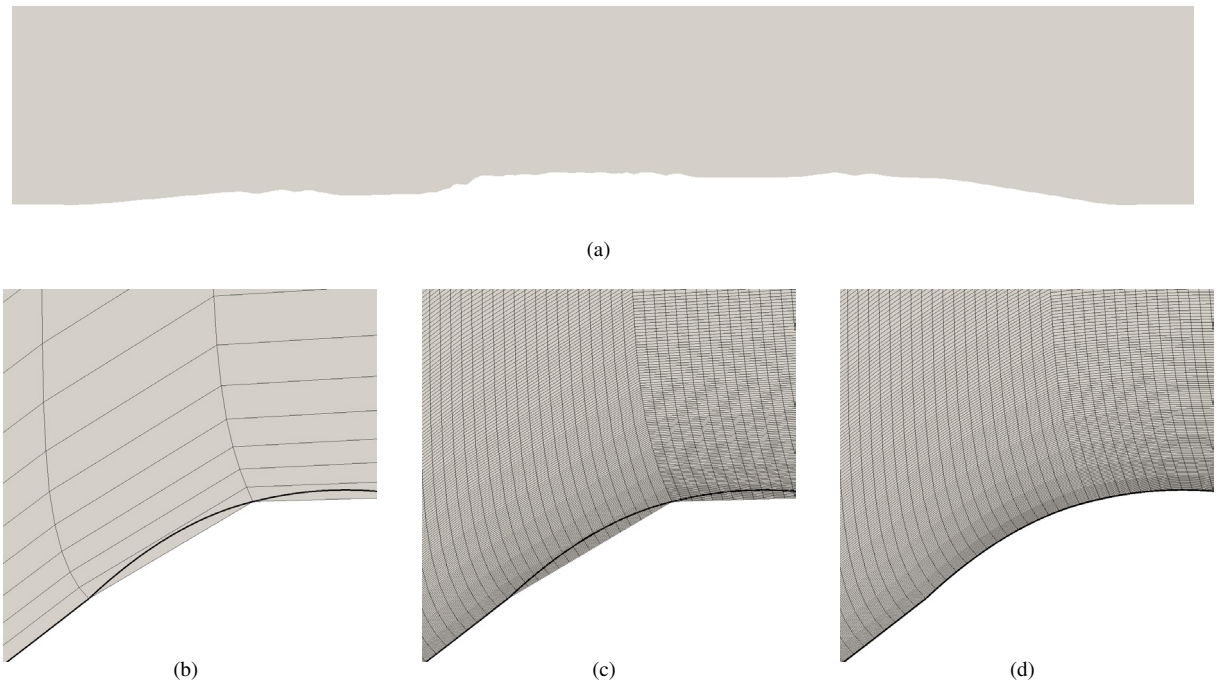

Figure 5. (a) Puebla topography overview. (b) Initial coarse mesh. (c) Mesh subdivision using standard approach. (d) Mesh subdivision using curved proposed approach. In (b,c,d) the geometry is displayed with a thick black line.

5(b) we observe that the actual geometry crosses almost completely the first layer of elements of the boundary layer. This intersection of the linear mesh with the geometry implies that in the standard subdivision approach, 2^4 levels of elements around the geometry will be folded by the new boundary nodes if they are projected to the geometry (using standard subdivision procedures with surface correction). This would demand of a robust and cost-consuming optimization process that features smoothing and untangling to obtain a final valid subdivided mesh. With the proposed approach, we use a area (volume) representation of the 2D (3D) geometry instead of a standard curve (surface) boundary representation. Accordingly, with our approach it is not required to project the new boundary nodes to the boundary representation of the geometry.

One of the main highlights of the presented technique is that, not only the boundary nodes are subdivided according to the geometry, but also the interior ones. The coordinates of a new node are never computed in the physical space. As highlighted in Sec. 2, the coordinates are always computed in the master space of each geometry element. Hence, once the subdivided master coordinates are computed, they are mapped to the physical space. To obtain a valid mesh after the subdivision process it is necessary that the element-wise polynomial approximation of the target geometry corresponds to a valid curved mesh. Note that for successive refinements the geometric accuracy of the subdivided mesh will converge to the geometric accuracy of the curved mesh.

In Figure 6 we illustrate the steady-state solution of the RANS equations for the mesh subdivided using the straight-sided (left) and curved (right) subdivision approaches. We highlight that, although for both meshes the flow solver converges to a steady-state solution, artificial features appear in the mesh obtained with the straight-sided subdivision approach. These features are originated close to the vertices of the initial linear mesh, since the straight-sided mesh subdivision process does not recover the curvature of the target geometry. Specifically, the variations between the normal vectors of adjacent boundary faces of the initial mesh are preserved during the subdivision process. On the contrary, with the proposed subdivision methods, the successive refinement of the mesh improves the geometric accuracy of the boundary representation. This is so, since new nodes are introduced on top of a curved approximation of the target geometry. This reduces the variation of the normal vectors between adjacent boundary faces and therefore, mitigates the artificial artifacts that appear in meshes generated with standard subdivision approach.

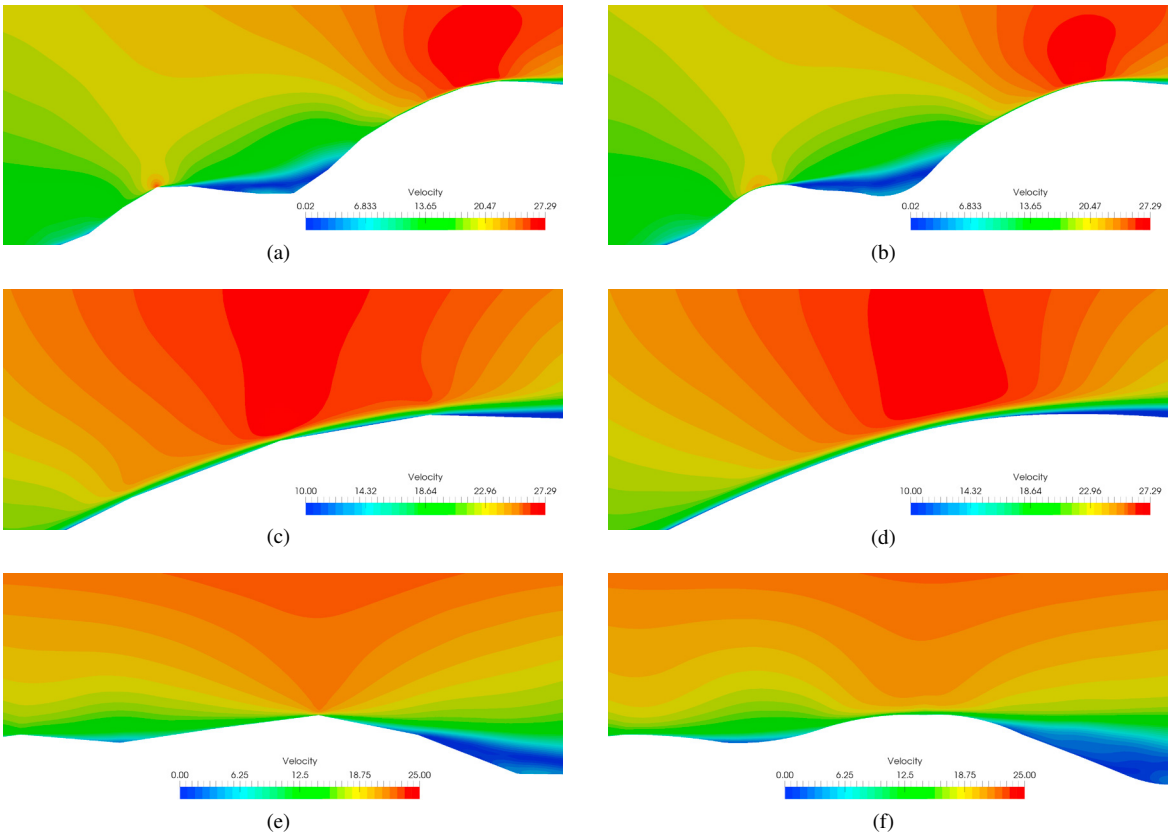

Figure 6. Velocity magnitude results with three levels of subdivision using (a,c,e) standard straight-sided approach, and (b,d,f) proposed curved process.

4.2. Scaling 2D

Two weak scaling tests have been carried out. The tests have been performed on the 2D topographic scenario presented in Figure 5. Five coarse initial meshes have been generated, featuring from 1.9K elements to 30K. The mesh subdivision process has been carried out in five different configurations, featuring from 16 to 256 cores. For each configuration, the initial number of elements per core is 118.

In Table 1 we illustrate the computational time for two different subdivision tests, using four and eight levels of subdivision. With four levels of subdivision, the number of elements of the initial mesh is multiplied by 256 (4^4) in the final mesh, whereas for eight levels, the final number is multiplied by 65536 (4^8). Using four levels, each processor will have a load of 30K elements, and the final meshes will be composed from 0.48 Million (M) elements to 7.7M. For the eight levels case, the final load of the processors is 7.7M elements, and the subdivided meshes are composed from 0.12 Billion (B) elements to 1.97B. We observe that, for both tests, the computational time for the mesh subdivision remains constant for a constant load of the cores. Therefore, the implementation of the proposed parallel mesh subdivision methods presents an ideal performance for the weak scaling test.

4.3. Preliminary results in 3D

Following the same ideas presented for triangle and quadrilateral meshes presented in Sec. 2, the curved subdivision technique can be extended to 3D meshes. The extension to hexahedral and tetrahedral meshes is currently under development, and some preliminary results for hexahedral meshes are presented in this section.

Table 1. Weak scaling for 4 and 8 subdivision steps, denoting by n_E^k is the number of elements for k subdivision steps

Cores	n_E^0	n_E^4	Time (s)	n_E^8	Time (s)
16	1.9K	0.48M	1.00	0.12B	273
32	3.8K	0.96M	1.01	0.25B	275
64	7.5K	1.93M	1.02	0.49B	275
128	15K	3.84M	1.01	0.98B	274
256	30K	7.70M	1.02	1.97B	273

Figure 7 presents a comparison between the straight-sided and the curved subdivision methods and the corresponding steady-state solution of a RANS simulation of the ABL flow over a synthetic topography. With the standard mesh subdivision process, the target geometry is discretized by a coarse linear mesh composed of 24K hexahedra, see Figure 7(a), and we refine it to obtain a finer mesh, see Figure 7(b). Accordingly, the refined mesh reproduces the straight-side geometry represented by the initial mesh. Using the new proposed curved subdivision approach, the target geometry is represented by an elemental field composed by curved quadratic hexahedra, see Figure 7(b). After four consecutive subdivisions with 3072 cores, with both approaches we obtain a finer distributed linear mesh composed of 98.3M elements ($24\text{K} \cdot 8^4$ elements), few seconds later and completely stored in the memory.

With the standard subdivision approach we are able to converge to a steady state of the flow. However, the finer representation of the initial straight-sided geometry leads to artificial features in the solution, see Figures 7(a) and 7(b). These features are visible in those regions where the mesh features sharp edges characterized by large changes in the normal vector between contiguous faces. Using the finer mesh obtained with the new approach, we are able to converge to a steady flow state with an improved accuracy of the topography geometry and without artificial flow features, see Figures 7(b) and 7(d). This artificial features are not present with the new technique because the subdivided mesh features a significantly smaller variation of the normal vector between adjacent faces.

In Figure 8, we present a general and a detailed view of the initial straight-sided coarse mesh used in the previous version of the code. In addition, we include a section of the steady state flow solution obtained with the standard subdivision approach, see Figures 8(a) and 7(c), and with our subdivision approach, see Figures 8(b) and 8(d) . When the standard subdivision approach is used, we observe that the artificial flow features are found next to the sharp edges of the initial straight-sided mesh. Instead, in Figures 8(b) and 8(d) we observe that the flow solution is smoother and without artificial features when our subdivision approach is used. The blank space between the initial straight-sided mesh and the flow solution illustrates the error in the approximation of the target geometry when a standard subdivision approach is used.

5. Concluding remarks and future work

In this paper, a new approach to perform a mesh subdivision process that recovers the curvature of a target geometry is presented. Once generated a linear coarse mesh, we assign to each of the element a curved high-order element that matches the target geometry. The mesh is partitioned and distributed, and the mesh subdivision process is performed computing the coordinates of the new nodes in the master element. Then, these master coordinates are mapped to the physical coordinates by means of the element-wise polynomial representation of the corresponding curved high-order elements that approximate the target geometry. Since the curved geometry is approximated element-wise, the mesh subdivision process is able to reproduce a element-wise polynomial approximation of the geometry without requiring to project nodes to the boundary.

We illustrate the proposed approach in a 2D simulation of the Atmospheric Boundary Layer flow over a section of a real topography, analyzing the advantages of the curved against the straight-sided subdivision. In addition to the improvement of the geometrical accuracy of the mesh representation, clear benefits on the CFD solution are observed. That is, the new approach mitigates the artificial artifacts of the flow arising close to the vertices of the initial mesh. We also perform a weak scaling test of the code up to 256 cores. Specifically, we tested that the code behaves ideally since it keeps a constant computational time for a constant load of the cores (increasing accordantly the number of cores and the number of elements of the mesh). Finally, we show some preliminary results in 3D, where we observe the same qualitative advantages than in the 2D case.

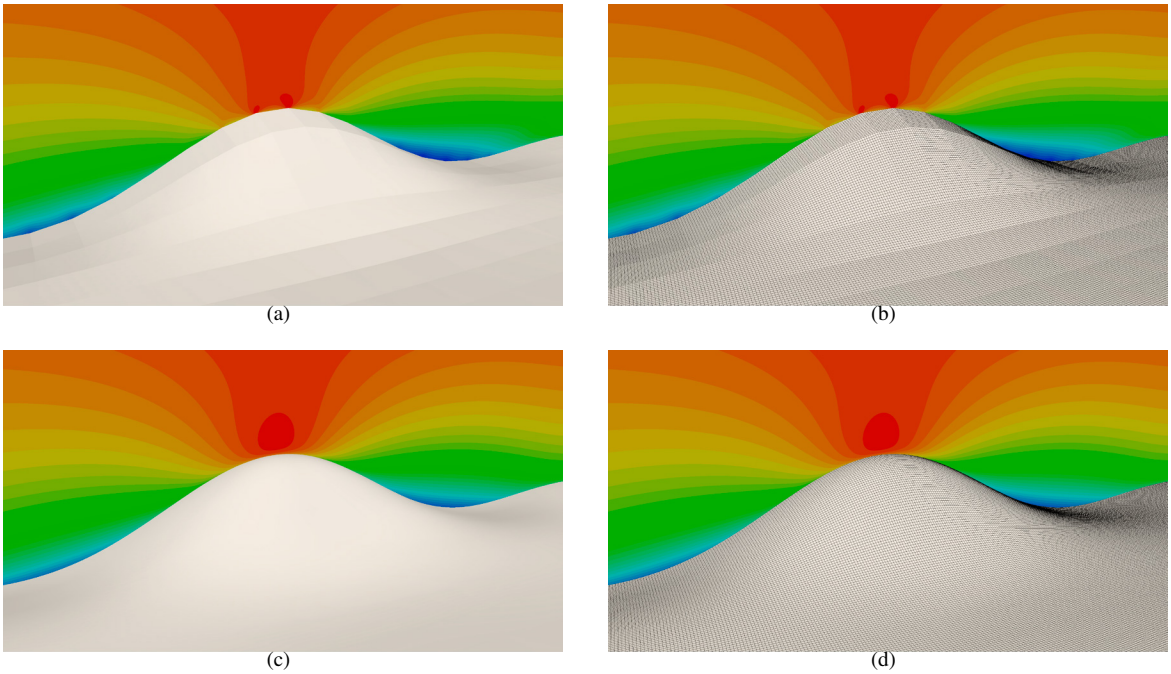

Figure 7. Standard subdivision approach: (a) initial approximation to the geometry and (b) refined mesh. Proposed subdivision approach: (c) initial curved approximation to the geometry and (d) refined mesh. For both approaches, the velocity magnitude of the steady state flow solution is illustrated.

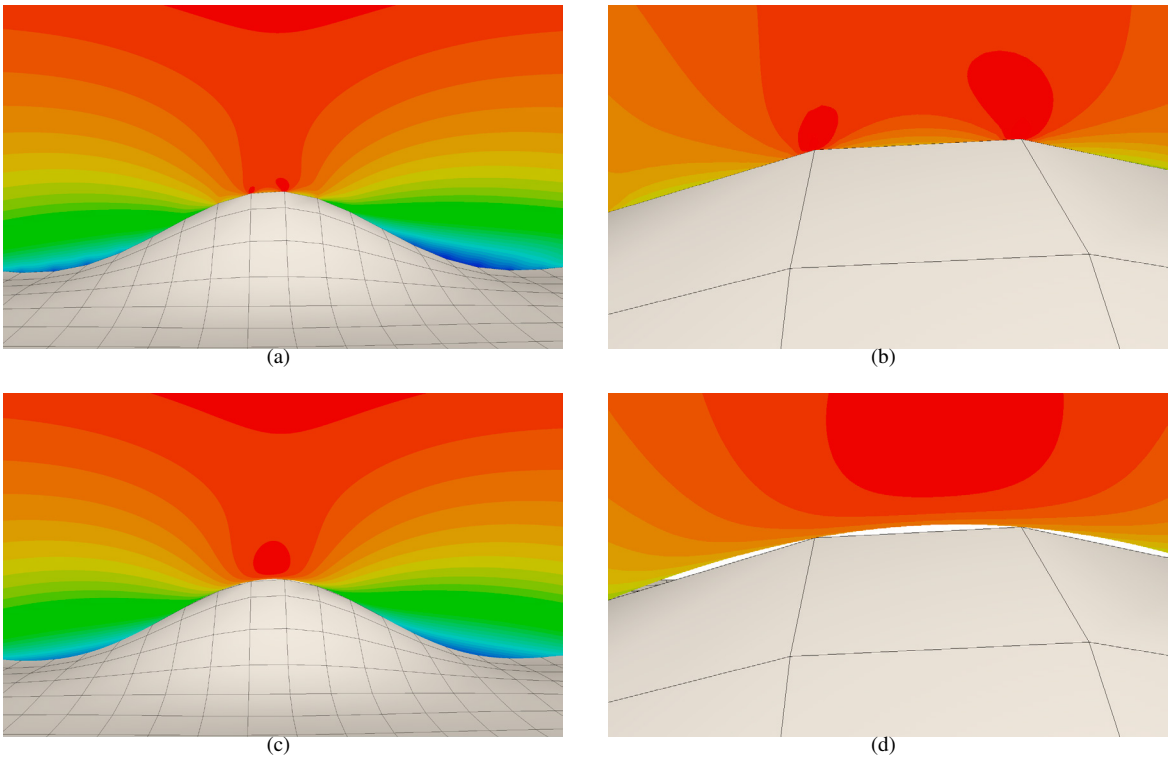

Figure 8. (a, b) General view and (c, d) detailed view of the standard subdivision mesh and a section of the velocity magnitude for the flow steady state on a mesh obtained with: (top) the standard subdivision approach and (bottom) the proposed subdivision approach.

The implementation in 3D of the proposed methods is still ongoing work. The subdivision technique is already implemented for hexahedra but, the implementation for tetrahedral elements has to be finished. Once the technique is fully implemented to obtain finer 3D linear elements, we plan on extending the subdivision technique to obtain finer 2D and 3D curved meshes. Specifically, we will approximate the target geometry with a polynomial degree higher than the desired polynomial degree of the final finer mesh. This will allow to improve the geometric accuracy in the successive application of the subdivision approach while obtaining a finer curved mesh.

Acknowledgments

This work was financially supported by the PRACE project funded in part by the EUs Horizon 2020 research and innovation program (2014-2020) under grant agreement 653838. This project has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme under grant agreement No 715546. The work of the corresponding author has been partially supported by the Spanish Ministerio de Economía y Competitividad under the personal grant agreement RYC- 2015-01633.

References

[1] R. Löhner, A parallel advancing front grid generation scheme, International Journal for Numerical Methods in Engineering 51 (2001) 663–678.
[2] R. Löhner, A 2nd Generation Parallel Advancing Front Grid Generator, Springer Berlin Heidelberg, 2013, pp. 457–474.
[3] Y. Ito, A. M. Shih, A. K. Erukala, B. K. Soni, A. Chernikov, N. P. Chrisochoides, K. Nakahashi, Parallel unstructured mesh generation by an advancing front method, Mathematics and Computers in Simulation 75 (2007) 200 – 209. Applied Scientific Computing: Advanced Grid Generation, Approximation and Simulation.
[4] T. Okusanya, J. Peraire, Parallel unstructured mesh generation, in: Joint ASME/ ASCE/SES summer meeting, Norris Center, Northwestern University, 1996, pp. 1–11.
[5] N. Chrisochoides, D. Nave, Parallel delaunay mesh generation kernel, International Journal for Numerical Methods in Engineering 58 (2003) 161–176.
[6] N. Chrisochoides, D. Nave, Simultaneous mesh generation and partitioning for delaunay meshes, Mathematics and Computers in Simulation 54 (2000) 321 – 339.
[7] A. Johnson, S. Aliabadi, A. H. P. C. R. Center, U. of Minnesota, Application of Automatic Mesh Generation and Mesh Multiplication Techniques to Very Large Scale Free-surface Flow Simulations, AHPCRC preprint, Army High Performance Computing Research Center, 1999.
[8] E. Yilmaz, S. Aliabadi, Mesh multiplication technique with surface correction, PARCFD2011, Barcelona (Spain) (2011) 16–20.
[9] E. Yilmaz, S. Aliabadi, Surface conformed linear mesh and data subdivision technique for large-scale flow simulation and visualization in variable intensity computational environment, Computers & Fluids 80 (2013) 388–402.
[10] Ó. Antepara Zambrano, O. Lehmkuhl Barba, R. Borrell Pol, C. Oliet Casasayas, A. Oliva Llena, Parallel mesh multiplication and adaptation technique for turbulent flow simulation using unstructured meshes, in: ParCFD 2015: 27th International Conference on Parallel CFD, 2015, pp. 1–8.
[11] M. Staten, B. Carnes, C. Mcbride, C. Stimpson, J. Cox, Mesh scaling for affordable solution verification, Procedia Engineering 163 (2016) 46–58.
[12] H. Chaurasia, X. Roca, P. Persson, J. Peraire, A coarse-to-fine approach for efficient deformation of curved high-order meshes, in: Research Notes, 21st Int. Meshing Roundtable, Springer International Publishing, 2012, pp. 1–5.
[13] A. Gargallo-Peiró, X. Roca, J. Peraire, J. Sarrate, Inserting curved boundary layers for viscous flow simulation with high-order tetrahedra, in: Research Notes, 22nd Int. Meshing Roundtable, Springer International Publishing, 2013, pp. 1–5.
[14] D. Moxey, M. Green, S. Sherwin, J. Peiró, An isoparametric approach to high-order curvilinear boundary-layer meshing, Computer Methods in Applied Mechanics and Engineering 283 (2015) 636 – 650.
[15] D. Moxey, M. Hazan, S. J. Sherwin, J. Peiro, Curvilinear Mesh Generation for Boundary Layer Problems, Springer International Publishing, 2015, pp. 41–64.
[16] G. Houzeaux, R. de la Cruz, H. Owen, M. Vázquez, Parallel uniform mesh multiplication applied to a navier-stokes solver, Computers & Fluids 80 (2013) 142 – 151. Selected contributions of the 23rd International Conference on Parallel Fluid Dynamics ParCFD2011.
[17] G. Houzeaux, M. Vázquez, R. Aubry, J. Cela, A massively parallel fractional step solver for incompressible flows, J. Comput. Phys. 228 (2009) 6316 – 6332.
[18] G. Houzeaux, B. Eguzkitza, M. Vázquez, A variational multiscale model for the advection-diffusion-reaction equation, Commun. Numer. Meth. En. 25 (2009) 787–809.
[19] A. Gargallo-Peiró, X. Roca, J. Peraire, J. Sarrate, Optimization of a regularized distortion measure to generate curved high-order unstructured tetrahedral meshes, Int. J. Numer. Meth. Eng. 103 (2015) 342–363.
[20] A. Gargallo-Peiró, X. Roca, J. Peraire, J. Sarrate, A distortion measure to validate and generate curved high-order meshes on CAD surfaces with independence of parameterization, Int. J. Numer. Meth. Eng. 106 (2015) 1100–1130.
[21] P.-O. Persson, J. Peraire, Curved mesh generation and mesh refinement using lagrangian solid mechanics, in: Proc. 47th AIAA, 2009, pp. 1–11.

[22] A. Gargallo-Peiró, Validation and generation of curved meshes for high-order unstructured methods, Ph.D. thesis, Universitat Politècnica de Catalunya, 2014.
[23] A. Gargallo-Peiró, X. Roca, J. Peraire, J. Sarrate, Distortion and quality measures for validating and generating high-order tetrahedral meshes, Eng. Comput. 31 (2015) 423–437.
[24] G. Karypis, V. Kumar, Metis-family of multilevel partitioning algorithms, http://glaros. dtc. umn. edu/gkhome/views/metis, 1998.
[25] A. Gargallo-Peiró, M. Avila, H. Owen, L. Prieto, A. Folch, Mesh generation for atmospheric boundary layer simulation in wind farm design and management, Procedia Engineering 124 (2015) 239–251.
[26] A. Gargallo-Peiró, X. Roca, J. Sarrate, A surface mesh smoothing and untangling method independent of the CAD parameterization, Comput. Mech. 53 (2014) 587–609.