PRE-CONDITIONING AND CONTINUATION FOR PARALLEL DISTRIBUTED MESH CURVING

Eloi Ruiz-Gironés Xevi Roca

Computer Applications in Science and Engineering, Barcelona Supercomputing Center - BSC, 08034 Barcelona, Spain. {eloi.ruizgirones,xevi.roca}@bsc.es

ABSTRACT

To generate large-scale meshes with highly stretched elements for complex geometries, we propose a distributed parallel Newton-GMRES penalty solver. For each non-linear iteration, we solve a pre-conditioned sparse linear problem. To choose the parallel pre-conditioner, we compare an algebraic multi-grid implementation with a restricted additive Schwarz domain decomposition with one level of overlap and local problems approximated with symmetric successive over-relaxation. We show that domain decomposition is faster and more energy efficient. Furthermore, to accelerate the penalty based solver, we propose a novel p-continuation technique with two unique features. First, it has an early termination criterion to stop the optimization of the initial polynomial degrees. Second, it estimates the initial penalty parameter for each polynomial degree. We conclude that this continuation can reduce four times (eight times) the wall clock time (energy per core) required to curve a whole boundary layer quartic mesh using the chosen domain decomposition.

Keywords: High-order mesh curving, parallel, pre-conditioner, p-continuation

1. INTRODUCTION

Curved high-order meshes are required for unstructured high-order methods to keep their advantages [1–5]. These advantages come in the form of geometric flexibility, high accuracy, and low numerical dissipation and dispersion. High-order methods feature exponential convergence rates and therefore, they have been proven to be faster than low-order methods in several applications [6–14], especially in those problems where an implicit solver is required [15].

Usually, to generate a curved high-order mesh an *a posteriori* approach is used [16–26]. First, a linear mesh with elements of the desired shape and size is generated, and then, the mesh boundary is curved to match the target geometry. This step may introduce low-quality and inverted elements that have to be repaired using a high-order mesh curving technique. There are several manners to formulate the mesh curving problem: PDE-based methods like solid mechanics analogies [21, 24, 27–30] or the Winslow equation [25], and optimization-based methods [23, 31–34]. However, none of these methods can run in a distributed parallel environment while taking into account virtual geometry to generate large curved high-order meshes of complex domains, as we detail in Section 2. This capability, is critical to perform in parallel unsteady (fine graded meshes) or steady state (fine meshes with stretched elements) flow simulations for large Reynolds numbers with unstructured high-order methods.

To generate fine graded and stretched curved high-order meshes in parallel, we parallelize a high-order mesh curving formulation that takes into account virtual geometry [35]. The previous method poses the mesh curving problem as a constrained optimization problem in which the mesh distortion is minimized constrained to a non-linear boundary condition. This non-linear constraint allows different types of boundary conditions such as fixed boundary nodes, nodes sliding on the geometry, and nodes sliding on top of

virtual entities. The constrained optimization is solved using a penalty method in which the boundary condition is introduced into the objective function using a penalty parameter. Then, several optimization problems are solved while increasing the penalty parameter to enforce the boundary condition. he optimization process is devised to favor that at each optimization step, a valid mesh is deformed to a valid mesh. To this end, two main ingredients have been considered. The first one is a functional that penalizes inverted elements by taking an infinite value. The second ingredient is a backtracking line search to Newton's method. Thus, if a Newton full step were deforming a valid mesh to an invalid mesh, leading to an infinite value of the functional, the backtracking line-search would decrease the step length until a valid mesh is obtained, leading to a finite value.

To solve the large linear problems arising in the global optimization of large curved high-order meshes, we need a pre-conditioned distributed parallel linear solver. The pre-conditioner is required since fine graded meshes with highly stretched elements hamper the condition number of the linear systems. The distributed parallel linear solver allows accommodating the large linear systems across the memory of different machines.

To select the parallel pre-conditioner, we have to consider the influence of the number of elements, element stretching, and the number of cores used by the linear solver. This consideration is so since when higher is the number of elements and their stretching more ill-conditioned is the linear system and thus, greater is the number of linear iterations. Moreover, increasing the number of cores might reduce the capacity of the parallel pre-conditioner to propagate the information across the mesh boundaries.

The first contribution of this work focuses on the selection of pre-conditioners for the linear problems that arise during the curving of high-order meshes. We consider two pre-conditioners, and we compare the time to solve the linear problems, the number of iterations of the linear solvers, and the energy consumption to generate the high-order curved mesh. As the method uses more computational resources during more time, the economic cost of generating the high-order mesh also increases. Therefore, it is important to devise efficient mesh curving methods in terms of computational resources like time and memory. This consideration is especially important when curving large-scale meshes in supercomputers where the computational resources are directly translated into energy consumption and economic costs.

In the second contribution, we develop a p-continuation technique to increase the robustness and computational efficiency of the mesh curving process. The main idea is to use the converged solution of a given polynomial degree as the initial condition for the next polynomial degree. Thus, the p-continuation technique can be interpreted as a methodology to compute the initial condition for the final polynomial degree.

We propose two unique features in the p-continuation technique to reduce the number of linear and non-linear problems to solve, and to improve the robustness of the optimization. The first novelty is to introduce an early-termination criterion to finalize the optimization of the initial polynomial degrees. Using this criterion, we avoid solving additional non-linear problems, and therefore, the use of computational resources is reduced. The second novelty is to estimate the value of the penalty parameter in the first iteration of each polynomial degree. The value of the penalty parameter is critical to converge the mesh curving process. If the penalty parameter is too high, the linear problems become more difficult, and the linear solver may not be able to solve the linear problem.

The rest of the paper is structured as follows. Section 2 reviews the literature related to the presented work. Section 3 presents the formulation of the proposed high order mesh curving methodology. Section 4 presents several examples to show the capabilities of the proposed formulation. Finally, Section 5 details the conclusions of this work.

2. RELATED WORK

Mesh curving methods can be divided into global and local methods. Global methods move all the nodes at the same time, while local methods move one node at a time. Global methods can be further divided into implicit methods that need to solve a sparse linear system, and explicit methods that move the nodes using an explicit formula. Note that herein, global (local) methods do not refer to obtain the global (local) minimum of an objective function.

One of the main bottlenecks in global implicit methods is the solution of a sparse linear problem to relocate all the mesh nodes at the same time. Therefore, efficient sparse linear solvers and pre-conditioners are necessary to curve high-order meshes composed of a large number of elements. In several works, the linear systems are solved using sparse direct solvers for morphing linear meshes [36], and curving high-order meshes [29, 37]. Nevertheless, when the number of elements in the mesh increases, it is necessary to use iterative solvers. In general, the linear problems are solved either with MINRES, GMRES or conjugated gradients [25, 34, 38–40]. To improve the computational efficiency, the linear problems can be pre-conditioned in different manners. For instance,

diagonal pre-conditioners [38], incomplete factorizations [25, 39], or especially designed pre-conditioners for the mesh curving problem [41]. Global methods need to assemble a sparse matrix and solve the associated linear problem. Therefore, the parallelization of these methods need to assemble the matrix in parallel and apply a parallel sparse iterative linear solver, see [34, 35, 40].

To avoid solving a sparse linear system, reference [42] relocates the nodes using a first order steepest descent minimization method. That is, the method relocates the nodes using a multiple of the objective function gradient. Although the convergence rate is lower than using Newton's method, the iterations are performed faster and the memory requirements are reduced.

Instead of solving a fully coupled linear problem to move all the nodes at the same time, local approaches move one node at a time. This approach is both applied for linear meshes [43, 44] and high-order meshes [22,26,45]. Although the local approach uses less memory and each iteration is faster than in the global approach, the convergence rate to the optimal solution can be hampered. In local approaches, nodes that do not belong to the same element can be relocated at the same time. Thus, several authors propose to color the nodes of the mesh in such a manner that nodes of the same color can be relocated at the same time [26, 43, 44]. Moreover, in [44], they propose a mesh partitioning method to ensure that the cost to relocate the nodes in each sub-domain is roughly the same.

In the proposed work, we propose a distributed memory implementation that moves all the nodes at the same time. To accomplish this, we solve a sparse linear system of equations derived from Newton's method. Therefore, we expect faster convergence rates than first-order methods or local methods.

3. PARALLEL MESH CURVING SOLVER

We first briefly summarize the proposed mesh curving solver and non-linear solver, as presented in [35], and then introduce the new p-continuation technique to increase the robustness and the efficiency of the method.

3.1 Mesh Curving Problem

Given an initial linear mesh, $\mathcal{M}_I$, we want to characterize a curved high-order one, $\mathcal{M}_P$, in terms of a diffeomorphism $\boldsymbol{\phi}^*$ [45, 46]. The optimal diffeomorphism presents optimal point-wise distortion, and satisfies a prescribed boundary condition. That is, $\boldsymbol{\phi}^*$ is the minimizer of

$$\begin{aligned} &\min_{\boldsymbol{\phi}\in\mathcal{V}} E(\boldsymbol{\phi}) = \|M\boldsymbol{\phi}\|^2 \\ &\text{subject to:} \\ &\boldsymbol{T}\boldsymbol{\phi} = \mathbf{g}_D(\boldsymbol{T}\boldsymbol{\phi}), \end{aligned} \tag{1}$$

where $\boldsymbol{T}$ is the trace operator, $\mathbf{g}_D(\boldsymbol{T}\boldsymbol{\phi})$ is a non-linear Dirichlet boundary condition on $\partial\mathcal{M}_I$ that depends on the values of $\boldsymbol{\phi}$, and

$$M\boldsymbol{\phi}(\mathbf{y}) = \eta(\mathbf{D}\boldsymbol{\phi}(\mathbf{y})) = \frac{\|\mathbf{D}\boldsymbol{\phi}(\mathbf{y})\|^2}{n\sigma_0(\mathbf{D}\boldsymbol{\phi}(\mathbf{y}))^{2/n}}$$

is a regularized point-wise distortion measure [23] defined in terms of the shape distortion measure for linear simplices [47], where $\|\cdot\|$ is the Frobenius norm for matrices, and

$$\sigma_0 = \frac{1}{2}\left(\sigma + |\sigma|\right), \tag{2}$$

being $\sigma(\cdot)$ the determinant function. The regularized distortion measure takes a value of infinity when the determinant is negative or equal to zero, and takes finite values when the determinant is positive.

The non-linear boundary condition allows integrating a geometric model in the mesh curving process. In this work, we define the boundary condition as

$$\mathbf{g}_D(\boldsymbol{T}\boldsymbol{\phi}) = \sum_{i=1}^{N_b} \Pi(\mathbf{x}_i) N_i^b, \tag{3}$$

where $\mathbf{x}_i$ are the coordinates of the mesh nodes, N_b is the number of boundary nodes, $\{N^b\}_{i=1,\ldots,N_b}$ is a Lagrangian basis of shape functions continuous between adjacent boundary faces, and $\Pi(\cdot)$ is a geometric orthogonal projection onto the CAD model. The boundary condition can be interpreted as in interpolation of the geometric model, in which the interpolation points are the projection of the boundary nodes. This boundary condition is non-linear and depends on the orthogonal projection of the boundary nodes.

3.2 Mesh Curving Non-Linear Solver

To solve the constrained optimization problem in (1), we use a penalty approach, see [48], in which we introduce the boundary constraint into the objective function in a weak sense as follows

$$\min_{\boldsymbol{\phi}\in\mathcal{V}} E_\mu(\boldsymbol{\phi}) = \frac{E(\boldsymbol{\phi})}{\|1\|^2_{\mathcal{M}_I}} + \mu\frac{\|\boldsymbol{T}\boldsymbol{\phi} - \mathbf{g}_D(\boldsymbol{T}\boldsymbol{\phi})\|^2_{\partial\mathcal{M}_I}}{\|1\|^2_{\partial\mathcal{M}_I}}, \tag{4}$$

where μ is a penalty parameter that enforces the validity of the constraint when it tends to infinity. We have introduced the measures of the initial mesh and its boundary in order to balance the two contributions of the new functional.

The main idea is to solve several unconstrained optimization problems with increasing penalty parameter in order to enforce the boundary condition. Nevertheless, the boundary condition depends on the actual solution of the problem. Thus, we apply a fixed-point iteration as

$$\mathbf{g}_D^k = \mathbf{g}_D\Big(\boldsymbol{T}\boldsymbol{\phi}^k\Big),$$
$$\boldsymbol{\phi}^{k+1} = \underset{\boldsymbol{\phi}\in\mathcal{V}}{\arg\min}\, E_\mu\Big(\boldsymbol{\phi}; \mathbf{g}_D^k\Big),$$

being k the k-th iteration of the proposed fixed-point solver.

We optimize each non-linear problem of the proposed penalty method using a backtracking line-search method in which the advancing direction is computed using Newton's method and the step-length is set using the Armijo's rule, see [48] for more details.

3.3 p-Continuation Technique

To improve the robustness of the proposed solver and to compute an initial condition for the non-linear solver, we propose to apply a p-continuation technique. Instead of directly computing the optimal mesh for a given polynomial degree, we iterate through the polynomial degrees and optimize them. The initial condition for each polynomial degree is the optimized mesh of the previous one. There are two main new contributions in our p-continuation technique.

The first contribution is an early termination criterion to stop the optimization of the initial polynomial degrees. The proposed early termination criterion reduces the computational cost of the full optimization process. To this end, we consider the mesh of the current polynomial degree, $\boldsymbol{\phi}^p$, and let $\boldsymbol{\phi}^{p+1}$ be the interpolation of $\boldsymbol{\phi}^p$ using element-wise polynomials of degree $p+1$. The early termination criterion is defined using the boundary condition error of both meshes as

$$\alpha \left\|\boldsymbol{T}\boldsymbol{\phi}^p - \mathbf{g}_D(\boldsymbol{T}\boldsymbol{\phi}^p)\right\| < \left\|\boldsymbol{T}\boldsymbol{\phi}^{p+1} - \mathbf{g}_D\left(\boldsymbol{T}\boldsymbol{\phi}^{p+1}\right)\right\|. \tag{5}$$

That is, the optimization process of the current polynomial degree is finished when the error of the boundary condition is *comparable* to the error of the boundary condition of the next polynomial degree. In this work, we take $\alpha = 2$.

The second contribution is the calculation of the initial penalty parameter for the optimization of each polynomial degree. A correct value of the penalty parameter facilitates the solution of the linear and non-linear problems and therefore, increases the robustness of the optimization process. For a sufficiently large penalty parameter, after optimizing the functional in Equation (4), the Lagrange multipliers of the associated constraint are approximated as

$$\lambda^p \simeq -2\mu^p \frac{\left\|\boldsymbol{T}\boldsymbol{\phi}^p - \mathbf{g}_D(\boldsymbol{T}\boldsymbol{\phi}^p)\right\|}{\|1\|_{\partial\mathcal{M}_I}}. \tag{6}$$

When the early termination criteria is satisfied, we interpolate $\boldsymbol{\phi}^p$ using polynomials of degree $p+1$, and compute the associated boundary condition of $\boldsymbol{\phi}^{p+1}$. Assuming that the associated Lagrange multipliers approximated using Equation (6) are similar for both polynomial degrees, we consider

$$\lambda^p = \lambda^{p+1}.$$

Therefore, we obtain that

$$\mu^{p+1} = \mu^p \frac{\left\|\boldsymbol{T}\boldsymbol{\phi}^p - \mathbf{g}_D(\boldsymbol{T}\boldsymbol{\phi}^p)\right\|}{\left\|\boldsymbol{T}\boldsymbol{\phi}^{p+1} - \mathbf{g}_D(\boldsymbol{T}\boldsymbol{\phi}^{p+1})\right\|}. \tag{7}$$

3.4 Parallel Pre-Conditioned Linear Solver

At each iteration of the non-linear solver we solve the following linear system derived from Newton's method:

$$\mathbf{H}E_\mu(\mathbf{u})\delta\mathbf{u} = -\nabla E_\mu(\mathbf{u})$$

Since in this work we are dealing with large-scale meshes, we assemble and solve the linear system in a parallel framework. We use the FEniCS software to automatically generate the derivative expressions required to evaluate the Hessian and the gradient. Since we solve this problem in parallel, we need to distribute the mesh elements and degrees of freedom. To this end, we use the ParMETIS library. The resulting distribution allows computing the element contributions in a distributed manner. Specifically, we use the FEniCS software, where each elemental contribution to the residual and the residual Jacobian are computed in parallel. To this end, non-blocking communication is performed to communicate to send and receive the required data from adjacent processors. Then the contributions of the inner processor elements are computed and overlapped with the previously non-blocking communication. Then, the contributions from the processor boundary elements are computed. Finally, the residual and the residual Jacobian are assembled.

We have used the linear solvers and pre-conditioners implemented in the PETSc library [49]. We solve the linear systems using a GMRES method restarted every 20 iterations. To accelerate the convergence of the linear solver, we consider two pre-conditioners. The first one is a restricted additive Schwarz method with one overlap level. The local problems are approximated using two iterations of a symmetric successive over-relaxation method with $\omega = 1$. The second pre-conditioner is an algebraic multi-grid method in which the coarsest level is solved using an LU decomposition. The number of coarsening levels is automatically computed by the PETSc implementation.

3.5 Algorithm Description

Algorithm 1 describes the proposed penalty method with p-continuation technique for mesh curving. The input of the algorithm is an initial linear mesh $\mathcal{M}_I$, the final polynomial degree, $p_{\max}$, and the required tolerances for the boundary condition and the non-linear problem, ε^* and ω^*, respectively. In Lines 2–4, we set the initial polynomial degree, and we initialize the quadratic mesh, $\boldsymbol{\phi}^2$, to the identity mapping. That is, we introduce the additional nodes on the edges of the mesh, and we keep the straight-edged elements. The identity mapping is optimal with respect of the mesh quality however, it does not satisfy the boundary condition. The initial penalty parameter is set to 10. In Line 5 we start the p-continuation loop, in which we iterate through all the polynomial degrees. Then, in Line 7, we start the penalty method for the given polynomial degree. In Lines 8 and 9 we perform the fixed-point iteration. First, we update the boundary condition and then, we optimize the functional in Equation 4 to compute the new approximation of the optimal mesh. In Lines 10–16 we select the convergence criterion according to the current polynomial degree. If the convergence check passes and we are optimizing the last polynomial degree, the algorithm ends, Line 19. If the convergence check passes and we are not in the last polynomial degree, we compute the new penalty parameter, Line 21, according to Equation (7), and we perform the optimization of the next polynomial degree. If the convergence check fails, Line 24, we increase the penalty parameter and tighten the tolerances.

Algorithm 1
p-continuation penalty method for mesh curving.

Input: $\mathcal{M}_I$, $p_{\max}$, ε^*, ω^*
Output: Mapping $\boldsymbol{\phi}^p$

1: **function** `meshOptimization`

▷ Variable initialization
2: $p \leftarrow 2$
3: $\boldsymbol{\phi}^p \leftarrow \mathbf{Id}$
4: $\mu \leftarrow 10$, $m \leftarrow 10$, $\alpha \leftarrow 2$

▷ p-continuation loop
5: **for** $p \in 2, \dots, p_{\max}$ **do**
6: conv $\leftarrow$ **false**

▷ Penalty method loop
7: **while not** conv **do**

▷ Fixed-point iteration
8: $\mathbf{g}_D^p \leftarrow \mathbf{g}_D(\boldsymbol{T}\boldsymbol{\phi}^p)$
9: $\boldsymbol{\phi}^p \leftarrow$ `optimize`$(E_\mu(\boldsymbol{\phi}^p, \mathbf{g}_D^p), \omega)$

▷ Convergence criterion
10: **if** $p = p_{\max}$ **then**

▷ Convergence criterion for $p_{\max}$
11: conv $\leftarrow \|\boldsymbol{\phi}^p - \mathbf{g}_D^p\| < \varepsilon^*$ **and** $\left\|\nabla E_\mu(\boldsymbol{\phi}^p, \mathbf{g}_D^p)\right\| < \omega^*$
12: **else**

▷ Early termination criterion
13: $\boldsymbol{\phi}^{p+1} \leftarrow$ `interpolate`$(\boldsymbol{\phi}^p)$
14: $\mathbf{g}_D^{p+1} \leftarrow \mathbf{g}_D\left(\boldsymbol{T}\boldsymbol{\phi}^{p+1}\right)$
15: conv $\leftarrow$ $\alpha \|\boldsymbol{T}\boldsymbol{\phi}^p - \mathbf{g}_D^p\| < \left\|\boldsymbol{T}\boldsymbol{\phi}^{p+1} - \mathbf{g}_D^{p+1}\right\|$
16: **end if**

▷ Check convergence
17: **if** conv **then**
18: **if** $p = p_{\max}$ **then**

▷ Algorithm has finished
19: **return** $\boldsymbol{\phi}^p$
20: **else**

▷ μ calculation for next q
21: $\mu \leftarrow \mu \frac{\|\boldsymbol{\phi}^p - \mathbf{g}_D^p\|}{\|\boldsymbol{\phi}^{p+1} - \mathbf{g}_D^{p+1}\|}$
22: $\boldsymbol{\phi}^{p+1} \leftarrow \boldsymbol{\phi}^p$
23: **end if**
24: **else**

▷ Not converged, increase penalty and tighten tolerance
25: $\mu \leftarrow m\mu$
26: $\omega \leftarrow \max\{\omega/m, \omega^*\}$
27: **end if**
28: **end while**
29: **end for**
30: **end function**

4. EXAMPLES

This section presents several examples that show the capabilities of the presented high-order mesh curving method. Specifically, we show two three-dimensional weak scaling examples, and a large-scale example of a complex geometry with boundary layer.

To generate the initial linear meshes, we have used Pointwise [50]. The mesh curving solver has been implemented in Python [51] using the FEniCS [52] and the petsc4py [49] libraries. To project the boundary high-order nodes we have used both the geode [53] and the Open CASCADE [54] libraries interfaced with an in-house python wrapper developed using swig [55].

The optimization process has been performed in the MareNostrum4 super-computer located at the Barcelona Supercomputing Center. It is composed of 3456 nodes, connected using an Intel Omni-Path network. Each node contains two Intel Xeon Platinum 8160 CPU with 24 cores, each at 2.10 GHz, and 96 GB of RAM memory. To obtain the total energy consumed by all tasks of the optimization job, we have

used the `sacct` command of the SLURM Workload Manager. Note that only in case of exclusive job allocation, this value reflects the real energy consumption.

The mesh visualization has been performed using Paraview 5.5.2 [56] in parallel in the MareNostrum4 supercomputer. We have used the high-order mesh visualization implementation of Paraview that subdivides each element in a given number of sub-elements. Note that the mesh partition to perform the visualization does not need to coincide with the mesh partition to perform the optimization. In general, for visualization purposes, less cores are needed since no global matrices are assembled and no linear systems are solved.

We compute the elemental quality relative to the initial mesh as [23]

$$q_{e_P} = \frac{1}{\eta_{e_P}}, \quad \text{where} \quad \eta_{e_P} = \left(\frac{\int_{e_I} (M\phi)^2 \, \mathrm{d}\Omega}{\int_{e_I} 1 \, \mathrm{d}\Omega} \right)^{1/2} .$$

The relative element quality takes values in the interval $[0, 1]$. An ideal element has quality equal to one, and an inverted or tangled element has a quality of zero. In all the examples, we color the elements according to $1 - q_{e_P}$ in logarithmic scale to check how close is the element quality to one. Thus, lower values denote higher quality elements.

In the examples, we compare the two pre-conditioners presented in Section 3.4. To faithfully compare both pre-conditioners, we solve the linear systems with a relative tolerance of 10^{-9}. Thus, in all the cases, the evolution of the non-linear solver is not affected by the selection of the pre-conditioner, since the solution of the linear systems is numerically equal when different pre-conditioners are used.

In the examples we compare the two pre-conditioners presented in Section 3.4. To faithfully compare both pre-conditioners, we solve the linear systems with a relative tolerance of 10^{-9}. Thus, in all the cases, the evolution of the non-linear solver is not affected by the selection of the pre-conditioner, since the solution of the linear systems is numerically equal when different pre-conditioners are used. Although using both pre-conditioners the optimal meshes are numerically equivalent, the figures show the optimal high-order meshes optimized using the additive Schwarz pre-conditioner.

4.1 Weak Scaling: Isotropic Elements

In this example, we perform a weak scaling analysis on the number of elements. We generate five isotropic meshes increasing the number of elements and the used cores in such a way that the number of elements per core remains constant. The domain is a sphere of radius four with a spherical hole in the center of radius one. The element sizes are chosen such that there are around 1500 elements per cores, and we have used 480, 960, 1440, 1920 and 2400 cores. This leads to meshes that are composed of $0.72 \cdot 10^6$, $1.44 \cdot 10^6$, $2.16 \cdot 10^6$, $2.88 \cdot 10^6$ and $3.60 \cdot 10^6$ elements of polynomial degree four. Figures 1a to 1e show the five curved high-order meshes generated for this example.

Figure 2 show the evolution of the constraint norm during the iterations of the non-linear optimization for the five meshes. At each non-linear iteration, we optimize the functional in Equation (4) and, if needed, the penalty parameter is increased in order to enforce the boundary condition. Dark blue circles denote the initial iteration of each polynomial degree in the p-continuation technique. The evolution of the constraint norm in all cases follows a similar pattern, even though the boundary mesh is not the same. Thus, we show that the proposed formulation presents a mesh independent behavior at the non-linear level. Specifically, the number of iterations to perform the whole optimization process is the same in all the cases and moreover, the number of non-linear iterations at each polynomial degree is also the same. During the first iterations of the quadratic mesh, the constraint norm decreases *slowly*. Nevertheless, from iteration three onwards, the constraint norm decreases geometrically with the non-linear iterations. When the constraint norm is of the same order as the constraint norm of the next polynomial degree, the early-termination criterion is activated. Thus, the norm of the boundary condition is similar to the norm of the boundary condition of the next polynomial degree. Then, we compute the new penalty parameter for the next polynomial degree. Note that the constraint norm also decreases geometrically with the non-linear iterations. Therefore, this shows that we have correctly selected the value of the penalty parameter.

In Figure 3 we present the time to solve all the linear problems, the total number of linear solver iterations, the energy consumption, and the energy consumption per core for the two pre-conditioners. We use the red, green and blue colors to denote the quantities of interest for the meshes of polynomial degree two, three and four of the p-continuation technique. Using both pre-conditioners, the time to curve the finer meshes is larger than the time to curve the coarser meshes, even when the number of elements per core is the same. The number of linear iterations to curve the quadratic mesh is significantly higher than the rest of the curving process, while the time spent on curving the initial polynomial degrees is significantly lower than the rest of the curving process.

While in the case of the additive Schwarz pre-

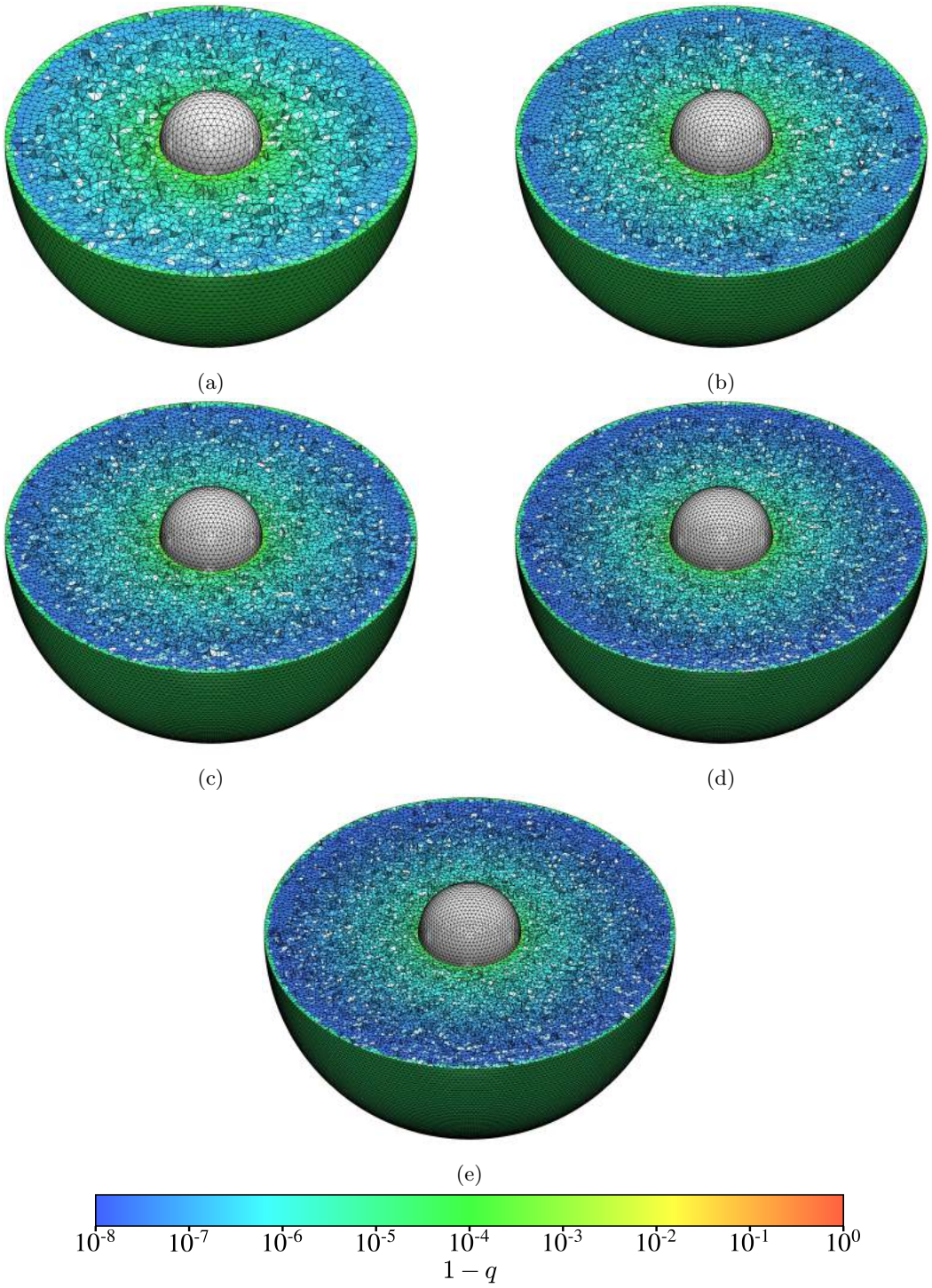

Figure 1: Optimized meshes of polynomial degree four using the proposed mesh curving solver, approximately composed of: (a) $0.72 \cdot 10^6$ elements on 480 cores; (b) $1.44 \cdot 10^6$ elements on 960 cores; (c) $2.16 \cdot 10^6$ elements on 1440 cores; (d) $2.88 \cdot 10^6$ elements on 1920 cores; and (e) $3.60 \cdot 10^6$ elements on 2400 cores.

conditioner the number of iterations of the linear solver increases, this is not the case when using the multi-grid pre-conditioner. However, the cost of each iteration is higher when using the multi-grid pre-

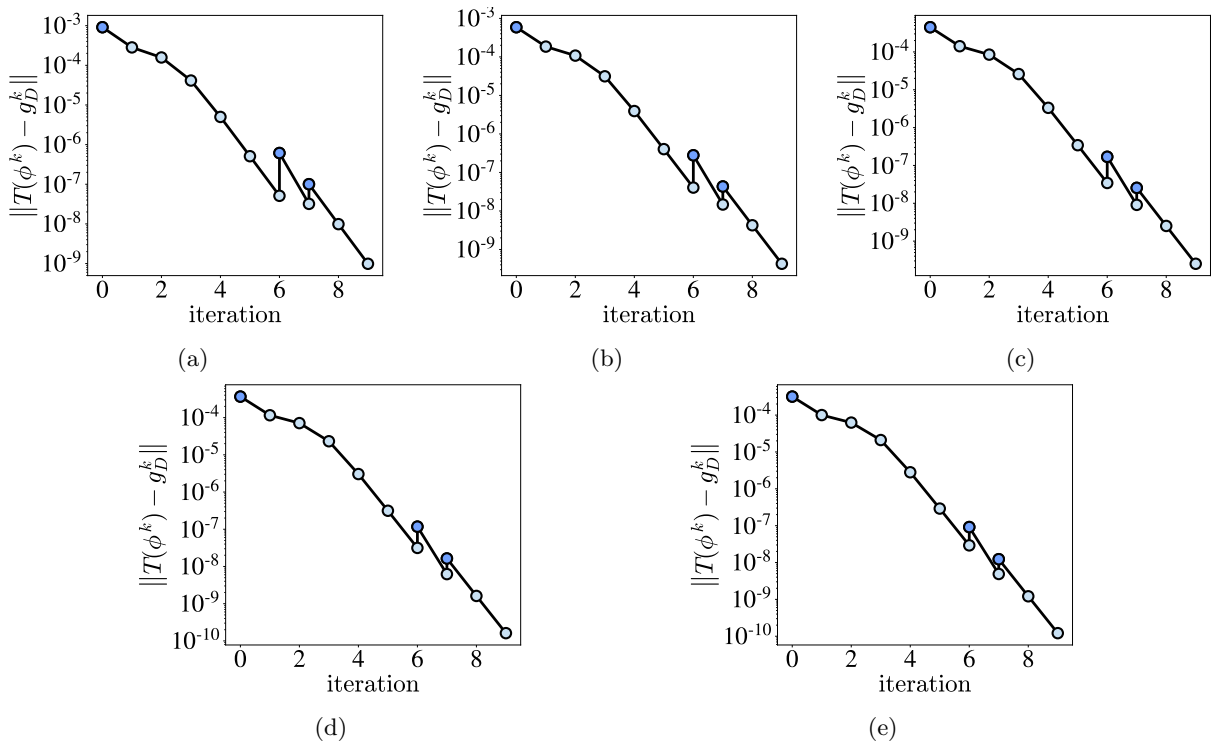

Figure 2: Evolution of the norm of the constraint through the optimization process for the mesh optimized with boundary layer stretching of: (a) 480 cores; (b) 960 cores; (c) 1440 cores; (d) 1920 cores; and (e) 2400 cores.

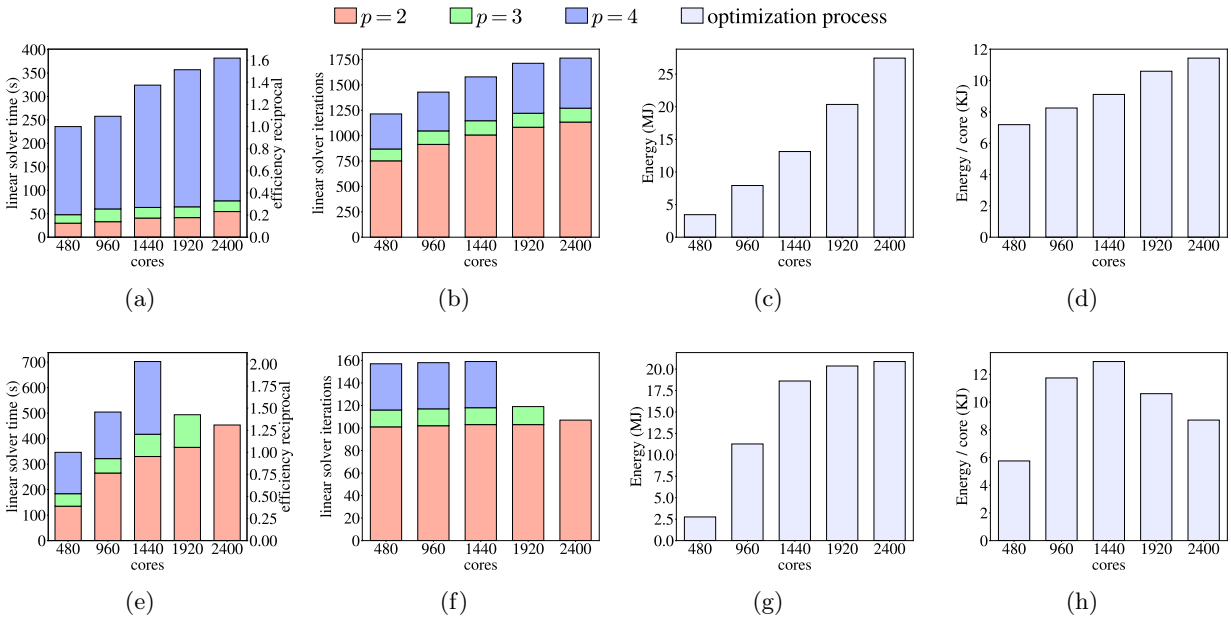

Figure 3: In rows, different pre-conditioners: (a), (b) (c) and (d) additive Schwarz; (e), (f), (g) and (h) multi-grid. In columns, in terms of the number of cores: (a) and (d) linear solver time; (b) and (e) linear solver iterations; (c) and (f) total energy consumption; and (d) and (h) energy consumption per core.

conditioner, and so is the total time of solving the linear systems. The main reason is that the multi-grid pre-conditioner uses more memory per core as the mesh becomes finer. Thus, using the multi-grid pre-conditioner we were not able to curve the last two meshes because of the high amount of used memory. Specifically, when curving the mesh with 1920 cores, we only obtained the cubic mesh, and when curving the mesh using 2400 cores, we reached the quadratic mesh.

Using both pre-conditioners, the total amount of consumed energy increases as the number of cores increase. This is expected because when curving finer meshes we are using more cores during more time. In addition, the memory consumption per core increases as the number of elements in the meshes increases. That is, when curving finer meshes, each core consumes more energy than when curving coarser meshes. Since multi-grid uses more computational resources than the additive Schwarz pre-conditioner, the multi-grid pre-conditioner leads to higher energy demands than the additive Schwarz pre-conditioner.

4.2 Weak Scaling: Boundary Layer Stretching

In this example, we perform a weak scaling analysis in terms of the stretching of the boundary layer and the number of elements. The domain is a sphere of radius four with a spherical hole in the center of radius one. We generate five meshes increasing the stretching and the number of layers of a boundary layer generated around the inner sphere. In all the meshes, the boundary mesh is the same. The growth factor of all the boundary layers is $10^{0.1} \simeq 1.259$, that ensures that every ten layers, the width of the layer is multiplied by ten. The maximum stretching of each mesh is $1:10^1$, $1:10^2$, $1:10^3$, $1:10^4$, $1:10^5$. The number of layers in the boundary layer has been chosen to obtain roughly 1500 elements per processor, and we have used 96, 192, 288, 384 and 480 cores. Specifically, the number of layers of each mesh is 10, 23, 37, 47 and 57. This leads to meshes that are composed of $135 \cdot 10^3$, $291 \cdot 10^3$, $460 \cdot 10^3$, $581 \cdot 10^3$ and $702 \cdot 10^3$ elements. Figures 4a to 4e show the five curved high-order meshes of this example.

Figure 5 show the evolution of the constraint norm over the non-linear iterations for the five cases. In this example, the boundary mesh is the same for all the cases and therefore, the evolution for the five meshes is practically the same. That is, the proposed solver exhibits mesh independence at the non-linear level. Moreover, in this example, the constraint norm decreases geometrically with the non-linear iterations. This is especially important during the first iterations of each polynomial degree in the p-continuation technique. The main reason is that we compute a *correct* value of the penalty parameter. Thus, the non-linear solver can perform the continuation of the solution when increasing the polynomial degree.

Figure 6 shows the time to solve the linear problems, the total number of linear solver iterations, the energy consumption, and the energy consumption per core for the two pre-conditioners. In both cases, the time to solve the linear systems becomes larger as the boundary layer stretching increases. The problem becomes more difficult to solve because the high-stretched elements increase the condition number of the linear systems. That is, the number of linear solver iterations increases with the boundary layer stretching. Similarly as in the previous example, the mesh curving process spends most of the linear solve iterations in the curving of the quadratic and cubic meshes. Nevertheless, when using the additive Schwarz pre-conditioner, the curving of the quadratic and cubic meshes is a small fraction of the total time. While the multi-grid pre-conditioner uses fewer linear solver iterations than the additive Schwarz one, the time to solve the linear systems is one order of magnitude lower when using the additive Schwarz pre-conditioner.

As we increase the stretching of the boundary layer, the energy consumption increases with both pre-conditioners. Nevertheless, the energy consumption when curving the meshes using the multi-grid pre-conditioner is roughly twenty times larger than when using the additive Schwarz one. Moreover, the energy consumption per core also increases when we increase the element stretching. The main reason is that the cores are computing during more time because the linear solver needs more iterations to converge. In this example, as expected, we show with numerical evidence that the problem of high-order mesh curving becomes more difficult as the elements become more stretched.

4.3 p-continuation Influence: Complex Geometry with Boundary Layer

We apply the proposed mesh curving solver to curve a large-scale mesh generated for a falcon aircraft, and we show the advantages of the proposed p-continuation technique. The initial linear mesh contains $3.91 \cdot 10^6$ elements, and a boundary layer around the aircraft with a maximum stretching of $1:400$. We perform the high-order mesh curving process with and without using the proposed p-continuation technique. Specifically, we apply the proposed solver to curve a mesh of polynomial degree four that contains $42 \cdot 10^6$ nodes using 2400 cores. Figure 9 shows the decomposition that we have used to perform the parallel mesh curving optimization process. We obtain the same solution whether we apply the proposed p-continuation tech-

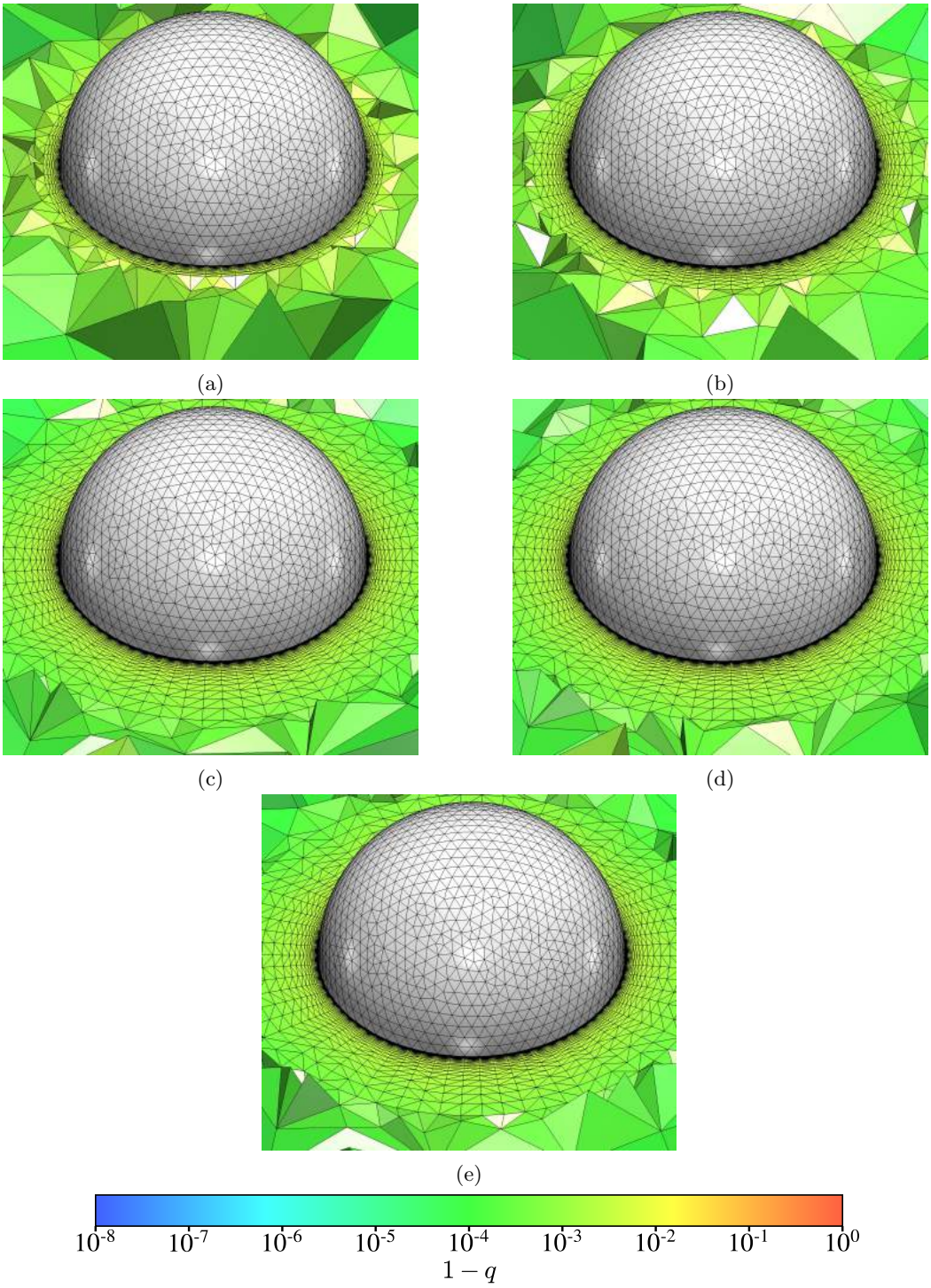

Figure 4: Optimized meshes using the proposed mesh curving solver with boundary layer stretching of: (a) $1:1\cdot10^{1}$; (b) $1:1\cdot10^{2}$; (c) $1:1\cdot10^{3}$; (d) $1:1\cdot10^{4}$; and (e) $1:1\cdot10^{5}$.

nique or not. Figure 7 shows two global views of the curved high-order mesh, while Figures 8a and 8b show detailed views of the mesh at the nose and at the wing-fuselage structure. Note that just the first layers close to the aircraft are curved and that the majority of the mesh contains straight-edged elements.

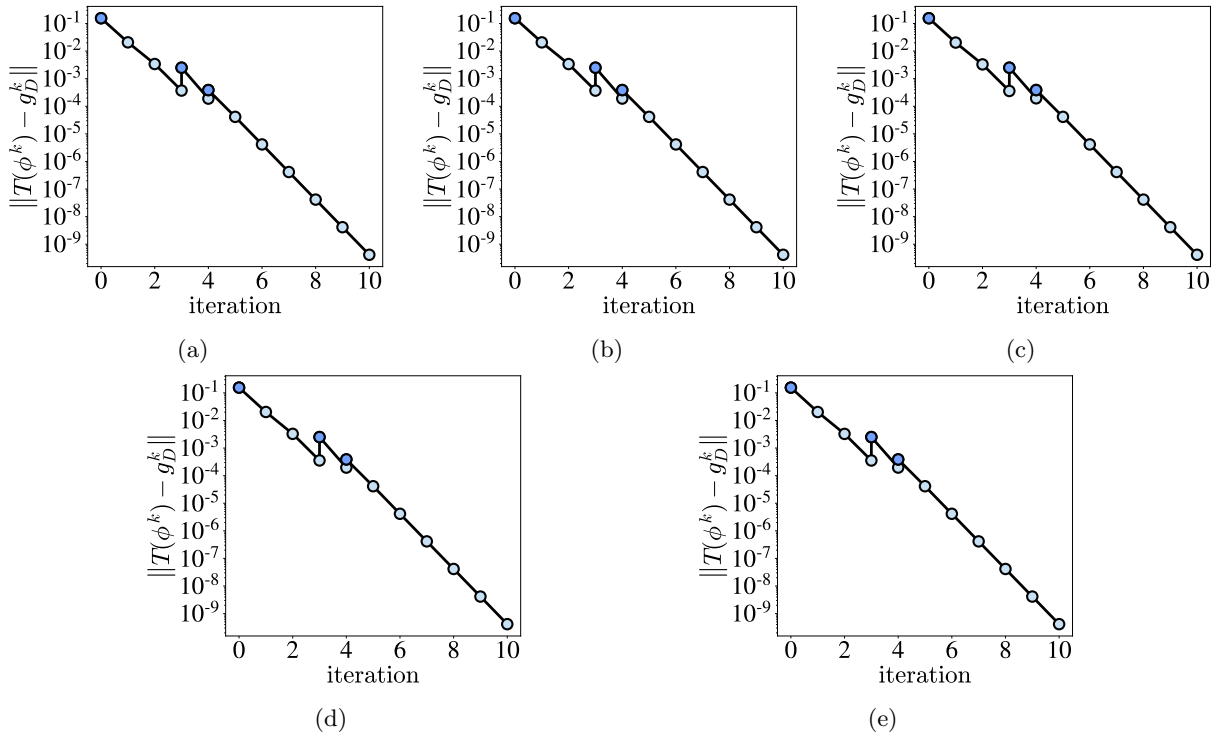

Figure 5: Evolution of the norm of the constraint through the optimization process for the mesh optimized with boundary layer stretching of: (a) $1:10^1$; (b) $1:10^2$; (c) $1:10^3$; (d) $1:10^4$; and (e) $1:10^5$.

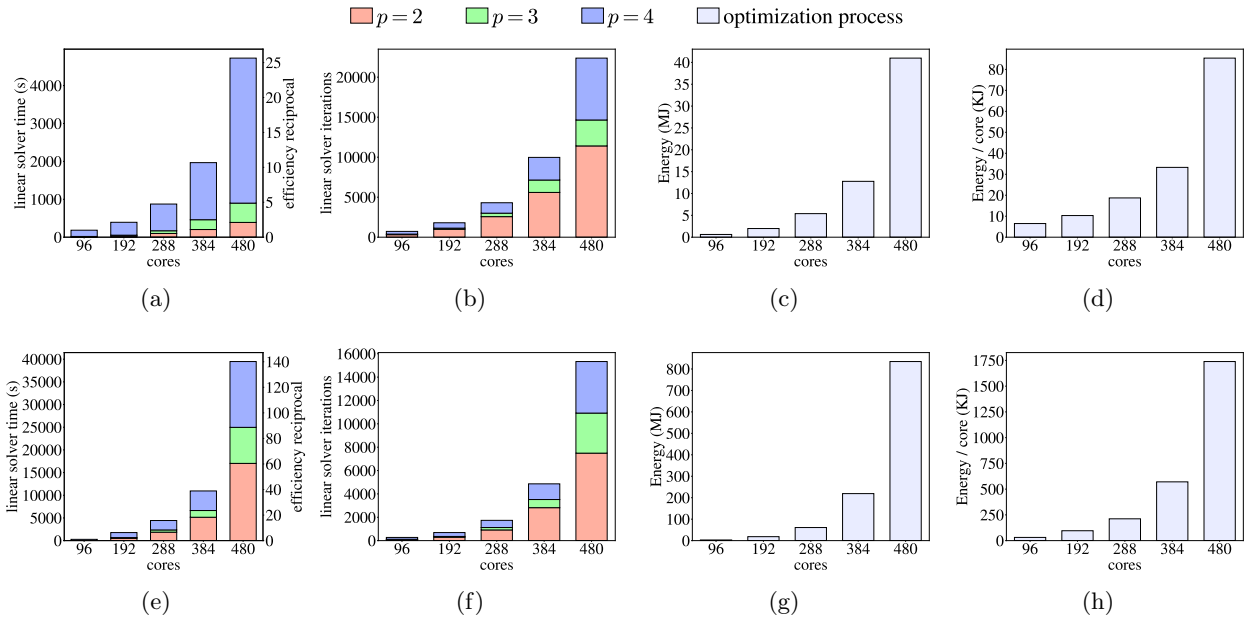

Figure 6: In rows, different pre-conditioners: (a), (b) (c) and (d) additive Schwarz; (e), (f), (g) and (h) multi-grid. In columns, (a) and (d) linear solver time; (b) and (e) linear solver iterations; (c) and (f) total energy consumption; and (d) and (h) energy consumption per core.

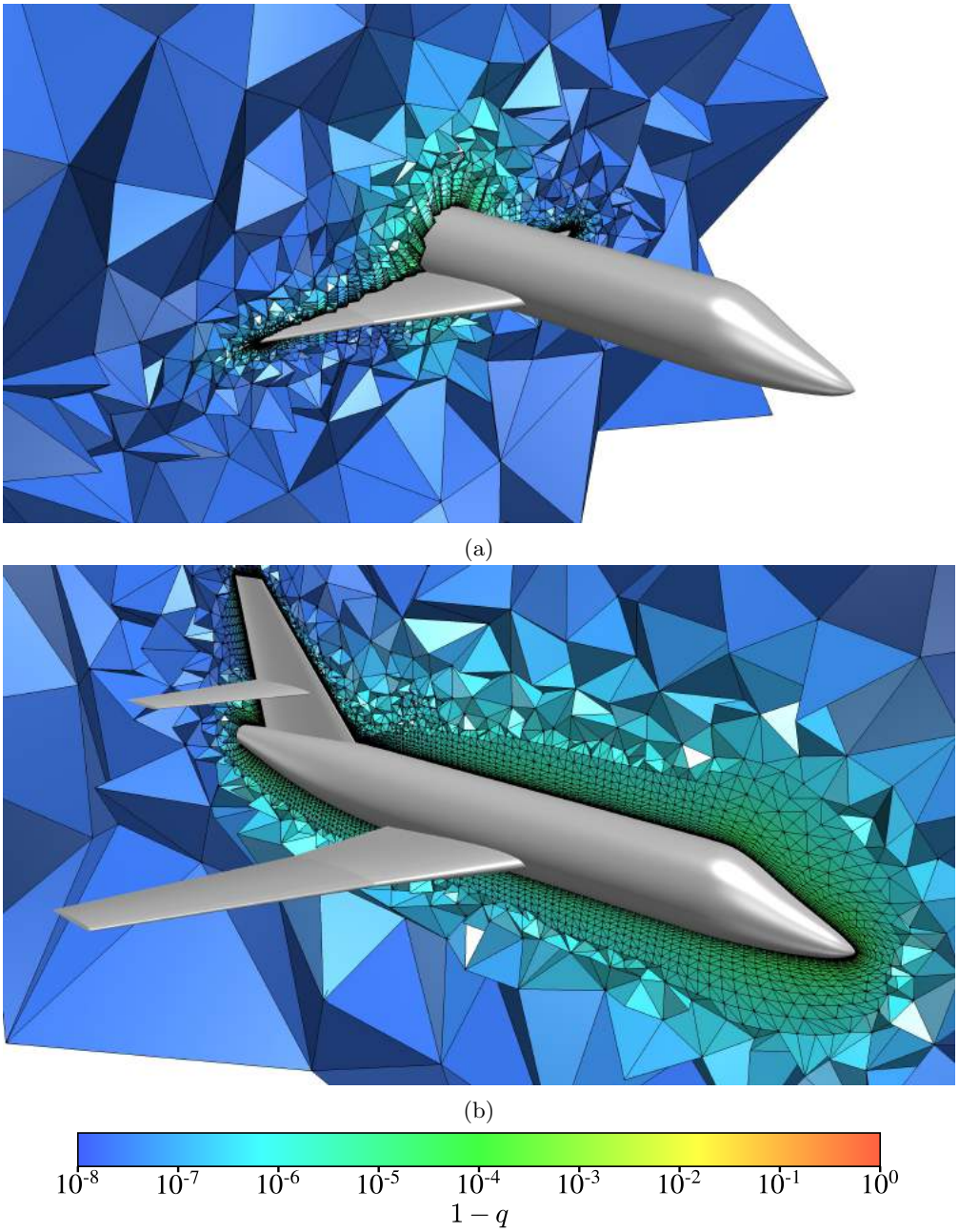

Figure 7: Optimized mesh of polynomial degree four of a falcon aircraft using the proposed mesh curving solver: (a) slice along the x axis; and (b) slice along the y axis.

Figures 10a and 10b show the evolution of the constraint norm through the non-linear iterations of the penalty method with and without using the proposed p-continuation technique, respectively. The dark blue circles denote the initial iteration of each polynomial degree. When using the p-continuation technique, the whole process takes nine iterations to converge, while the case of directly optimizing the mesh of polynomial degree four takes 7 iterations. In both cases, in the last iterations of each polynomial degree, the constraint norm decreases geometrically with the number of iterations. Note that when using the p-continuation technique the estimation of the penalty parameter ensures that in the first iterations of each polynomial degree the constraint norm also decreases geometrically.

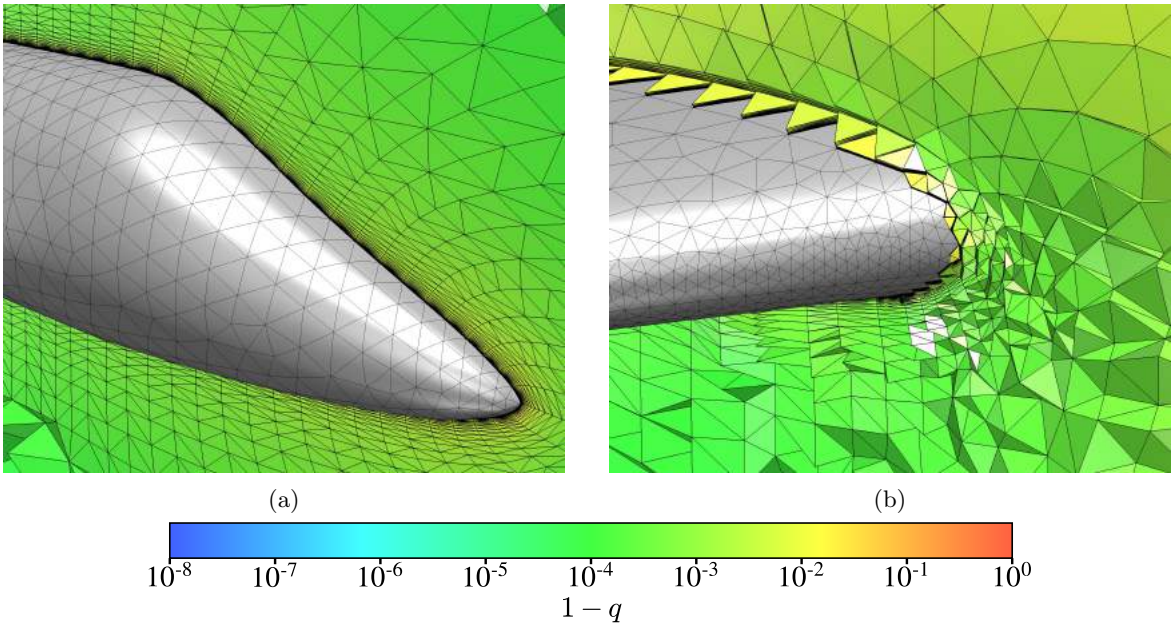

Figure 8: Detailed view of the optimized mesh of polynomial degree four of a falcon aircraft using the proposed mesh curving solver: (a) at nose; and (b) at the wing-fuselage.

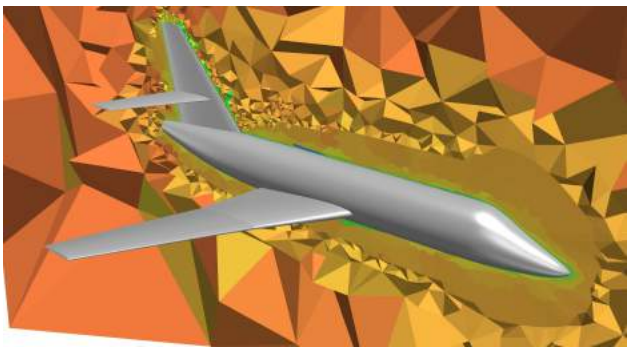

Figure 9: Parallel decomposition of the high-order mesh for a Falcon aircraft, where each color denotes a different processor.

Figure 11 shows the total time to solve the linear problems, the number of linear solver iterations, and the energy consumption for both cases. Note that the p-continuation technique reduces the time to solve the linear systems fourfold. The main reason is that we reduce in half the number of linear solver iterations and that only a fraction of the linear iterations are performed in the mesh of polynomial degree four. Since meshes of lower polynomial degree lead less unknowns and to matrices with fewer non-zero entries, each iteration of the linear solver is performed faster. For this reason, we have reduced the computational time and memory requirements with the proposed p-continuation technique and therefore, the energy consumption is reduced five times.

Table 1 presents the time breakdown of the optimization process with and without the proposed p-continuation technique. In both cases, the time to solve linear systems is the major contribution of the total time. The assembly time is lower when using the p-continuation technique because the elemental matrices are smaller. The rest of the time contains the mesh reading and writing, the nodal projection, and the control flow of the algorithm. In both cases, this time is similar. Thus, the main advantage of the proposed p-continuation technique is that we reduce the time to construct the linear systems and to solve them.

5. CONCLUDING REMARKS

5.1 Summary

We have presented a mesh curving solver to generate curved high-order meshes. Specifically, we have extended our high-order meshing solver to take advantage of a distributed parallel environment. The proposed formulation shows mesh independence at the non-linear level. Nevertheless, the obtained linear systems do not show this behavior. As the number of elements increase and the elements become more stretched, the linear systems are harder to solve because the condition number of the matrices increases. It is especially important to highlight the difficulty

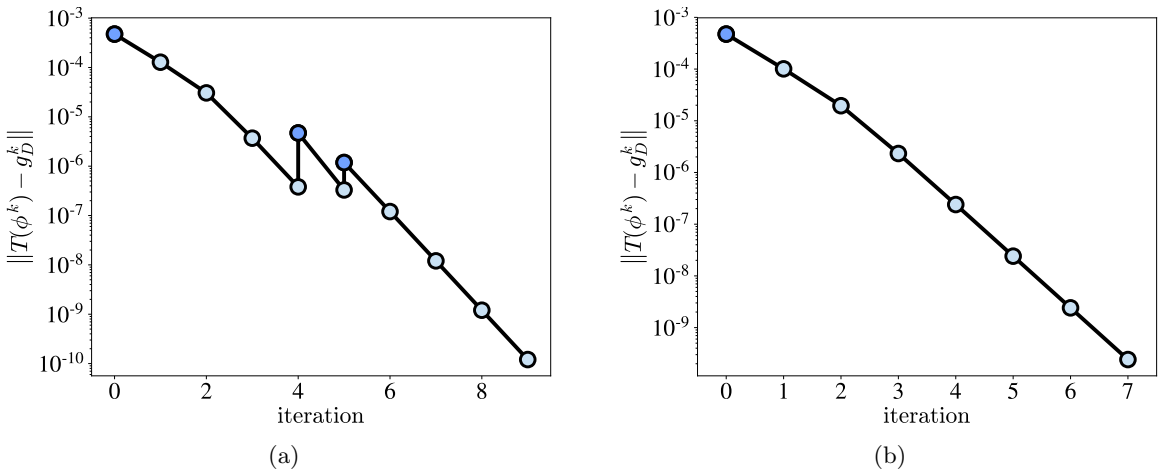

Figure 10: Evolution of the norm of the constraint through the high-order mesh curving process of a falcon aircraft.

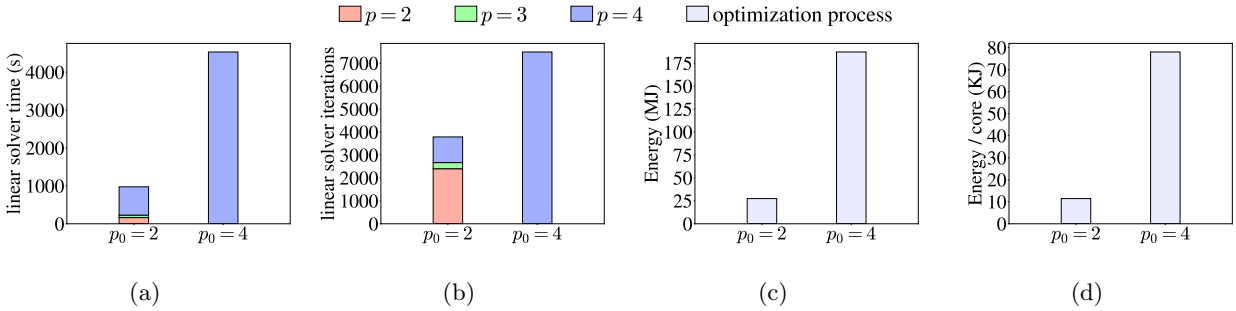

Figure 11: Curved high-order mesh generation for a falcon aircraft: (a) linear solver time; (b) linear solver iterations; and (c) energy consumption

Table 1: Time breakdown in seconds for the meshes generated of the Falcon aircraft

Case	Total time (s)	Assembly time (s)	Linear solver time (s)	other time (s)
$p_0 = 2$	1296	289.5	975.2	31.19
$p_0 = 4$	4921	355.7	4531	34.48

increase when the elements become more stretched. Thus, the main difficulty of the mesh curving process is solving the linear systems, especially when stretched elements appear in the mesh.

In this work, we have analyzed two pre-conditioners in order to compare their performance in terms of the number of linear solver iterations, the time to solve the linear systems, and the energy consumption of the mesh curving process. The first pre-conditioner is an additive Schwarz method with one level of overlap in which the local problem is approximated using two iterations of a symmetric successive over-relaxation method. The second pre-conditioner is an algebraic multi-grid pre-conditioner in which the coarsest problem is solved using an LU decomposition. The PETSc implementation of the algebraic multi-grid method automatically computes the optimal number of levels to solve the linear problem.

The additive Schwarz pre-conditioner leads to a higher number of linear iterations than the algebraic multi-grid. Nevertheless, each iteration of the algebraic multi-grid pre-conditioner takes more time than one iteration of the additive Schwarz one. Thus, the total time to solve the linear systems is lower when using the additive Schwarz pre-conditioner. Moreover, the algebraic multi-grid technique has to store the different coarsening levels and therefore it uses more memory than the additive Schwarz. Specifically, as the mesh becomes finer and the number of cores increases, the required memory per core increases. Moreover, even

when keeping the same number of elements per core, the memory requirements make the algebraic multi-grid method not usable for meshes with a high number of elements. Therefore, since the algebraic multi-grid method requires more memory and takes more time than the additive Schwarz method, the energy consumption is higher. Specifically, the economic cost of generating a curved high-order mesh is lower when using the additive Schwarz pre-conditioner.

We have proposed a p-continuation technique to reduce the computational resources of the mesh curving solver while increasing its robustness. The main idea is to use the solution of a polynomial degree as a starting position for the next polynomial degree. Therefore, the p-continuation technique can be interpreted as a methodology to compute an initial position of the mesh for the last polynomial degree. Since the initial position of the nodes for the last polynomial degree are close to the optimal position, the computational time to optimize the mesh is reduced and the robustness of the optimization process is increased.

We have devised two key ideas to define the p-continuation technique. The first one is an early-termination criterion to finalize the mesh curving of the initial polynomial degrees. Therefore, we solve less linear problems and we improve the computational efficiency. The second idea is to compute an estimation of the penalty parameter for the initial iteration of each polynomial degree. In our experience, the selection of the penalty parameter is crucial for the efficiency and convergence of the mesh curving process. On the one hand, if the penalty parameter is too low, the boundary condition is not enforced *enough*, and the mesh curving process needs additional iterations of the penalty method. On the other hand, if the penalty parameter is too high, the condition number of the linear systems increase and therefore, the linear systems become harder to solve. Thus, the mesh curving process may not finalize because the linear problem could not be solved.

We have shown that the p-continuation technique does not spend much time curving the initial polynomial degrees, although most of the iterations are performed in the initial polynomial degrees. Therefore, we accelerate the mesh curving solver because the linear solver iterations of lower polynomial degree are faster since the number of unknowns and the number of non-null entries in the matrix are lower. Moreover, in the presented examples, we have shown that the number of linear solver iterations is also reduced. Therefore, the necessary energy to generate a curved high-order mesh is reduced because we are using the computational resources during less time, and the process is less memory intensive.

5.2 Discussion

In our solver, we have tested additional pre-conditioners. For instance, when using an additive Schwarz pre-conditioner without overlap levels, the linear systems do not converge. The main reason is that it takes more linear solver iterations to transfer the information of one processor to the others. This is solved by increasing the overlap levels of the additive Schwarz method. Additionally, the local problem can be approximated using an incomplete LU factorization without fill-in levels, ILU(0). In this case, the linear solver does not converge because the local problem is not well approximated. Although we can increase the fill-in levels, this would lead to higher memory requirements. Finally, to reduce the memory requirements of the algebraic multi-grid it is possible to limit the number of coarsening levels to reduce the memory requirements. Nevertheless, since the coarsest level solver is a direct decomposition solver, its memory requirements increase. Therefore it is needed to find a balance between the number of coarsening levels and the memory requirements of the coarsest level solver.

In this work, we are performing a full optimization process of the whole mesh. Although this approach may seem expensive, we ensure that the final curved high-order mesh is optimal in all the elements. This is an important point because non-optimized elements may introduce spurious oscillations in the solution of a simulation process. Nevertheless, there are implementations that only optimize the worst quality elements. Specifically, they optimize the elements with quality lower than a given threshold, and some additional elements around to increase the feasible locations of the high-order nodes. Therefore, these methods require to select the threshold value in such a manner that the optimized mesh is good enough for the simulation, and how many elements to additionally optimize in order to obtain a feasible solution. By optimizing the whole mesh, we ensure the optimality of all the elements without additional parameters.

The optimization of the whole mesh is especially important in cases with a highly-stretched boundary layer. In these cases, it is of major importance to obtain optimal elements in the boundary layer and therefore the whole boundary layer should be optimized. Since the number of elements in the boundary layer can be a significant fraction of the total elements, it is not clear the reduction in computational resources of optimizing only a fraction of the elements.

Although the existent literature comparing local and global mesh optimization does not deal with piecewise polynomial curved meshes and highly stretched elements, it suggests that for mesh curving a specific-purpose global optimization method might be preferred. That is, existent literature in local and global

optimization methods for linear meshes shares a common conclusion, when highly optimized and accurate meshes are required, especially in isotropic meshes featuring high gradations of the element size, a specific-purpose global feasible Newton method outperforms local optimization methods. This setting also corresponds to the general mesh curving problem where we need to exploit the quadratic convergence of Newton's method since we want high-precision to both approximate the curved geometries and deal with highly stretched elements. Furthermore, we want to apply our mesh curving method to meshes featuring smaller size close the objects immersed in the fluid stream and bigger size in the far-field, and thus, we need to deal with high gradations of the element size. Nevertheless, in the near future, it could be interesting to compare our global parallel method featuring global quadratic convergence rate with a local parallel method featuring local quadratic convergence rate.

Note that we have only studied weak scaling results, but not strong scaling results, since we are interested in proposing a global method to generate large curved meshes in parallel. In our formulation, we need to store the Hessian pre-conditioner in a distributed fashion, and thus, these memory requirements determine the number of computing nodes. This need is so since each computing node has a fixed amount of main memory to deal with a maximum number of non-zero entries of the Hessian matrix, determined by the polynomial degree and the number of elements per processor. We prefer to use all the memory available in each computing node and thus, use a small number of processors for coarse meshes and a larger number of processors for fine meshes. This setting demands a weak scaling study, where instead of fixing the problem size and using more computing resources, we increase both the problem size, number of elements to curve, and the number of processors.

The process of generating a curved high-order mesh given a CAD model is decomposed in three stages. The first stage is the geometry healing and defeaturing of a given CAD model. The second stage is the generation of an initial linear mesh with elements of the desired shape and size. Finally, the third stage is the actual high-order mesh curving by optimizing the mesh distortion. In our particular case, we spend most of the time in the first two stages. Specifically, the first two stages require the work of trained personnel and specialized software.

Usually, if the initial linear mesh is *good* enough, the optimization process is performed without incident. In our experience, the high-order mesh needs enough resolution to represent the underlying CAD geometry. In addition, to avoid invalid elements, there should not be any elements with all the nodes on a smooth surface. Such elements have two high-order triangles that approximate the surface. Since the surface is smooth, this would lead to null Jacobians in the common edge.

5.3 Conclusion

The proposed p-continuation approach reduces four times (eight times) the total time (energy) used to curve a whole boundary layer mesh using 2400 cores. The p-continuation accelerates a Newton-GMRES penalty solver equipped with the chosen parallel preconditioner, a restricted additive Schwarz domain decomposition with one level of overlap and local problems approximated with two iterations of symmetric successive over-relaxation.

ACKNOWLEDGMENTS

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 715546. This work has also received funding from the Generalitat de Catalunya under grant number 2017 SGR 1731. The work of Xevi Roca has been partially supported by the Spanish Ministerio de Economía y Competitividad under the personal grant agreement RYC-2015-01633.

References

[1] Szabó B., Babuška I. *Finite Element Analysis*. John Wiley & Sons New York, 1991

[2] Schwab C. *p-and hp-finite element methods: Theory and applications in solid and fluid mechanics*. Clarendon Press Oxford, 1998

[3] Deville M., Fischer P., Mund E. *High-order methods for incompressible fluid flow*, vol. 9. Cambridge University Press, 2002

[4] Hesthaven J., Warburton T. *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*. Texts in Applied Mathematics. Springer, 2007

[5] Karniadakis G., Sherwin S. *Spectral/hp element methods for computational fluid dynamics*. Oxford University Press, 2013

[6] Vos P.E., Sherwin S., Kirby R. "From h to p efficiently: implementing finite and spectral/hp element methods to achieve optimal performance for low- and high-order discretisations." *J. Comput. Phys.*, vol. 229, no. 13, 5161–5181, 2010

[7] Cantwell C., Sherwin S., Kirby R., Kelly P. "From h to p efficiently: strategy selection for operator evaluation on hexahedral and tetrahedral

elements." *Comput. Fluids*, vol. 43, no. 1, 23–28, 2011

[8] Cantwell C., Sherwin S., Kirby R., Kelly P. "From h to p efficiently: selecting the optimal spectral/hp discretisation in three dimensions." *Math. Model. Nat. Phenom.*, vol. 6, no. 3, 84–96, 2011

[9] Löhner R. "Error and work estimates for high-order elements." *Int. J. Numer. Meth. Fluids*, vol. 67, no. 12, 2184–2188, 2011

[10] Yano M., et al. *An optimization framework for adaptive higher-order discretizations of partial differential equations on anisotropic simplex meshes*. Ph.D. thesis, Massachusetts Institute of Technology, 2012

[11] Kirby R., Sherwin S., Cockburn B. "To CG or to HDG: a comparative study." *J. Sci. Comput.*, vol. 51, no. 1, 183–212, 2012

[12] Huerta A., Roca X., Angeloski A., Peraire J. "Are High-order and Hybridizable Discontinuous Galerkin methods competitive?" *Oberwolfach Rep.*, vol. 9, no. 1, 485 – 487, 2012

[13] Löhner R. "Improved error and work estimates for high-order elements." *Int. J. Numer. Meth. Fluids*, vol. 72, 1207–1218, 2013

[14] Wang Z., Fidkowski K., Abgrall R., Bassi F., Caraeni D., Cary A., Deconinck H., Hartmann R., Hillewaert K., Huynh H., et al. "High-order CFD methods: current status and perspective." *Int. J. Numer. Meth. Fluids*, vol. 72, no. 8, 811–845, 2013

[15] Huerta A., Angeloski A., Roca X., Peraire J. "Efficiency of high-order elements for continuous and discontinuous Galerkin methods." *Int. J. Numer. Meth. Eng.*, vol. 96, 529–560, 2013

[16] Dey S., Shephard M., Flaherty J. "Geometry representation issues associated with p-version finite element computations." *Comput. Meth. Appl. M.*, vol. 150, no. 1–4, 39–55, 1997

[17] Dey S., O'Bara R., Shephard M.S. "Curvilinear mesh generation in 3D." *Comput. Aided Design*, vol. 33, 199–209, 2001

[18] Luo X., Shephard M.S., Remacle J.F., O'Bara R., Beall M., Szabó B., Actis R. "P-version mesh generation issues." *Proc. 11th Int. Meshing Roundtable*, pp. 343–354. Springer Berlin Heidelberg, 2002

[19] Luo X., Shephard M.S., O'Bara R., Nastasia R., Beall M. "Automatic p-version mesh generation for curved domains." *Eng. Comput.*, vol. 20, no. 3, 273–285, 2004

[20] Shephard M.S., Flaherty J.E., Jansen K., Li X., Luo X., Chevaugeon N., Remacle J.F., Beall M., O'Bara R. "Adaptive mesh generation for curved domains." *Appl. Numer. Math.*, vol. 52, no. 2-3, 251–271, 2005

[21] Persson P.O., Peraire J. "Curved Mesh Generation and Mesh Refinement using Lagrangian Solid Mechanics." *Proc. 47th AIAA*. 2009

[22] Moxey D., Green M., Sherwin S., Peiró J. "An isoparametric approach to high-order curvilinear boundary-layer meshing." *Computer Methods in Applied Mechanics and Engineering*, vol. 283, no. 0, 636 – 650, 2015

[23] Gargallo-Peiró A., Roca X., Peraire J., Sarrate J. "Optimization of a regularized distortion measure to generate curved high-order unstructured tetrahedral meshes." *International Journal for Numerical Methods in Engineering*, vol. 103, no. 5, 342–363, 2015. URL `http://dx.doi.org/10.1002/nme.4888`

[24] Moxey D., Ekelschot D., Keskin Ü., Sherwin S., Peiró J. "High-order curvilinear meshing using a thermo-elastic analogy." *Computer-Aided Design*, vol. 72, 130–139, 2016

[25] Fortunato M., Persson P. "High-order unstructured curved mesh generation using the Winslow equations." *Journal of Computational Physics*, vol. 307, 1–14, 2016

[26] Eichstädt J., Green M., Turner M., Peiró J., Moxey D. "Accelerating high-order mesh optimisation with an architecture-independent programming model." *Computer Physics Communications*, vol. 229, 36–53, 2018

[27] Sherwin S., Peiró J. "Mesh generation in curvilinear domains using high-order elements." *Int. J. Numer. Meth. Eng.*, vol. 53, no. 1, 207–223, 2002

[28] Xie Z., Sevilla R., Hassan O., Morgan K. "The generation of arbitrary order curved meshes for 3D finite element analysis." *Comput. Mech.*, vol. 51, 361–374, 2012

[29] Poya R., Sevilla R., Gil A.J. "A unified approach for a posteriori high-order curved mesh generation using solid mechanics." *Computational Mechanics*, vol. 58, no. 3, 457–490, 2016

[30] Sevilla R., Rees L., Hassan O. "The generation of triangular meshes for NURBS-enhanced FEM." *International Journal for Numerical Methods in Engineering*, vol. 108, no. 8, 941–968, 2016

[31] Toulorge T., Geuzaine C., Remacle J.F., Lambrechts J. "Robust untangling of curvilinear meshes." *J. Comput. Phys.*, vol. 254, 8 – 26, 2013

[32] Karman S.L., Erwin J.T., Glasby R.S., Stefanski D. "High-Order Mesh Curving Using WCN Mesh Optimization." *46th AIAA Fluid Dynamics Conference*, p. 3178. 2016

[33] Stees M., Shontz S.M. "A high-order log barrier-based mesh generation and warping method." *Procedia Engineering*, vol. 203, 180–192, 2017

[34] Panitanarak T., Shontz S.M. "A parallel log barrier-based mesh warping algorithm for distributed memory machines." *Engineering with Computers*, vol. 34, no. 1, 59–76, 2018

[35] Ruiz-Gironés E., Roca X. "Imposing boundary conditions to match a CAD virtual geometry for the mesh curving problem." *Proceedings of the 27th International Meshing Roundtable*, pp. 343–361. Springer, 2018

[36] Staten M.L., Owen S.J., Shontz S.M., Salinger A.G., Coffey T.S. "A comparison of mesh morphing methods for 3D shape optimization." *Proceedings of the 20th International Meshing Roundtable*, pp. 293–311. Springer, 2011

[37] Ruiz-Gironés E., Sarrate J., Roca X. "Generation of Curved High-order Meshes with Optimal Quality and Geometric Accuracy." *Procedia Engineering*, vol. 163, 315–327, 2016

[38] Toulorge T., Lambrechts J., Remacle J. "Optimizing the geometrical accuracy of curvilinear meshes." *Journal of Computational Physics*, 2016

[39] Ruiz-Gironés E., Gargallo-Peiró A., Sarrate J., Roca X. "Automatically imposing incremental boundary displacements for valid mesh morphing and curving." *Computer-Aided Design*, 2019

[40] Dobrev V., Knupp P., Kolev T., Mittal K., Tomov V. "The Target-Matrix Optimization Paradigm for High-Order Meshes." *SIAM Journal on Scientific Computing*, vol. 41, no. 1, B50–B68, 2019

[41] Garanzha V., Kudryavtseva L. "Hyperelastic springback technique for construction of prismatic mesh layers." *Procedia engineering*, vol. 203, 401–413, 2017

[42] Karman S. "Curving for Viscous Meshes." *Proceedings of the 27th International Meshing Roundtable*, pp. 303–325. Springer, 2018

[43] Sastry S.P., Shontz S.M. "A parallel log-barrier method for mesh quality improvement and untangling." *Engineering with Computers*, vol. 30, no. 4, 503–515, 2014

[44] Benítez D., Escobar J., Montenegro R., Rodríguez E. "Parallel Performance Model for Vertex Repositioning Algorithms and Application to Mesh Partitioning." *Proc. 27th Int. Meshing Roundtable*, 2018

[45] Ruiz-Gironés E., Roca X., Sarrate J. "High-order mesh curving by distortion minimization with boundary nodes free to slide on a 3D CAD representation." *Computer-Aided Design*, vol. 72, 52–64, 2016

[46] Gargallo-Peiró A., Roca X., Peraire J., Sarrate J. "Distortion and quality measures for validating and generating high-order tetrahedral meshes." *Engineering with Computers*, vol. 31, no. 3, 423–437, 2015

[47] Knupp P.M. "Algebraic mesh quality metrics." *SIAM J. Numer. Anal.*, vol. 23, no. 1, 193–218, 2001

[48] Nocedal J., Wright S. *Numerical optimization*. Springer Verlag, 1999

[49] petsc4py. "PETSc for Python.", 2018. URL `https://bitbucket.org/petsc/petsc4py`

[50] Pointwise Inc. "Mesh Generation Software for CFD — Pointwise, Inc." `http://www.pointwise.com`, 2018

[51] Python Software Foundation. "Python." `http://www.python.org`, 2018

[52] Alnæs M.S., Blechta J., Hake J., Johansson A., Kehlet B., Logg A., Richardson C., Ring J., Rognes M.E., Wells G.N. "The FEniCS Project Version 1.5." *Archive of Numerical Software*, vol. 3, no. 100, 2015

[53] Geode. "Project Geode: Geometry for Simulation." `http://www.pointwise.com/geode/`, 2018

[54] CASCADE O. "Open CASCADE Technology, 3D modeling and numerical simulation." `www.opencascade.org`, 2012

[55] swig. "Simplified Wrapper and Interface Generator.", 2018. URL `https://swig.org`

[56] Ahrens J., Geveci B., Law C. "Paraview: An end-user tool for large data visualization." *The visualization handbook*, vol. 717, 2005