High-order unstructured curved mesh generation using the Winslow equations

Meire Fortunato, Per-Olof Persson*

Department of Mathematics, University of California, Berkeley, Berkeley, CA 94720-3840, USA

Abstract

We propose a method to generate high-order unstructured curved meshes using the classical Winslow equations. We start with an initial straight-sided mesh in a reference domain, and fix the position of the nodes on the boundary on the true curved geometry. In the interior of the domain, we solve the Winslow equations using a new continuous Galerkin finite element discretization. This formulation appears to produce high quality curved elements, which are highly resistant to inversion. In addition, the corresponding nonlinear equations can be solved efficiently using Picard iterations, even for highly stretched boundary layer meshes. Compared to several previously proposed techniques, such as optimization and approaches based on elasticity analogies, this can significantly reduce the computational cost while producing curved elements of similar quality. We show a number of examples in both two and three space dimensions, including complex geometries and stretched boundary layers, and demonstrate the high quality of the generated meshes and the performance of the nonlinear solver.

Keywords: Unstructured high-order meshes, curved mesh generation, Winslow equations

1. Introduction

High-order methods are receiving considerable interest from the computational community because of their potential to achieve higher accuracy with reduced computational cost compared to traditional low-order approaches. The need for curved (or *curvilinear*) meshes comes from the fact that linear elements are not adequate for high-order methods, because of the numerical errors introduced by linear approximations of the domain boundary, which would negate the advantages of using high-order methods. Therefore, curved meshes are essential for the broader adoption of high-order accurate discretizations.

While block structured grid generators have long supported curved boundaries, the extension to unstructured meshes has proved to be more difficult. Unstructured meshes are the preferred choice for a wide range of real-world problems, mainly because of the availability of a number of good methods for automatic tetrahedral mesh generation [17, 21, 25, 16] directly from CAD data. These standard mesh generators produce straight-sided elements, but can be used as a starting point for approximating the physical domain. By adding extra geometric data, the generated meshes can be curved such that the curvilinear elements well approximate the curved boundaries.

For simple cases such as well-resolved isotropic triangular elements, a local approach of simply conforming the boundary of elements which are in contact with the curved boundary will oftentimes generate adequate curved meshes. However, for more complex 3D domains and large unstructured simplex elements, such an approach would in general produce low-quality or inverted elements. In addition, high-order methods are often used for problems which need coarse and highly anisotropic elements, therefore in order to produce

*Corresponding author. Tel.: +1-510-642-6947; Fax.: +1-510-642-8204.
Email addresses: meirefortunato@berkeley.edu (Meire Fortunato), persson@berkeley.edu (Per-Olof Persson)

good quality meshes we need to approach the problem of generating high-order curved meshes in a global fashion.

Previous work on curved mesh generation include Refs. [3, 12, 23], where various algorithms for curvilinear meshing are proposed. These methods identify mesh entities that produce invalid elements, and eliminate these problems by a combination of local mesh refinements, edge and face swaps, and node relocations. In Ref. [24], some further options were proposed, including hybrid meshing using prism elements close to the curved boundary, and a curvature-based refinement procedure. Methods based on a solid mechanics analogy have also been proposed, such as the linear elasticity approaches in Refs. [14, 15, 30, 13] or the Lagrangian nonlinear elasticity method of Ref. [18]. Finally, several authors have recently considered an optimization perspective, where the goal is to maximize a Jacobian-based quality measure which penalizes invalid and distorted elements, see for example Refs. [28, 19, 5, 6].

While many of these methods are highly resistant to inverted elements and typically produce good quality meshes, they can be computationally expensive in the presence of boundary layers. One of the motivations of this work is to develop methods which allow for more efficient solvers. In an attempt to reduce computational costs while not significantly decrease the element quality, we propose the generation of unstructured high-order meshes by solving the classical Winslow equations. These are second-order nonlinear elliptic partial differential equations which are obtained by enforcing the computational coordinates to be harmonic. More specifically, choosing the computational domain (with known coordinates $\boldsymbol{\xi}$) to be a linear approximation of the physical domain (with unknown coordinates $\boldsymbol{x}$); we denote the mapping between the two domains by $\boldsymbol{\xi} = \boldsymbol{\xi}(\boldsymbol{x})$ and $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi})$. The Winslow equations are then obtained by rewriting the equations $\Delta_{\boldsymbol{x}}\boldsymbol{\xi}(\boldsymbol{x}) = 0$ in the computational space; where Δ denotes the Laplace operator (see Fig. 1). These equations are solved using a nonlinear Picard approach to obtain the desired physical coordinates. The continuous form of these equations is known to have the desirable property that it defines a smooth mapping for sufficiently regular boundary deformations. Although there is no guarantee that the approach will produce a non-inverted mapping when discretized on a finite-dimensional function space, it does in general produce well-shaped curved meshes when the mesh resolution is reasonable compared to the level of deformation.

Solving the Winslow equations (or elliptic mesh generation) is a well-known tool used in the generation of structured meshes. Grids based on the Winslow equations are the so-called Laplace or Harmonic grids and were first introduced by A. Winslow [29], and have been studied extensively since them, see for example [27, 26, 10, 2]. While often solved in a finite difference setting, a variational formulation for the equations was derived in [1]. Approaches using finite element and finite volume methods for mesh smoothing were previously developed for example in Refs. [7] and [8, 9]. However, we note that these unstructured extensions are typically used for mesh improvement and smoothing of linear meshes, and not for the curved mesh generation problem.

In this paper, we describe a new continuous Galerkin finite element formulation of the standard Winslow equations, which we use for generation of well-shaped high-order unstructured curved meshes. Compared to other finite element formulations in the literature, our discretization attempts to directly mimic the non-conservative form used by most finite difference solvers, which allows for a highly efficient Picard solver. This is achieved by splitting the equations into a system which defines the weak derivatives of the metric tensor in the same discretization space as the mesh deformation, and re-writing the Winslow equations as a conservative second-order term plus a first-order term. The resulting formulation is simple to implement, allows for highly efficient solvers, and typically produces high quality curved meshes. The approach appears to be particularly effective for anisotropic boundary layers, and we show that the number of Picard iterations is essentially constant regardless of the refinement level, compared to a standard adaptive Newton solver as well as a non-linear elasticity solver which both need additional iterations for higher levels of refinement.

We illustrate the performance of our method through examples in two and three dimensions, and analyze the quality of the generated meshes. Examples include meshes with anisotropic boundary layers, thin regions, and coarse meshes for complex geometries.

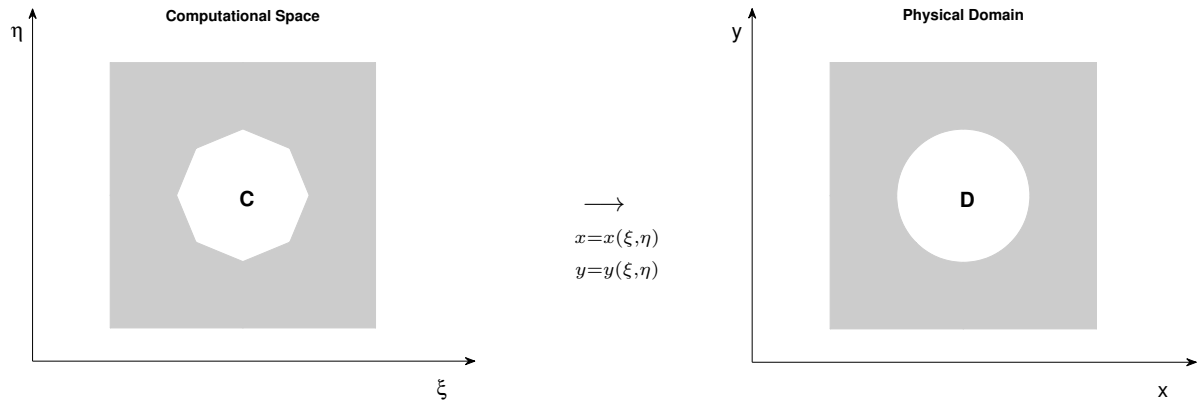


Figure 1: Winslow Equations in 2D: obtained by enforcing the computational coordinates to be harmonic, that is, the equations $\Delta\xi(x,y) = 0$ and $\Delta\eta(x,y) = 0$ are rewritten and solved in the computational space.

2. Problem Formulation

Our high-order curved mesh generation approach starts from an initial straight sided mesh of the actual geometry, which can be generated using a number of well-established approaches. On this *computational domain*, or reference domain, we define a space of piecewise polynomials, which we use to represent the coordinates of the curved *physical domain*. In this work we simply constrain all the boundary nodes in the physical domain to be located on the true curved geometry, but more sophisticated node placement strategies can be employed [30, 20]. The remaining interior nodes are determined from the solution of a system of second order elliptic equations known as the Winslow equations in the computational domain, and the corresponding deformed coordinates in the physical domain is the final curved mesh.

The Winslow equations are obtained by enforcing the computational coordinates to be harmonic, as illustrated in Fig. 1. More specifically, let $D \subset \mathbb{R}^n$ be the simply connected bounded physical domain in n space dimensions, $C \subset \mathbb{R}^n$ the computational domain, and define the mapping $\boldsymbol{x} : C \to D$, where $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi}) = (x_1(\boldsymbol{\xi}), \ldots, x_n(\boldsymbol{\xi}))$. Conversely, the mapping from domain D to C will be denoted by $\boldsymbol{\xi} = \boldsymbol{\xi}(\boldsymbol{x}) = (\xi_1(\boldsymbol{x}), \ldots, \xi_n(\boldsymbol{x}))$.

In the derivations below, we use Einstein's summation notation. We define the covariant base vectors as

$$\boldsymbol{g}_i = \partial_i \boldsymbol{x}, \quad \text{for } i = 1, \ldots, n.$$

Naturally, the covariant metric tensor components g_{ij} are defined as the inner product of the covariant base vectors, i.e,

$$g_{ij} = (\boldsymbol{g}_i, \boldsymbol{g}_j), \quad \text{for } i, j = 1, \ldots, n.$$

Furthermore, the contravariant base vectors $\boldsymbol{g}^i = \nabla_{\boldsymbol{x}} \xi_i$ satisfy

$$(\boldsymbol{g}^i, \boldsymbol{g}_j) = \delta^i_j, \quad \text{for } i, j = 1, \ldots, n,$$

where δ^i_j is the Kronecker delta.

Next, define the contravariant metric tensor components

$$g^{ij} = (\boldsymbol{g}^i, \boldsymbol{g}^j), \quad \text{for } i, j = 1, \ldots, n.$$

so that $g^{ij} g_{jk} = \delta^i_k$, and let g be the determinant of the covariant metric tensor $g = \det(g_{ij})$.

Consider the arbitrary function $\phi = \phi(\boldsymbol{\xi})$ defined in the computational domain C, then ϕ is also defined in D through the mapping $\boldsymbol{\xi}(\boldsymbol{x})$. It is a well-know result from differential geometry that its Laplace operator can be written as (see, for example, [11])

$$\Delta_{\boldsymbol{x}} \phi = \frac{1}{\sqrt{g}} \partial_i \left(\sqrt{g} g^{ij} \partial_j \phi \right). \tag{1}$$

Setting $\phi = \xi_j$ in this expression, we get

$$\Delta_{\boldsymbol{x}} \xi_j = \frac{1}{\sqrt{g}} \partial_i \left(\sqrt{g} g^{ij} \right) \quad \text{for } j = 1, \dots, n. \tag{2}$$

Finally, we can rewrite equation (1) as

$$\Delta_{\boldsymbol{x}} \phi = \frac{1}{\sqrt{g}} \partial_i \left(\sqrt{g} g^{ij} \right) \partial_j \phi + g^{ij} \partial_i \partial_j \phi =_{(2)} \Delta_{\boldsymbol{x}} \xi_j \partial_j \phi + g^{ij} \partial_i \partial_j \phi. \tag{3}$$

Imposing our computational coordinates to be harmonic, we get

$$\Delta_{\boldsymbol{x}} \xi_j = 0, \quad \text{for } j = 1, \dots, n.$$

Plugging this equation into (3), and taking $\phi = x_k$ it follows that

$$0 = \Delta_{\boldsymbol{x}} x_k = g^{ij} \partial_i \partial_j x_k \quad \text{for } k = 1, \dots, n,$$

which leads to the following simple form of the Winslow equations in physical coordinates:

$$g^{ij} \partial_i \partial_j x_k = 0 \quad \text{for } k = 1, \dots, n, \tag{4}$$

where, again, g^{ij} are defined through the relation $g^{ij} g_{jk} = \delta_{ik}$ and $g_{ij} = (\partial_i \boldsymbol{x}, \partial_j \boldsymbol{x})$.

Note that equations (4) form a nonlinear system, since the contravariant metric tensor depends on the unknown solution $\boldsymbol{x}$. One of the main advantages of this particular formulation is that it allows for a highly efficient solution strategy using Picard iterations, where the components of g^{ij} are computed from an old solution and a linear problem is solved for a new improved solution.

However, because of the non-conservative form, it is not obvious how to discretize equations (4) using a finite element approach. In our formulation, we address this by rewriting the equations as a system involving both $\boldsymbol{x}$ and the new variables $\boldsymbol{\alpha}$ defined as the negative derivatives of the contravariant metric tensor. Assuming sufficient smoothness of the solution fields, we can then rewrite the Winslow equations as a conservative second-order term plus a first order term involving $\boldsymbol{\alpha}$, to obtain the final form of our governing equations:

$$\partial_i (g^{ij}) + \alpha_j = 0, \qquad \text{for } j = 1, \dots, n, \tag{5}$$

$$\partial_i (g^{ij} \partial_j x_k) + \alpha_j \partial_j x_k = 0, \qquad \text{for } k = 1, \dots, n. \tag{6}$$

As mentioned before, we impose standard Dirichlet conditions on the solution field $\boldsymbol{x}$ on the entire boundary of the computational domain:

$$\boldsymbol{x}(\boldsymbol{\xi}) = \boldsymbol{x}_{\text{bnd}}(\boldsymbol{\xi}), \quad \boldsymbol{\xi} \in \partial C, \tag{7}$$

where $\boldsymbol{x}_{\text{bnd}}(\boldsymbol{\xi})$ represents the true curved boundary.

3. Discretization and Solution Method

3.1. Finite Element Formulation

Our finite element discretization is based on a standard continuous Galerkin formulation of the split form (5)-(6) of the Winslow equations. Continuous and piecewise polynomial approximation spaces of a given degree p are used for both $\boldsymbol{\alpha}$ and $\boldsymbol{x}$. After integrating by parts, we reduce the regularity requirements and obtain a formulation of this second-order system which is well-defined even for linear elements.

First, we define the elements of the straight sided mesh for the computational domain C,

$$\mathcal{T}_h = \{K_1, K_2, \dots\},$$

where $C = \cup_{K \in \mathcal{T}_h} K$. On this triangulation, we define the space of n continuous piecewise polynomials of degree p by

$$V_h^p = \{\boldsymbol{v} \in [\mathcal{C}_0(C)]^n \mid \boldsymbol{v}|_K \in [\mathcal{P}_p(K)]^n \ \forall K \in \mathcal{T}_h\},$$

where $\mathcal{P}_p(K)$ is the space of polynomials of degree at most $p \geq 1$ on K. We also introduce the subspace of functions in V_h^p satisfying the non-homogeneous Dirichlet conditions

$$V_{h,D}^p = \{\boldsymbol{v} \in V_h^p, \boldsymbol{v}|_{\partial C} = \boldsymbol{x}_{\text{bnd}}^p\},$$

as well as the homogeneous Dirichlet boundary conditions

$$V_{h,0}^p = \{\boldsymbol{v} \in V_h^p, \boldsymbol{v}|_{\partial C} = 0\}.$$

Here $\boldsymbol{x}_{\text{bnd}}^p$ is a suitable projection of $\boldsymbol{x}_{\text{bnd}}$ onto the space of piecewise polynomials of order p defined over ∂C. In this work, we use a standard nodal interpolant.

Our finite element formulation of (5)-(6) seeks approximate solution fields $\boldsymbol{\alpha}^h \in V_h^p$ and $\boldsymbol{x}^h \in V_{h,D}^p$. First consider equation (5). Multiply by an arbitrary test function $\boldsymbol{z} = (z_1, \ldots, z_n) \in V_h^p$, integrate over the domain C, and integrate by parts, to obtain the corresponding finite element formulation: Find $\boldsymbol{\alpha}^h = (\alpha_1^h, \ldots, \alpha_n^h) \in V_h^p$ such that

$$\int_C \alpha_j^h z_j \, dV = \int_C g^{ij} \partial_i z_j \, dV - \int_{\partial C} g^{ij} \hat{n}_i z_j \, dS, \tag{8}$$

for all $\boldsymbol{z} \in V_h^p$, where $\hat{\boldsymbol{n}} = (\hat{n}_1, \ldots, \hat{n}_n)$ is the outward normal at the domain boundary ∂C and the components g^{ij} are defined by element-wise differentiation of the approximate solution $\boldsymbol{x}^h$. The system (8) is discretized using a standard nodal finite element approach. We consider a nodal basis $\{\boldsymbol{\varphi}_k\}_{k=1}^m$ of the m-dimensional space V_h^p, and write the approximate solution fields as

$$\boldsymbol{\alpha}^h = \sum_{k=1}^m \alpha_k^h \boldsymbol{\varphi}_k, \qquad \boldsymbol{x}^h = \sum_{k=1}^m x_k^h \boldsymbol{\varphi}_k, \tag{9}$$

where α_k^h and x_k^h are the expansion coefficients in the nodal basis. We impose equations (8) for all basis functions $\boldsymbol{\varphi}_k$, $k = 1, \ldots, m$. We compute the integrals using high-order Gauss integration rules, and after a standard matrix assembly approach we obtain a system of the form

$$M\alpha_j^h = b_j, \qquad j = 1, \ldots, n, \tag{10}$$

where M is a symmetric mass matrix, which is constant over all components of $\boldsymbol{\alpha}^h$. Note that our definition of $\boldsymbol{\alpha}^h$ can be seen as a weak projection of the derivatives of g^{ij} onto the approximation space V_h^p.

The finite element formulation of equation (6) with boundary conditions (7) follows the same lines: We seek a solution $\boldsymbol{x}^h = (x_1^h, \ldots, x_n^h) \in V_{h,D}^p$, such that for all functions $\boldsymbol{z} = (z_1, \ldots, z_n) \in V_{h,0}^p$ we have

$$\int_C g^{ij} \partial_j x_k^h \partial_i z_k \, dV - \int_C \alpha_j \partial_j x_k^h z_k \, dV = 0. \tag{11}$$

Similarly to before, we impose equation (11) for $\boldsymbol{z} = \boldsymbol{\varphi}_k$, for all $k = 1, \ldots, m$. For the assembly of these equations, we treat the coefficients $\boldsymbol{\alpha}^h$ and g^{ij} as constant fields which leads to a linear system in $\boldsymbol{x}^h$:

$$Kx_j^h = c_j, \qquad j = 1, \ldots, n, \tag{12}$$

which also incorporates the strongly enforced Dirichlet boundary conditions (7).

3.2. Solution Procedure

Our discretization lead to the final discrete system of equations (10) and (12), which we can write in the following form to emphasize the nonlinear dependencies:

$$M\alpha_j^h = b_j(\boldsymbol{x}^h), \qquad j = 1, \ldots, n, \tag{13}$$

$$K(\boldsymbol{\alpha}^h, \boldsymbol{x}^h)x_j^h = c_j(\boldsymbol{\alpha}^h, \boldsymbol{x}^h), \qquad j = 1, \ldots, n. \tag{14}$$

We solve these nonlinear equations using Picard iterations as follows. Set the initial guess to the straight sided mesh at the interior nodes and to the curved boundary at the boundary nodes:

$$\boldsymbol{x}^{(0)}(\boldsymbol{\xi}) = \begin{cases} \boldsymbol{x}^p_{\text{bnd}}(\boldsymbol{\xi}), & \text{for } \boldsymbol{\xi} \text{ on } \partial C, \\ \boldsymbol{\xi}, & \text{for } \boldsymbol{\xi} \text{ in } C. \end{cases} \tag{15}$$

For a given solution iterate $\boldsymbol{x}^{(\ell)}$, we compute an improved iterate by the following steps:

1. Assemble (13) using $\boldsymbol{x}^h = \boldsymbol{x}^{(\ell)}$ and solve for $\boldsymbol{\alpha}^h = \boldsymbol{\alpha}^{(\ell)}$.
2. Assemble (14) using $\boldsymbol{x}^h = \boldsymbol{x}^{(\ell)}$ and $\boldsymbol{\alpha}^h = \boldsymbol{\alpha}^{(\ell)}$, and solve for $\boldsymbol{x}^{(\ell+1)}$.

These iterations are repeated until the norm of the difference between two iterates is smaller than a given tolerance.

We note the following important properties of this solution procedure:

- As defined, the initial guess is highly likely to contain invalid elements. However, the Picard iterations appear to be quite insensitive to this and typically repairs (or *untangles*) the curved mesh so that the final mesh is valid.
- Unlike for example non-linear elasticity based mesh curving approaches, the iterations usually converge in at most 10-20 iterations, even for highly stretched meshes that produce severe inversions in the initial guess.
- While the right hand side of equation (13) must be re-assembled in each iteration, the system matrix M is a standard continuous Galerkin mass matrix which is constant for all iterations and solution components. It does require a global linear solution to find $\boldsymbol{\alpha}^h$, however, the matrix is well-conditioned and can be solved by any standard linear solver for SPD systems.
- Both the matrix and the right hand side of equation (14) must be re-assembled in each iteration. However, the system matrix K is the same for all n solution components, and the systems can again be solved efficiently using standard solvers for general linear equations.
- All parts of the assembly and linear solution procedures can be parallelized using standard approaches.

In our implementation, we use a direct sparse solver for all 2D and for the small 3D problems. For the large 3D problems, we use a conjugate gradient / GMRES solver for the two systems, preconditioned by incomplete factorizations. All our results are computed in serial.

For comparison, in two dimensions, we also solve the non-linear system of equations (13) and (14) by solving a corresponding residual equation $\boldsymbol{r}(\boldsymbol{\alpha}^h, \boldsymbol{x}^h) = 0$ using Newton's method. Similarly to the Picard case, we set the initial guess to the straight sided mesh at the interior nodes and to the curved boundary at the boundary nodes, given by equation (15). As expected, Newton's method shows poor global convergence and typically fails to converge for initial conditions with tangled elements. This is the case for most of our challenging examples, in particular the boundary layer example presented in Section 4.2.

An alternative for the cases where the standard Newton's method does not converge is to use the adaptive Newton solver proposed in [18]. Instead of solving the residual equation $\boldsymbol{r}(\boldsymbol{\alpha}^h, \boldsymbol{x}^h) = 0$ directly, we use homotopy in a new scalar variable β. We define a series of problems $\boldsymbol{r}_\beta(\boldsymbol{\alpha}^h_\beta, \boldsymbol{x}^h_\beta) = 0$, parametrized by β, with Dirichlet boundary conditions

$$\boldsymbol{x}^h_\beta(\boldsymbol{\xi}) = (1-\beta)\boldsymbol{x}^p_{\text{ref}} + \beta\boldsymbol{x}^p_{\text{bnd}}, \qquad \text{for } \boldsymbol{\xi} \in \partial C,$$

where $\boldsymbol{x}^p_{\text{bnd}}$ are the original boundary conditions associated with the problem $\boldsymbol{r}(\boldsymbol{\alpha}^h, \boldsymbol{x}^h) = 0$, and $\boldsymbol{x}^p_{\text{ref}}$ are the boundary nodes on the reference mesh. By slowly increasing β from 0 to 1, we approach the actual problem with boundary conditions $\boldsymbol{x}^p_{\text{bnd}}$. This results in a series of well-behaved problems that can be solved using regular Newton iterations.

The selection of the sequence of β-values can be automated in an adaptive way by monitoring the Newton convergence. If the method does not converge or inverted elements are found during the assembly process, then $\Delta\beta$ is reduced. If the Newton method converges fast enough, $\Delta\beta$ is increased. For more details on the implementation of this procedure, see [18]

4. Results

Here we give a number of examples of our approach and study the quality of the generated curved meshes as well as the behavior of the solution procedure. In all our examples below, we use polynomial approximations of degree $p = 4$ unless otherwise specified. To measure how much an element has been deformed, we use the determinant of the Jacobian matrix of the mapping $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi})$ between the computational and the physical domain. More specifically, let $J(\boldsymbol{\xi}) = \det(\partial\boldsymbol{x}/\partial\boldsymbol{\xi})$, and define the quality of an element to be the *scaled Jacobian* [3]

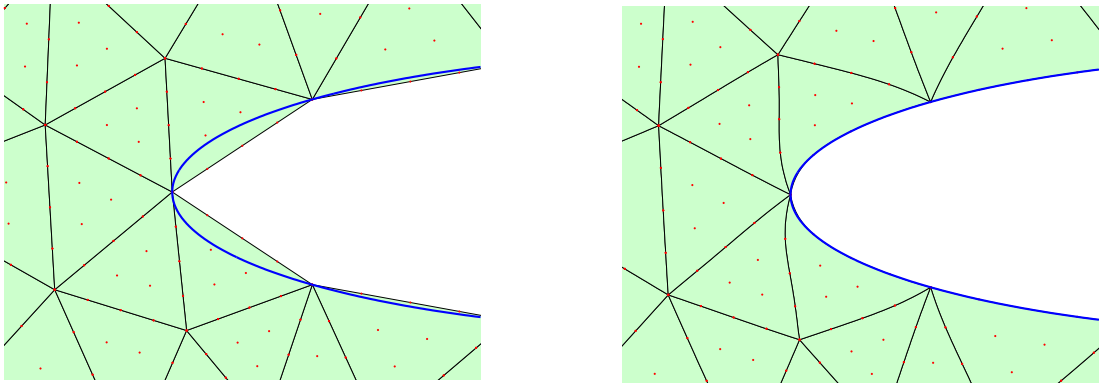
$$I = \frac{\min_{\boldsymbol{\xi}\in\kappa} J(\boldsymbol{\xi})}{\max_{\boldsymbol{\xi}\in\kappa} J(\boldsymbol{\xi})}.$$

In practice, we approximate this expression by evaluating the Jacobian at the points of a high-order Gauss integration rule.

We note that $I \leq 1$, and that $I = 1$ for all straight-sided simplex elements. Therefore, this measure will not capture well the *element quality* of the corresponding mesh element [4]. On the other hand, if I has a negative or very small value, it indicates that the element is inverted or close to degenerate. The presence of such elements in the mesh decreases the quality of the domain discretization and makes the corresponding systems of equations ill-conditioned.

4.1. Simple isotropic mesh

We start our examples with a two dimensional isotropic mesh. Fig. 2 shows a zoom-in around the curved boundary of the initial configuration, and the final curved mesh.

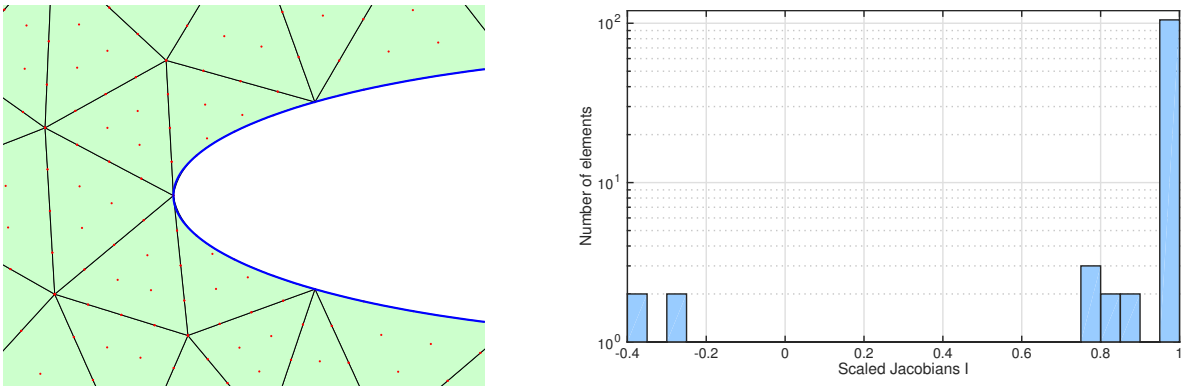


Straight sided mesh and initial configuration

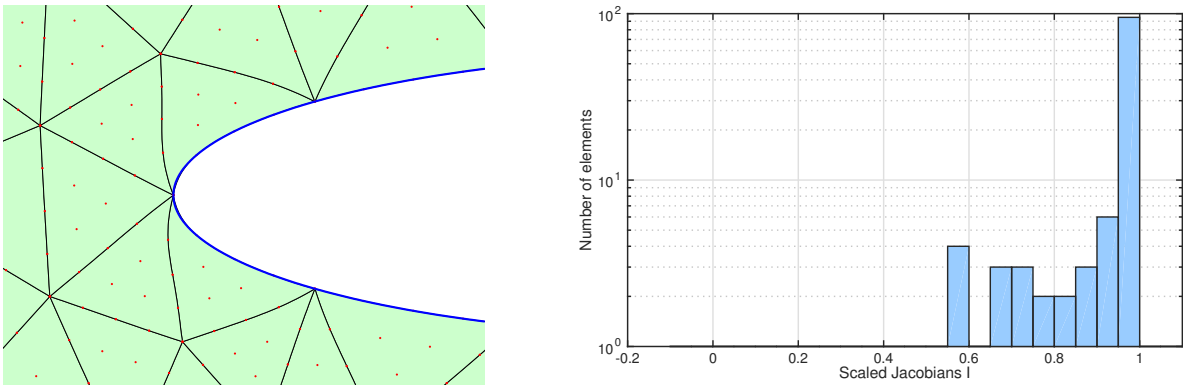
Final curved mesh

Figure 2: Curved mesh generation using the Winslow Equations

Even in this simple case, the local technique of conforming the boundary of elements which are in contact with the curved boundary does not produce a valid mesh (see Fig. 3.a). The elliptic smoothing will generate a final valid mesh, and the quality of the elements on the curved boundary can be seen in Fig. 3.b.



a) Conforming boundary nodes only



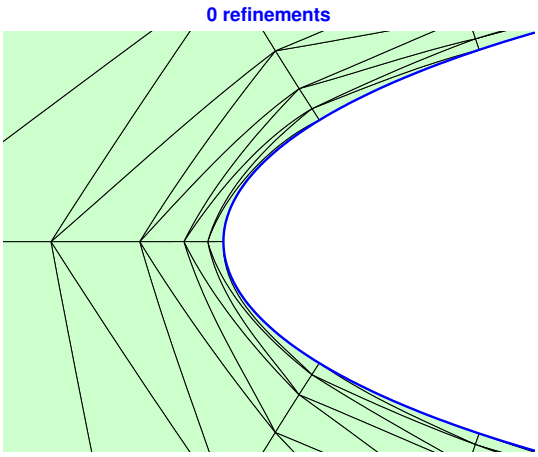
b) Converged Winslow smoothing

Figure 3: A simple example of isotropic two dimensional curved mesh generation. Initially, inverted elements are generated when conforming the boundary nodes to match the true geometry; these elements are later untangled and the solver produces a final valid mesh.

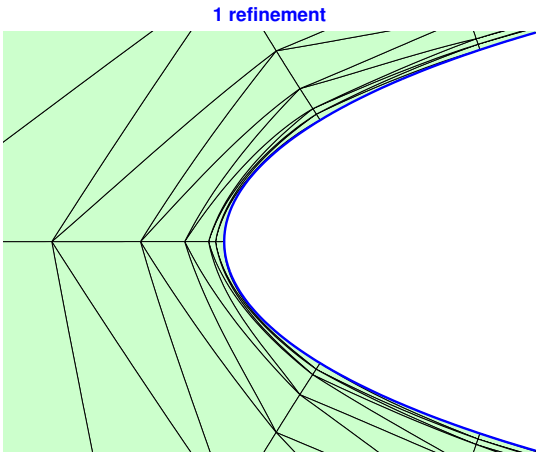
We also note that the elements further away from the boundaries are very close to straight sided. Therefore, when plotting the scaled Jacobians for all the examples in this paper we only consider elements that are adjacent to the curved boundary and their 4 closest neighbors.

4.2. Anisotropic triangular mesh and refinement study

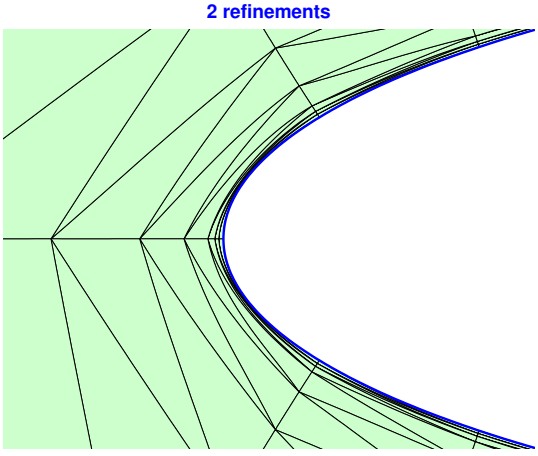
Here we test our method for an anisotropic boundary layer mesh. We also study the behavior of our solver as well as two other solvers as we locally refine the mesh in the normal direction, producing a sequence of increasingly stretched meshes. The original mesh is a mapped Cartesian grid, with each quadrilateral element split into two triangles with alternating directions (see Fig. 4). At each refinement level, we split the elements adjacent to the boundary in the normal direction, which results in a graded mesh with a growth factor of two. We perform up to four levels of these refinements, giving element aspect ratios up to about 100:1.



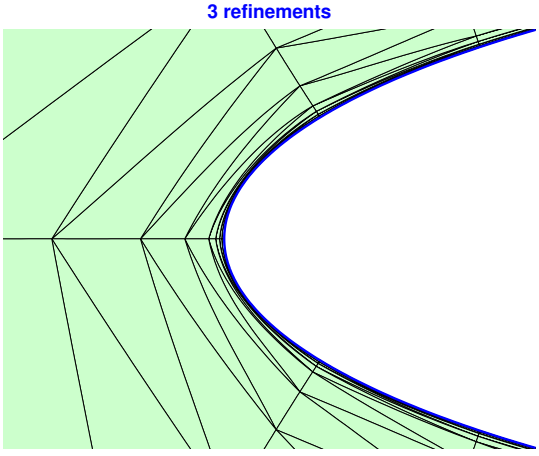
Final curved mesh, convergence in 10 iterations



Final curved mesh, convergence in 11 iterations



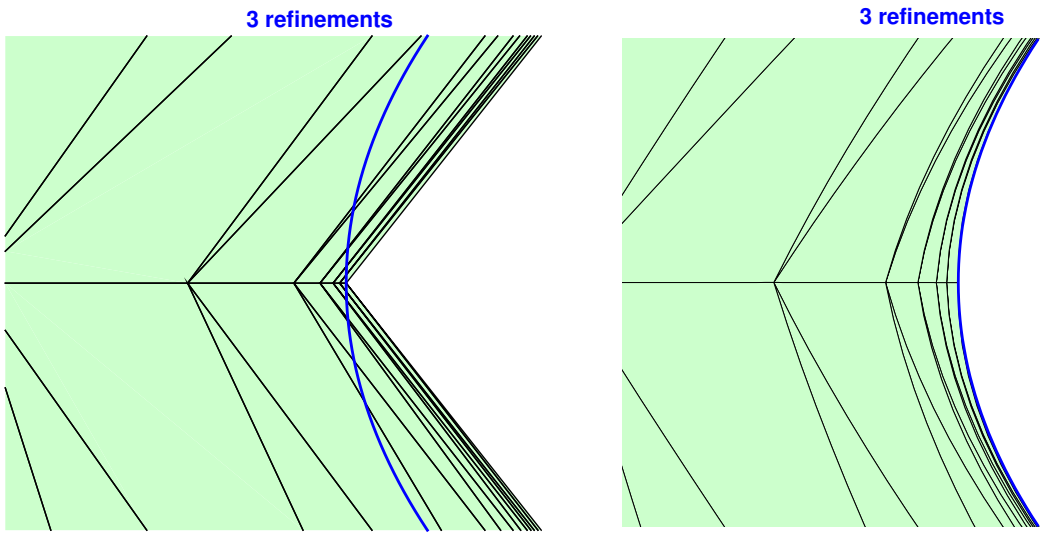
Final curved mesh, convergence in 11 iterations



Final curved mesh, convergence in 12 iterations

Figure 4: Refinement study: the reference mesh is locally refined close to the boundary layer, and our Winslow smoothing scheme is used to generate the corresponding curved mesh. We observe that the number of Picard iterations required for convergence remains mainly constant. The figures show the final curved meshes for the four refinement levels.

We see that even for the cases where the boundary layer is highly stretched, our solution method is capable of producing well-shaped curved elements. The most refined mesh is shown in Fig. 5 as a closer zoom-in around the curved boundary, as well as the corresponding histogram of the scaled Jacobians.



(a) Initial configuration of finest refinement level

(b) Final curved mesh of finest refinement level

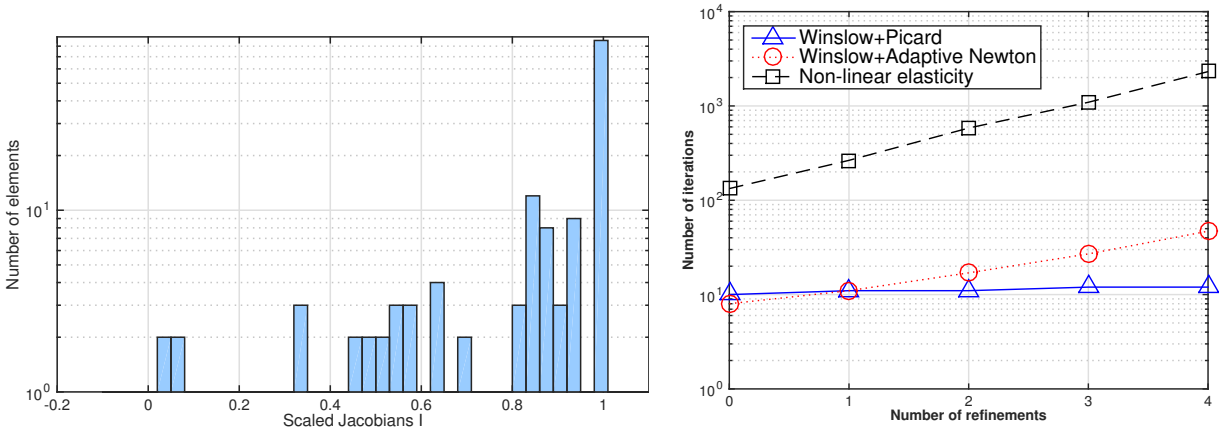
Figure 5: Refinement study: In (a) and (b), we see a zoom-in on the curved boundary layer mesh for the finest refinement level in our study, in the initial and final configurations.

We count the number of iterations the method takes to converge, and observe that it remains mainly constant as we refine the mesh. The fact that the number of iterations does not depend on the refinement level is quite remarkable, since other approaches such as the elasticity analogies typically require a number of small steps that scales by the inverse of the thickness of the boundary layer. If we use the Winslow formulation with the adaptive Newton solver discussed in Section 3.2 instead of the Picard linearization, we also do not observe the independence of the number of iterations on the refinement level. This is illustrated in Fig. 6, where we directly compare the number of iterations versus refinement level for our Winslow formulation using Picard iterations as well as Newton's method, and for the non-linear elasticity approach [18].

The graph clearly shows how the Newton solver and the non-linear elasticity solver need an increasing number of iterations as the mesh is refined. Also note that these two solvers require a higher cost per iteration than the Picard iterations, since they involve larger coupled linear systems of equations. As an example, for the finest mesh in this 2-D case each non-linear elasticity solution was about 3.5 times slower than a Winslow-Picard solution, and each Winslow-Newton solution was about 30 times slower (due to the full coupling between all components). Therefore, in our experience the Winslow-Picard solver is often magnitudes faster than the alternatives, although this is highly dependent on the implementation and the solver choices.

4.3. Airfoil quad-mesh

To demonstrate that our method is also suitable for higher orders of approximation, we consider a quadrilateral mesh with a stretched boundary layer and polynomial degrees $p = 7$. The reason we use quadrilateral elements for this example, is that they are implemented more efficiently in our code for high polynomial degrees (using outer product Gauss-Lobatto nodes). The final curved mesh is valid and its scaled Jacobians are plotted in Fig. 7. Note that unlike for simplex elements, the scaled Jacobian measure for quadrilateral elements can be less than one even for all straight-sided elements. However, it is still a useful measure and in particular it shows that the element is non-inversed if positive.



(a) Scaled Jacobian for final curved mesh

(b) Iterations comparison

Figure 6: (a) The scaled Jacobians for the boundary elements and their four closest neighbors. (b) A comparison of the number of iterations for non-linear elasticity [18] and our Winslow formulation using both the Picard and the Newton solver. Note that the linear systems solved in each iteration are much larger for the Newton solver and the non-linear elasticity solver, but for simplicity we only compare the number of iterations.

4.4. Simple tetrahedral mesh example

Next, in Fig. 8 we show the results of the generation of a tetrahedral curved mesh for a door hinge structure. The initial straight-sided mesh was obtained through the NETGEN mesh generator [22]. While this example does not have any stretched elements, it shows that the procedure handles general 3D unstructured Delaunay meshes well.

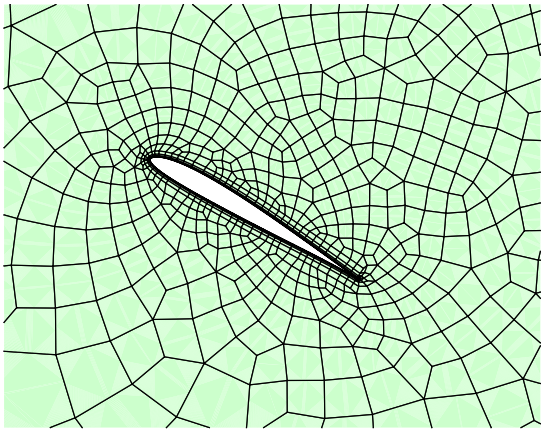
4.5. Tetrahedral mesh of cylindrical component

Here we use a geometry of a cylindrical component with thin walls, and again generate an unstructured tetrahedral mesh using the NETGEN mesh generator, see Fig. 9 and 10. Note that the inner and the outer surfaces of the cylinder are meshed independently from the CAD geometry, the nodes are not necessarily aligned in a straight-forward way. Also, the volume mesh consists of only a single layer of unstructured tetrahedra. The scaled Jacobians show that it is indeed difficult to curve these elements, but the Winslow procedure does again succeed in producing a valid mesh after only four Picard iterations.

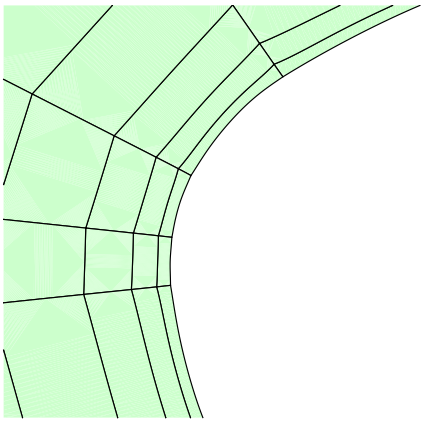
4.6. Tetrahedral mesh of a NACA 0012 wing

Here we consider an unstructured mesh of an extruded NACA 0012 airfoil with rounded edges, see Fig. 11. The mesh has a slightly stretched boundary layer, typical for LES-type simulations, with an aspect ratio of about 10:1. Note the highly coarse elements at the rounded wing-tip, which proved to be challenging for the Picard solver and in order to obtain a converged solution we had to prescribe the corner node locations of all the tetrahedral elements to their original positions in the straight sided mesh. This technique appears to improve the convergence properties of the Picard iterations in general, but for simplicity we only apply it to this example.

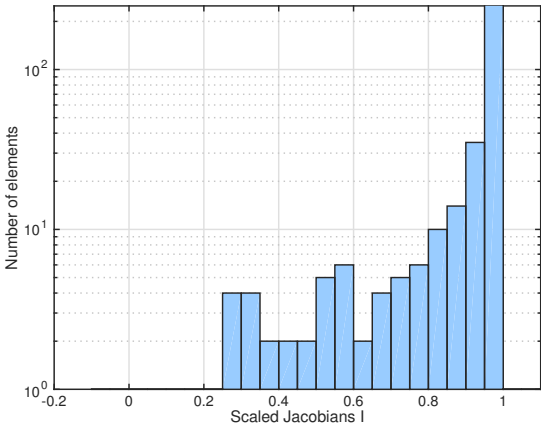
We note that the highly distorted elements are located around edges of high curvature, as expected. Fig. 12 shows that only a few elements have a scaled Jacobian smaller than 0.3. With the prescribed element corner positions, our solver converged in only nine iterations.



Final curved mesh

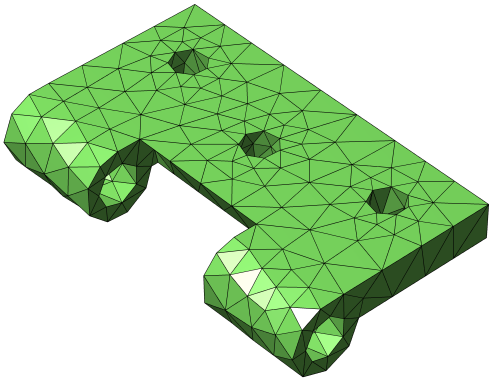


Final curved mesh – boundary close up

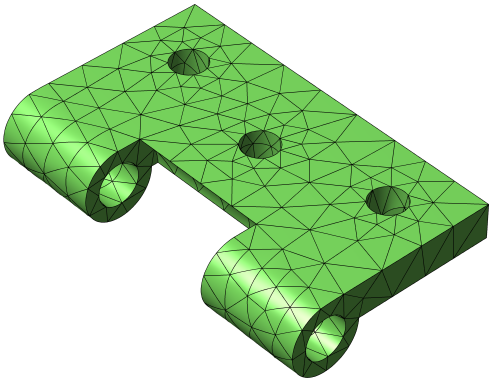


Scaled Jacobians

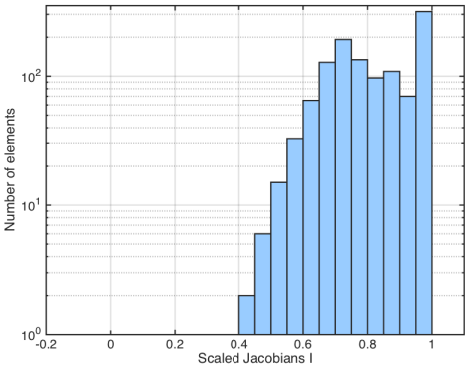
Figure 7: Curved mesh generation of an airfoil with quadrilateral mesh. Here we use polynomial approximations of degree $p = 7$.

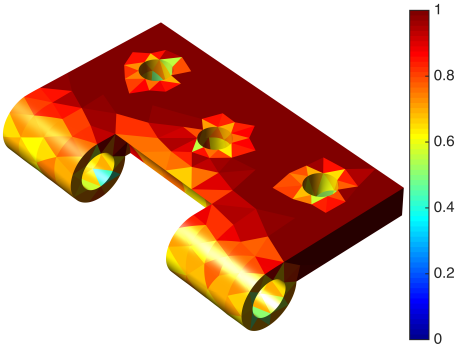


Initial straight mesh



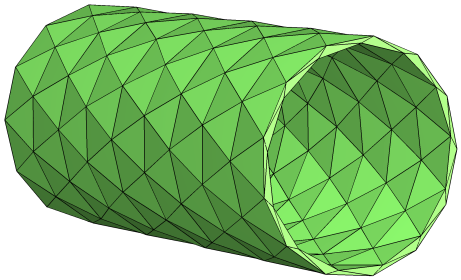
Final curved mesh



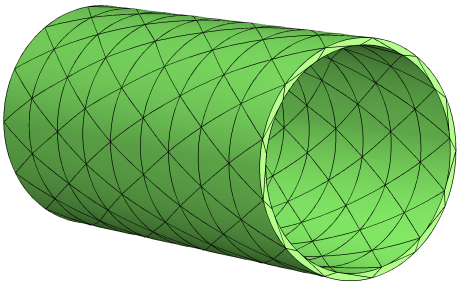


Scaled Jacobians

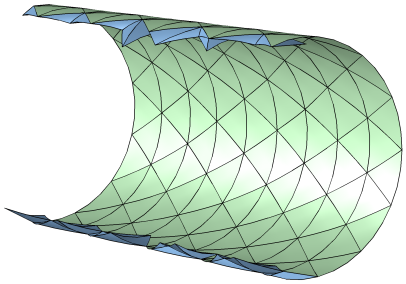
Figure 8: Curved mesh generation of a door hinge geometry. The top figures show the initial straight sided mesh and the final curved mesh. The bottom figures show the scaled Jacobians, both as histogram and a surface plot.



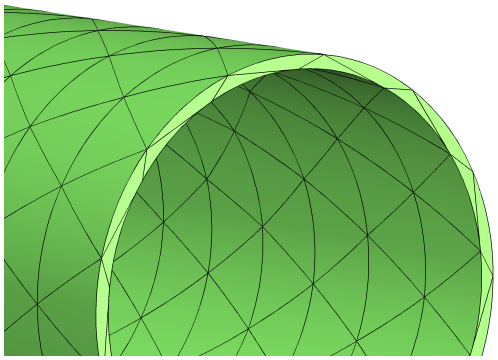
Initial straight mesh



Final curved mesh

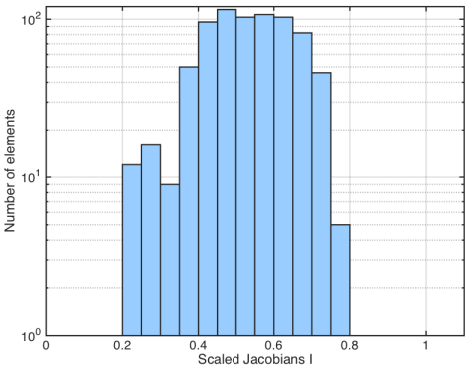


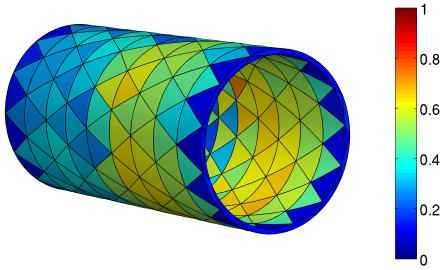
Curved mesh, split view



Curved mesh, zoom-in at the thin structure

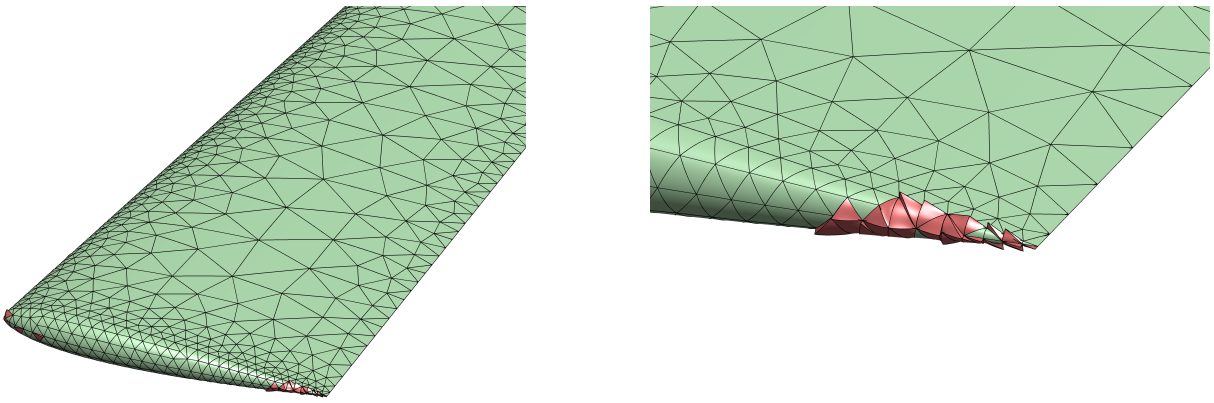
Figure 9: Curved mesh of a cylindrical component with thin structures. Note that the volume mesh consists of only a single layer of unstructured tetrahedra, and the inner and outer surfaces are meshed independently.





Scaled Jacobians

Figure 10: Scaled Jacobians for the final curved mesh of the cylindrical component. Note that many of the elements are highly distorted but still valid, due to the coarse mesh and the thin structure.



Curved mesh, elements with $I < 0.5$ | Curved mesh, elements with $I < 0.5$

Figure 11: NACA 0012 airfoil final curved mesh: highly distorted elements are located around edges of high curvature.

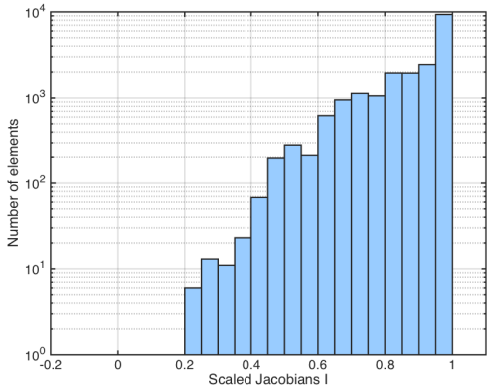
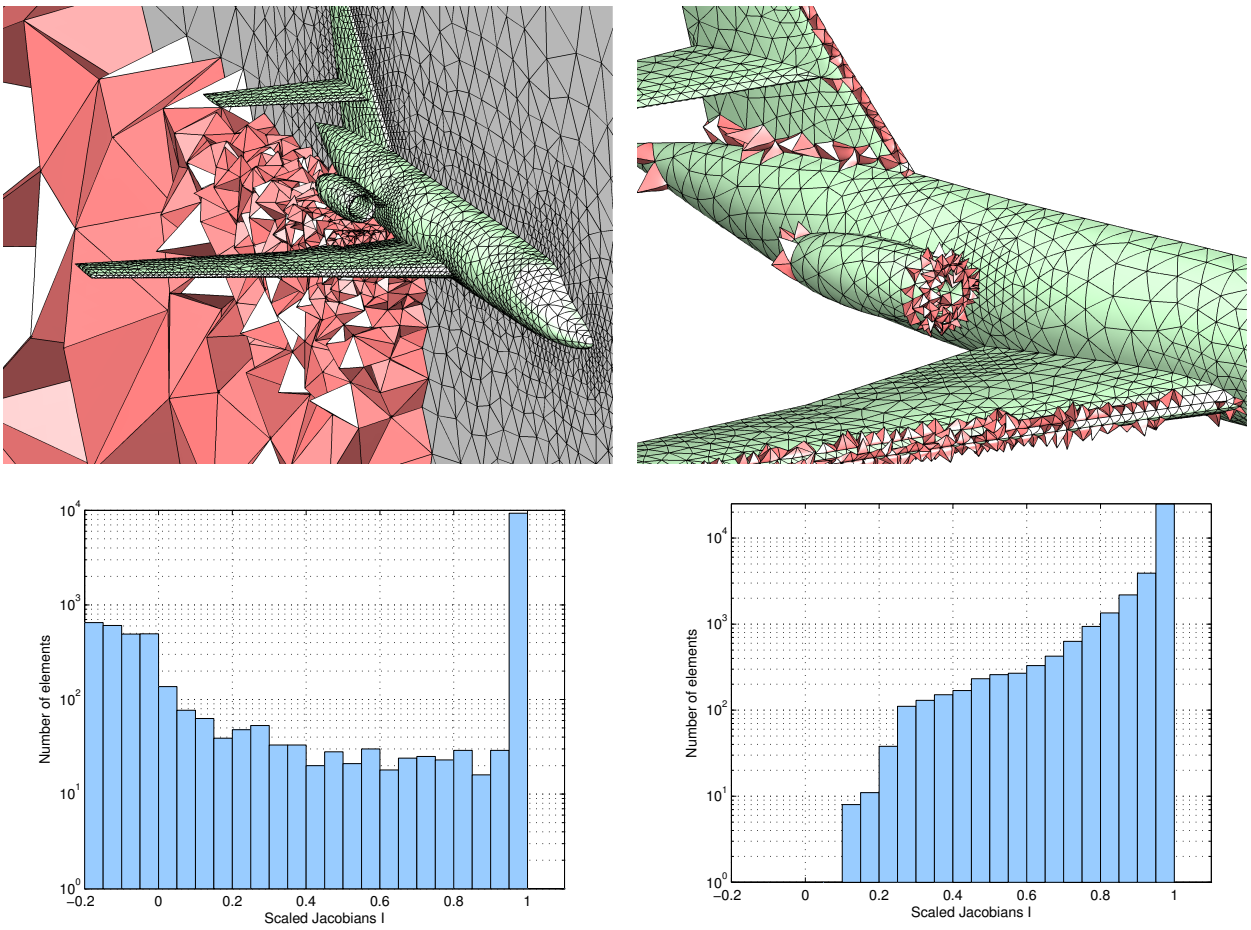


Figure 12: Scaled Jacobians of NACA 0012 wing final curved mesh. We report numbers for all elements on curved boundary and their four closest neighbors. Note that there are only a few elements with scaled Jacobians smaller than 0.3.

4.7. Tetrahedral mesh of Falcon aircraft

In our final example we consider a complex geometry of a complete Falcon aircraft configuration, see Fig. 13. The mesh is approximately isotropic, however, the tetrahedra are very coarse close to regions with high curvature which makes this a challenging case for mesh curving. Once again, we note that although the initial curved mesh has many inverted elements, the final converged mesh converged after 29 iterations and has a smallest scaled Jacobian of about 0.2.



Scaled Jacobians, local curving only

Scaled Jacobians, final configuration

Figure 13: Falcon aircraft configuration: The top right plot shows the elements with scaled Jacobian smaller than 0.5. Note that in the initial condition, where the elements are only curved locally, we have many inverted elements ($I < 0$). All of these are untangled by the Winslow smoothing scheme and the final result is a mesh with well-shaped elements.

5. Conclusions

We have presented a new approach for generating high-order curved meshes using the classical Winslow equations. We use a continuous Galerkin finite element formulation based on a split form of the equations, which in particular allows for efficient solution using Picard iterations. The linear systems that arise are small (only one solution component) and easy to solve by direct or iterative methods.

We demonstrated the procedure in both two and three dimensions, using triangular, quadrilateral, and tetrahedral meshes. In all examples, we were able to produce valid meshes with well-shaped elements, even when considering challenging geometries with thin structures and anisotropic boundary layer elements.

The Winslow smoothing along with our finite element formulation has proven to be particularly interesting in the presence of boundary layers. Unlike many other methods, the number of iterations the method requires to converge appears to remain constant as the mesh is refined with increasing aspect ratios. This could translate into performance advantages of the order of magnitudes for many problems.

Acknowledgements

We would like to acknowledge the generous support from the AFOSR Computational Mathematics program under grant FA9550-10-1-0229, the Alfred P. Sloan foundation, and the Director, Office of Science, Computational and Technology Research, U.S. Department of Energy under Contract No. DE-AC02-05CH11231. The first author was also supported by the Fulbright and CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, Brazil).

References

[1] Charakhch'yan, A.A., Ivanenko, S.A.: A variational form of the Winslow grid generator. J. Comput. Phys. **136**(2), 385–398 (1997). DOI 10.1006/jcph.1997.5750. URL http://dx.doi.org/10.1006/jcph.1997.5750

[2] Codd, A.L., Manteuffel, T.A., McCormick, S.F., Ruge, J.W.: Multilevel first-order system least squares for elliptic grid generation. SIAM J. Numer. Anal. **41**(6), 2210–2232 (2003). DOI 10.1137/S0036142902404418. URL http://dx.doi.org/10.1137/S0036142902404418

[3] Dey, S., O'Bara, R., Shephard, M.: Curvilinear mesh generation in 3D. Comput. Aided Geom. Design **33**, 199–209 (2001)

[4] Field, D.A.: Qualitative measures for initial meshes. Internat. J. Numer. Methods Engrg. **47**, 887–906 (2000)

[5] Gargallo-Peiró, A., Roca, X., Peraire, J., Sarrate, J.: Defining quality measures for mesh optimization on parameterized cad surfaces. In: Proceedings of the 21st International Meshing Roundtable, pp. 85–102. Springer (2013)

[6] George, P.L., Borouchaki, H.: Construction of tetrahedral meshes of degree two. Internat. J. Numer. Methods Engrg. **90**(9), 1156–1182 (2012)

[7] Hansen, G., Zardecki, A., Greening, D., Bos, R.: A finite element method for unstructured grid smoothing. Journal of Computational Physics **194**(2), 611 – 631 (2004)

[8] Karman Jr, S.L.: Virtual control volumes for two-dimensional unstructured elliptic smoothing. In: Proceedings of the 19th International Meshing Roundtable, pp. 121–142. Springer (2010)

[9] Karman Jr, S.L.: Virtual control volumes for three-dimensional unstructured elliptic smoothing. In: 49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition (2011)

[10] Knupp, P.M.: Jacobian-weighted elliptic grid generation. SIAM J. Sci. Comput. **17**(6), 1475–1490 (1996). DOI 10.1137/S1064827594278563. URL http://dx.doi.org/10.1137/S1064827594278563

[11] Kreyszig, E.: Differential geometry. Dover Publications, Inc., New York (1991). Reprint of the 1963 edition

[12] Luo, X., et al.: Automatic p-version mesh generation for curved domains. Eng. Comput. **20**(3), 273–285 (2004)

[13] Moxey, D., Ekelschot, D., Keskin, Ü., Sherwin, S., Peiró, J.: A thermo-elastic analogy for high-order curvilinear meshing with control of mesh validity and quality. Procedia Engineering **82**, 127–135 (2014)

[14] Nielsen, E.J., Anderson, W.K.: Recent improvements in aerodynamic design optimization on unstructured meshes. AIAA Journal **40**(6), 1155–1163 (2002)

[15] Oliver, T.A.: A high-order, adaptive, discontinuous Galerkin finite element method for the Reynolds-averaged Navier-Stokes equations. ProQuest LLC, Ann Arbor, MI (2008). Thesis (Ph.D.)–Massachusetts Institute of Technology

[16] Owen, S.J.: A survey of unstructured mesh generation technology. In: Proceedings of the 7th International Meshing Roundtable, pp. 239–267. Sandia Nat. Lab. (1998)

[17] Peraire, J., Vahdati, M., Morgan, K., Zienkiewicz, O.C.: Adaptive remeshing for compressible flow computations. J. Comput. Phys. **72**(2), 449–466 (1987)

[18] Persson, P.O., Peraire, J.: Curved mesh generation and mesh refinement using lagrangian solid mechanics. In: Proceedings of the 47th AIAA Aerospace Sciences Meeting and Exhibit (2009)

[19] Roca, X., Gargallo-Peiró, A., Sarrate, J.: Defining quality measures for high-order planar triangles and curved mesh generation. In: W. Quadros (ed.) Proceedings of the 20th International Meshing Roundtable, pp. 365–383. Springer Berlin Heidelberg (2012)

[20] Ruiz-Gironés, E., Roca, X., Sarrate, J.: Optimizing mesh distortion by hierarchical iteration relocation of the nodes on the cad entities. Procedia Engineering **82**, 101–113 (2014)

[21] Ruppert, J.: A Delaunay refinement algorithm for quality 2-dimensional mesh generation. J. Algorithms **18**(3), 548–585 (1995)

[22] Schöberl, J.: NETGEN An advancing front 2D/3D-mesh generator based on abstract rules. Computing and visualization in science **1**(1), 41–52 (1997)

[23] Shephard, M.S., Flaherty, J.E., Jansen, K.E., Li, X., Luo, X., Chevaugeon, N., Remacle, J.F., Beall, M.W., O'Bara, R.M.: Adaptive mesh generation for curved domains. Appl. Numer. Math. **52**(2-3), 251–271 (2005)

[24] Sherwin, S.J., Peirò, J.: Mesh generation in curvilinear domains using high-order elements. Internat. J. Numer. Methods Engrg. **53**(1), 207–223 (2002)

[25] Shewchuk, J.R.: Delaunay refinement algorithms for triangular mesh generation. Comput. Geom. **22**(1-3), 21–74 (2002). 16th ACM Symposium on Computational Geometry (Hong Kong, 2000)
[26] Soni, B.K.: Elliptic grid generation system: control functions revisited. I. Appl. Math. Comput. **59**(2-3), 151–163 (1993). DOI 10.1016/0096-3003(93)90086-T. URL http://dx.doi.org/10.1016/0096-3003(93)90086-T
[27] Thompson, J.F.: Elliptic grid generation. Appl. Math. Comput. **10/11**, 79–105 (1982). DOI 10.1016/0096-3003(82)90188-6. URL http://dx.doi.org/10.1016/0096-3003(82)90188-6. Numerical grid generation (Nashville, Tenn., 1982)
[28] Toulorge, T., Geuzaine, C., Remacle, J.F., Lambrechts, J.: Robust untangling of curvilinear meshes. J. Comput. Phys. **254**, 8–26 (2013)
[29] Winslow, A.M.: Numerical solution of the quasilinear Poisson equation in a nonuniform triangle mesh. J. Computational Phys. **1**, 149–172 (1967)
[30] Xie, Z.Q., Sevilla, R., Hassan, O., Morgan, K.: The generation of arbitrary order curved meshes for 3D finite element analysis. Comput. Mech. **51**(3), 361–374 (2013)