An active set strategy to generate quadrilateral grids

N. Egidi*, P. Maponi

Dipartimento di Matematica e Informatica, Università di Camerino, 62032 Camerino (MC), Italy

Received 1 October 2006; received in revised form 28 June 2007

Abstract

We consider the problem of the generation of quadrilateral grids on planar domains. This problem is numerically solved by a two phases method: an iterative procedure based on the well-known variational approach, and an active set procedure to obtain unfolded quadrilaterals. This second phase is performed only when it is really necessary, in fact the first phase alone gives satisfactory results on a large number of domains. This two phases approach provides a robust method with low computational cost. Numerical experiments show that this method is able to generate unfolded grids also on complex domains.

© 2007 Elsevier B.V. All rights reserved.

MSC: 65N50; 65D18

Keywords: Quadrilateral grid generation; Heat equation; Discrete variational method; Optimization problem; Steepest descent method

1. Introduction

Numerical methods for grid generation are usually employed in several applications where the approximate solution of partial differential equations is required, see [5,7,11] for some examples. The accuracy of these numerical solutions strictly depends on the quality of the discretization grid, so the grid generation is a crucial step in such approximation processes, and, as consequence of this fact, several generation techniques have been proposed in the scientific literature, see [11] for a complete survey.

We consider the following grid generation problem: given a compact, simply connected planar domain \( \Omega \), compute a quadrilateral grid \( \mathcal{Q} \) on \( \Omega \); note that, in general, a grid can be seen as a special partition of a given domain, see [10, p. 2] for a precise definition. The grid generation problem can be formulated as a variational problem, where each grid feature is modeled by an appropriate functional. The minimizer obtained from a suitable combination of these functionals gives the coordinates of the vertices in the corresponding optimal grid, see [9] for details. We propose a method constituted by two different phases: (1) an iterative solution of the variational problem, (2) an active set procedure to obtain unfolded quadrilaterals. The first phase is based on the steepest descent technique and a simple procedure to adapt some weights in the objective function; note that these weights depend on the particular domain \( \Omega \) taken into account, so they cannot be chosen with a priori criterion. This phase alone gives satisfactory results on a large number of domains; however, it may produce low-quality grids or folded grids, depending on the difficulty of the domain taken into account. Note that this is a quite general drawback of the variational formulation [1,9].

* Corresponding author.

E-mail addresses: nadaniela.egidi@unicam.it (N. Egidi), pierluigi.maponi@unicam.it (P. Maponi).

0377-0427/S - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2007.07.030
In these cases, the second phase tries to change this grid improving the worst oriented quadrilateral and deteriorating as little as possible the quality of the remaining quadrilaterals. This second phase is the main result of the present paper and we implicitly require the convexity of the quadrilaterals as a constraint in the above-mentioned variational problem. We note that this phase resembles the well-known active set strategy in the constrained optimization theory, see [8, Chapter 5] for details.

Finally, we report some numerical results obtained with the proposed method. In virtue of these interesting results we believe that this method deserves further investigations for its generalization to more challenging problems, such as, for example, the generation of hexahedral grids on three-dimensional domains, and the generation of adaptive grids.

In Section 2 we illustrate the variational formulation of the grid generation problem and the corresponding steepest descent technique. In Section 3 we propose some algorithms for the generation of quadrilateral grids. In Section 4 we present the active set strategy used in the second phase of the proposed method. In Section 5 we present some numerical experiments. In Section 6 we give our conclusions. In Appendices A and B we provide the explicit derivation of relevant formulas.

2. The variational method

Let \( \mathbb{N} \) and \( \mathbb{R} \) be the sets of natural and real numbers, respectively. Let \( n \in \mathbb{N} \), we denote with \( \mathbb{R}^n \) the \( n \)-dimensional real Euclidean space. Let \( A \) be a set with a finite number of elements, we denote with \(|A|\) its cardinality. Let \( v_1, v_2 \in \mathbb{R}^n \) we denote with \( v_1^T v_2 \) the usual scalar product of two vectors of \( \mathbb{R}^n \), where the superscript \( T \) denotes the transposition operation, with \( \| \cdot \| \) the Euclidean norm on \( \mathbb{R}^n \). Let

\[
A = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}
\]

and \( v_1, v_2 \in \mathbb{R}^2 \), then \( |v_1^T A v_2| \in \mathbb{R} \) is the area of a parallelogram with edges \( v_1, v_2 \). Let \( N, M \in \mathbb{N} \), we need the following sets of indices: \( I = \{(i, j), i = 0, 1, \ldots, N, j = 0, 1, \ldots, M\} \), \( I_1 = \{(i, j), i = 0, 1, \ldots, N-1, j = 0, 1, \ldots, M\} \), \( I_2 = \{(i, j), i = 0, 1, \ldots, N, j = 0, 1, \ldots, M-1\} \), \( I_0 = \{(i, j), i = 1, \ldots, N-1, j = 1, \ldots, M-1\} \), \( I_3 = I \setminus I_0 \). Let \( R = [0, 1] \times [0, 1] \) be the unit square, we denote with \( \mathcal{R} \) the uniform grid on \( R \) made of \( N \times M \) rectangles; its vertices have the following coordinates: \( \xi_{i,j} = (\xi_i^1, \xi_j^2)^T = (i/N, j/M)^T \in \mathbb{R}^2 \), \( (i, j) \in I \). Note that \( I_0 \) contains the indices of the internal vertices of \( \mathcal{R} \) and \( I_3 \) contains the indices of the boundary vertices of \( \mathcal{R} \). Let \( \Omega \subset \mathbb{R}^2 \) be a compact, simply connected domain, let \( \mathcal{U} = (u^1, u^2)^T : R \to \Omega \) be a parameterization of \( \Omega \), we denote with \( \mathcal{J} = \mathcal{U}(\mathcal{R}) \) the quadrilateral grid on \( \Omega \) whose vertices are

\[
u_{i,j} = (u_{i-1,j}, u_{i,j}^2)^T = (u^1(\xi_{i,j}), u^2(\xi_{i,j}))^T \in \mathbb{R}^2, \quad (i, j) \in I.
\]

The grid generation problem can be formulated in the following way: given a parameterization \( \mathcal{U}_{\partial} : \partial R \to \partial \Omega \) of the boundary \( \partial \Omega \) of \( \Omega \), compute a parameterization \( \mathcal{U} = (u^1, u^2)^T : R \to \Omega \) of the domain \( \Omega \subset \mathbb{R}^2 \), such that \( \mathcal{U}|_{\partial R} = \mathcal{U}_{\partial} \). In the following we show that some common geometric features of the quadrilateral grid \( \mathcal{J} = \mathcal{U}(\mathcal{R}) \) can be defined by the elements of the covariant metric tensor, see [10] for a more detailed discussion. Let \( \tilde{\xi} = (\tilde{\xi}^1, \tilde{\xi}^2)^T \in \mathbb{R} \), the Jacobi matrix is defined as

\[
J(\tilde{\xi}) = \begin{pmatrix} u^1_{\tilde{\xi}^1}(\tilde{\xi}) & u^1_{\tilde{\xi}^2}(\tilde{\xi}) \\ u^2_{\tilde{\xi}^1}(\tilde{\xi}) & u^2_{\tilde{\xi}^2}(\tilde{\xi}) \end{pmatrix},
\]

where, for \( h, k = 1, 2 \), \( u^h_{\tilde{\xi}^k} \) is the partial derivative of \( u^h \) with respect to \( \tilde{\xi}^k \); the Jacobian \( J(\tilde{\xi}) \) is defined as the determinant of \( J(\tilde{\xi}) \), that is \( J(\tilde{\xi}) = \det(J(\tilde{\xi})) \). Let \( u^h_{\tilde{\xi}^k} = (u^1_{\tilde{\xi}^k}, u^2_{\tilde{\xi}^k})^T \), \( h = 1, 2 \), \( g_{hk}(\tilde{\xi}) = u^T_{\tilde{\xi}^h}(\tilde{\xi}) u^k_{\tilde{\xi}^k}(\tilde{\xi}) \), \( h, k = 1, 2 \), the covariant metric tensor is defined as the matrix

\[
G(\tilde{\xi}) = \begin{pmatrix} g_{11}(\tilde{\xi}) & g_{12}(\tilde{\xi}) \\ g_{21}(\tilde{\xi}) & g_{22}(\tilde{\xi}) \end{pmatrix}.
\]
We denote with \( g(\hat{\zeta}) = \text{det}(G(\hat{\zeta})) \), and hence \( g(\hat{\zeta}) = J^2(\hat{\zeta}) \). For \( h = 1, 2 \), we denote with \( E_{i,j}^h \), \((i, j) \in I_h\) the edges of the quadrilaterals of \( \mathcal{Q} \), that is, the diagonal elements of a suitable minimization problem, where the objective function provides a quality measure of the grid.

**Length Functional:**

\[
I_L(u) = \frac{1}{2} \int_R \left( (g_{11} + g_{22}) d\zeta^1 d\zeta^2 \right) = \frac{1}{2} \int_R \left( (\nabla u^1)^T \nabla u^1 + (\nabla u^2)^T \nabla u^2 \right) d\zeta^1 d\zeta^2, 
\]

**Area Functional:**

\[
I_A(u) = \frac{1}{2} \int_R g d\zeta^1 d\zeta^2 = \frac{1}{2} \int_R \left( (\nabla u^1)^T A \nabla u^2 \right) d\zeta^1 d\zeta^2,
\]
where $\nabla$ denotes the gradient operator with respect to the Cartesian coordinates $\xi^1$, $\xi^2$, that is $\nabla u^h = (u^h_{\xi^1}, u^h_{\xi^2})^T$, $h = 1, 2$. Thus, given $w_L$, $w_A \geq 0$, we consider the following functional:

$$I_{w_L,w_A}(u) = w_L L(u) + w_A I_A(u)$$

and the minimizing problem

$$\min_{u \in \mathcal{D}} I_{w_L,w_A}(u),$$

where $\mathcal{D}$ is the set of all parameterizations $u = (u^1, u^2)^T : R \rightarrow \Omega$ of $\Omega$ such that integrals (5), (6) are well defined, and $u|_{\partial R} = u_{\partial R}$. For each $v = (v^1, v^2)^T : R \rightarrow \mathbb{R}^2$ such that integrals (5), (6) are well defined, and $v|_{\partial R} = 0$, we have

$$\lim_{t \rightarrow 0^+} \frac{I_{w_L,w_A}(u + tv) - I_{w_L,w_A}(u)}{t} = \nabla I_{w_L,w_A}(u) \circ v,$$

where “$\circ$” denotes the scalar product in the Hilbert space of square integrable functions $f : R \rightarrow \mathbb{R}^2$, see [12, p. 54] for details. We note that $\nabla I_{w_L,w_A}(u)$ is a linear functional that maps $v = (v^1, v^2)^T : R \rightarrow \nabla I_{w_L,w_A}(u) \circ v \in \mathbb{R}$.

From formulas (5)–(9) we have the following expression for the gradient of $I_{w_L,w_A}$:

$$\nabla I_{w_L,w_A}(u) = -(w_L \Delta u^1 + w_A \nabla((\nabla u^1)^T A \nabla u^2))^T A \nabla u^2,$$

$$w_L \Delta u^2 + w_A \nabla((\nabla u^2)^T A \nabla u^1))^T A \nabla u^1,$$

where $\Delta$ is the Laplacian operator with respect to the Cartesian coordinates $\xi^1, \xi^2$. In Appendix A there is a detailed derivation of expression (10).

From (10) we can obtain a steepest descent method for an approximation of the solution of (8), that is, starting from $u = u^{(0)}$, a sequence $\{u^{(v)}\}_{v \in \mathbb{N}}$ is iteratively generated by choosing $u^{(v+1)} = u^{(v)} - \lambda \nabla I_{w_L,w_A}(u^{(v)})$ along the direction $-\nabla I_{w_L,w_A}(u^{(v)})$. So that, given an initial approximation $u^{(0)}$ we compute subsequent approximations by the following formula:

$$u^{(v+1)} = u^{(v)} - \lambda \nabla I_{w_L,w_A}(u^{(v)}), \quad v = 0, 1, \ldots, \mathcal{V} - 1,$$

where $\lambda \in \mathbb{R}$, with $0 < \lambda < 1$, is a given parameter, and $\mathcal{V}$ is a suitable integer. For $v = 0, 1, \ldots, \mathcal{V}$, $(i, j) \in I^0$, we denote with $\mathcal{G}_{w_L,w_A;i,j}^{(v)}$, the approximation of $\nabla I_{w_L,w_A}(u^{(v)}(\xi_{i,j}))$ obtained by evaluating the derivatives of $u^{(v)}$ with the usual second-order central finite differences, see Appendix B for details. We denote with $\mathcal{G}_{i,j}^{(v)}$ the approximation of $\nabla I_{w_L,w_A}(u^{(v)}(\xi_{i,j}))$, obtained by the following formula:

$$\mathcal{G}_{i,j}^{(v+1)} = \mathcal{G}_{i,j}^{(v)} - \lambda \mathcal{G}_{w_L,w_A;i,j}^{(v)}, \quad (i, j) \in I^0, \quad v = 0, 1, \ldots, \mathcal{V} - 1.$$

Note that $\mathcal{G}_{i,j}^{(v)} = \mathcal{G}_{i,j}(\xi_{i,j})$, $(i, j) \in \partial I$, $v = 0, 1, \ldots, \mathcal{V}$, so only $\mathcal{G}_{i,j}^{(v)}$, $(i, j) \in I^0$, $v = 0, 1, \ldots, \mathcal{V}$ must be computed.

3. Algorithms for grid generation

We describe some grid generation algorithms based on formula (12). Let $\Omega \subset \mathbb{R}^2$ be a given domain, and $\mathcal{G}_0$ be a parameterization of its boundary. In the following, for $v = 0, 1, \ldots, \mathcal{V}$, we denote with $\mathcal{G}^{(v)}$ the quadrilateral grid having vertices $\mathcal{G}_{i,j}^{(v)} \in \mathbb{R}^2$, $(i, j) \in I$, such that $\mathcal{G}_{i,j}^{(v)} = \mathcal{G}_{i,j}(\xi_{i,j})$, $(i, j) \in \partial I$. Moreover we denote with $l_{\text{min}}$ the minimum length of the boundary edges of $\mathcal{G}^{(0)}$.

**Algorithm 1.** Let $\mathcal{G}^{(0)}$ be a given initial quadrilateral grid. Let $w_L$, $w_A \geq 0$ be given weights, let $\tau_0 \in \mathbb{R}$ be a given positive tolerance, let $\lambda \in \mathbb{R}$, with $0 < \lambda < 1$. Compute the quadrilateral grid $\mathcal{G}_{w_L,w_A}^{(v)}$ as the grid $\mathcal{G}^{(v)}$ obtained by the iterative formula (12), where $\mathcal{V}$ is chosen as the minimum integer $v$ such that $\|\mathcal{G}_{i,j}^{(v)} - \mathcal{G}_{i,j}^{(v-1)}\| \leq \tau l_{\text{min}}$, for every $(i, j) \in I^0$. 
is a smooth function. However, \( A = \) Fig. 2. The grids obtained by using Algorithm 1, with \( w_L = 1, w_A = 0, \tau_0 = 0.0001, \lambda = 0.005 \), for a convex domain, on the left-hand side, and for a nonconvex domain, on the right-hand side.

Let \( A \) be a given quadrilateral grid with vertices \( u_{i,j} = (u_{i,j}^1, u_{i,j}^2)^T \), \( (i, j) \in I \), such that \( u_{i,j} = u_{2i+1,2j} \), \( (i, j) \in \partial I \). For \((i, j) \in I_1 \cap I_2 \) we define the following quantities:

\[
\mathcal{A}_{1,i,j}(\mathcal{D}) = \frac{1}{2} (E_{i,j}^1)^T \Delta E_{i,j}^2, \quad \text{the oriented area of the triangle } T_{1,i,j} \text{ having vertices } u_{i,j}, u_{i+1,j}, u_{i,j+1},
\]

\[
\mathcal{A}_{2,i,j}(\mathcal{D}) = \frac{1}{2} (E_{i,j+1}^1)^T \Delta E_{i+1,j}^2, \quad \text{the oriented area of the triangle } T_{2,i,j} \text{ having vertices } u_{i,j+1}, u_{i,j+1}, u_{i,j+1},
\]

\[
\mathcal{A}_{3,i,j}(\mathcal{D}) = \frac{1}{2} (E_{i,j}^1)^T \Delta E_{i+1,j}^2, \quad \text{the oriented area of the triangle } T_{3,i,j} \text{ having vertices } u_{i,j}, u_{i,j+1}, u_{i,j+1},
\]

\[
\mathcal{A}_{4,i,j}(\mathcal{D}) = \frac{1}{2} (E_{i,j+1}^1)^T \Delta E_{i,j+1}^2, \quad \text{the oriented area of the triangle } T_{4,i,j} \text{ having vertices } u_{i,j}, u_{i,j+1}, u_{i,j+1}, u_{i,j+1}, \text{ see Fig. 3 for an example of these triangles.}
\]

Let

\[
\mathcal{A}_{\min}(\mathcal{D}) = \min \{ \mathcal{A}_{\mu,i,j}(\mathcal{D}) \}, \quad \mu = 1, 2, 3, 4, (i, j) \in I_1 \cap I_2.
\]

Note that \( \mathcal{A}_{\min}(\mathcal{D}) \leq 0 \) when the grid \( \mathcal{D} \) is folded or when \( \mathcal{D} \) has at least a nonconvex quadrilateral. Finally, we denote \( \mathcal{A} = \min \{ \text{Area}(\Omega)/2NM, L_{\min}^2/2 \} \).
Algorithm 2. Let $\mathcal{Q}^{(0)}$ be a given initial quadrilateral grid. Let $p, \lambda \in \mathbb{R}$, with $0 < p, \lambda < 1$, let $\tau_0, \tau_1 > 0$ be given tolerances. Compute the quadrilateral grid $\mathcal{Q}^{\tau_{wL,wA}}$ performing the following steps:

1. set $w_L = 1, v_A = 0, w_A = 1 / \mathcal{A}$;
2. compute grid $\mathcal{Q}^{\tau_{wL,vA}}$ using Algorithm 1 with weights $w_L, v_A$, tolerance $\tau_0$ and starting from $\mathcal{Q}^{(0)}$;
3. if $\mathcal{A}_{\min}(\mathcal{Q}^{\tau_{wL,vA}}) > \tau_1 \mathcal{A}$ then set $w_A = 0$ and go to step 7;
4. compute grid $\mathcal{Q}^{\tau_{wL,wA}}$ using Algorithm 1 with weights $w_L, w_A$, tolerance $\tau_0$ and starting from grid $\mathcal{Q}^{\tau_{wL,vA}}$;
5. if $\mathcal{A}_{\min}(\mathcal{Q}^{\tau_{wL,wA}}) > \mathcal{A}_{\min}(\mathcal{Q}^{\tau_{wL,vA}})$ then set $w_L = v_L, w_A = v_A$, and go to step 7;
6. if $\mathcal{A}_{\min}(\mathcal{Q}^{\tau_{wL,wA}}) > \tau_1 \mathcal{A}$ then go to step 7; else set $w_L = v_L, v_A = w_A, w_L = p \cdot w_L$ and go to step 4;
7. return grid $\mathcal{Q}^{\tau_{wL,wA}}$.

This algorithm computes grids $\mathcal{Q}^{\tau_{wL,wA}}$ using Algorithm 1, where the relative contributions of the Length and Area Functionals are determined automatically in order to obtain optimal grid quality measures. In particular, Algorithm 2 starts with weights $w_L = 1, w_A = 0$. When the corresponding grid does not satisfy the stopping criterion in step (3), it is computed the grid $\mathcal{Q}^{\tau_{wL,wA}}$ with weights $w_L = 1, w_A = 1 / \mathcal{A}$ and, iteratively, with a more and more high contribution of the Area Functional. This process terminates when the current grid $\mathcal{Q}^{\tau_{wL,wA}}$ satisfies either the stopping criterion in step (6) or the failure criterion in step (5). Note that this last criterion usually holds for grid obtained with a too small weight $w_L$, so a stopping criterion on the maximum number of iterations can be avoided. Algorithm 2 is able to deal with quite complex domains and its computational cost is proportional to the difficulty of the domain under consideration, in fact complex domains usually require many iterations while convex domains usually require only one iteration. However Algorithm 2 may give a folded grid when the domain $\Omega$ is much complex, see the example in Fig. 4.

4. The active set strategy

The active set strategy is a well-known optimization technique to deal with inequality constraints, where, at each iteration, only a subset of all the constraints is used to compute the search direction, see [8, Chapter 5] for a detailed discussion. We propose a similar approach to deal with folded grids $\mathcal{Q}$ obtained by Algorithm 2. For $\mu = 1, \ldots, 4$, $(i, j) \in I_1 \cap I_2$, we denote with $\nabla \mathcal{A}_{\mu,i,j}(\mathcal{Q}) \in \mathbb{R}^{(N-1)(M-1)}$ the gradient of area functions $\mathcal{A}_{\mu,i,j}(\mathcal{Q})$, with respect to the variables $u_{1,m}^h, u_{2,m}^h, (n, m) \in I^h$.

We recall that $\mathcal{A}_{1,i,j}(\mathcal{Q}) = \frac{1}{2} (E_{1,j}^1)^T A E_{1,j}^1 = \frac{1}{2} [(u_{i+1,j}^1 - u_{i,j}^1)(u_{i+1,j}^1 - u_{i,j}^1) - (u_{i+1,j}^2 - u_{i,j}^2)(u_{i+1,j}^2 - u_{i,j}^1)]$ so that the components of $\nabla \mathcal{A}_{1,i,j}(\mathcal{Q})$, for $(n, m) \in I^h$ and $h = 1, 2$, are given by

$$
\frac{\partial \mathcal{A}_{1,i,j}(\mathcal{Q})}{\partial u_{n,m}^h} = \begin{cases} 
(-1)^h (u_{i,j+1}^{3-h} - u_{i+1,j}^{3-h}) & \text{if } (n, m) = (i, j), \\
(-1)^h (u_{i+1,j}^{3-h} - u_{i,j+1}^{3-h}) & \text{if } (n, m) = (i + 1, j), \\
(-1)^h (u_{i+1,j}^{3-h} - u_{i,j}^{3-h}) & \text{if } (n, m) = (i, j + 1), \\
0 & \text{otherwise}.
\end{cases}
$$

(15)
In a similar way we obtain that the components of $\nabla A_{2,i,j}(\mathcal{Q})$, $\nabla A_{3,i,j}(\mathcal{Q})$ and $\nabla A_{4,i,j}(\mathcal{Q})$, for $(n, m) \in I^o$ and $h = 1, 2$, are given by

\[
\frac{\partial A_{2,i,j}(\mathcal{Q})}{\partial u_{n,m}^j} = \begin{cases} 
(-1)^h(u_{i+1,j+1}^3 - u_{i+1,j}^3) & \text{if } (n, m) = (i, j), \\
(-1)^h(u_{i,j+1}^3 - u_{i+1,j}^3) & \text{if } (n, m) = (i+1, j), \\
-1^h(u_{i,j}^3 - u_{i+1,j}^3) & \text{if } (n, m) = (i+1, j+1), \\
0 & \text{otherwise},
\end{cases}
\]

\[
\frac{\partial A_{3,i,j}(\mathcal{Q})}{\partial u_{n,m}^j} = \begin{cases} 
(-1)^h(u_{i+1,j+1}^3 - u_{i+1,j}^3) & \text{if } (n, m) = (i, j), \\
(-1)^h(u_{i,j+1}^3 - u_{i+1,j}^3) & \text{if } (n, m) = (i+1, j), \\
(-1)^h(u_{i,j}^3 - u_{i+1,j}^3) & \text{if } (n, m) = (i+1, j+1), \\
0 & \text{otherwise},
\end{cases}
\]

\[
\frac{\partial A_{4,i,j}(\mathcal{Q})}{\partial u_{n,m}^j} = \begin{cases} 
(-1)^h(u_{i,j+1}^3 - u_{i+1,j+1}^3) & \text{if } (n, m) = (i, j), \\
(-1)^h(u_{i+1,j+1}^3 - u_{i+1,j}^3) & \text{if } (n, m) = (i+1, j), \\
(-1)^h(u_{i,j+1}^3 - u_{i+1,j+1}^3) & \text{if } (n, m) = (i+1, j+1), \\
0 & \text{otherwise}.
\end{cases}
\]

Let $\mu_0 \in \{1, 2, 3, 4\}$ and $(i_0, j_0) \in I_1 \cap I_2$ be such that $A_{\mu_0,i_0,j_0}(\mathcal{Q}) = A_{\min}(\mathcal{Q})$. When Algorithm 2 fails we have that $A_{\mu_0,i_0,j_0}(\mathcal{Q})$ is under a given threshold, so we want to modify grid $\mathcal{Q}$ in order to increase the value of $A_{\mu_0,i_0,j_0}(\mathcal{Q})$. This modification should be performed by changing the coordinates $u_{1,m}^n, u_{2,m}^n, (n, m) \in I^o$ of the internal vertices of $\mathcal{Q}$ along the direction $\nabla A_{\mu_0,i_0,j_0}(\mathcal{Q})$; however, also the oriented areas of other triangles are usually changed by this operation. We avoid that the other oriented areas become too small by considering a suitable projection of $\nabla A_{\mu_0,i_0,j_0}(\mathcal{Q})$. Let $\tau_2$, with $0 < \tau_2 < 1$, be a given tolerance, let

\[
\Gamma_0 = \{ (\mu, i, j) : \mu = 1, 2, 3, 4, (i, j) \in I_1 \cap I_2, A_{\mu,i,j}(\mathcal{Q}) < \tau_2 A_{\mathcal{Q}} \},
\]

we denote with $M$ the matrix whose columns are given by vectors

\[
\nabla A_{\mu,i,j}(\mathcal{Q}) \| \nabla A_{\mu,i,j}(\mathcal{Q}) \|, \quad (\mu, i, j) \in \Gamma_0 \setminus \{ (\mu_0, i_0, j_0) \};
\]

finally, we denote with

\[
\nabla A_{\mu_0,i_0,j_0} = \frac{\nabla A_{\mu_0,i_0,j_0}(\mathcal{Q})}{\| \nabla A_{\mu_0,i_0,j_0}(\mathcal{Q}) \|};
\]

We can easily see that vector

\[
n_{\mu_0,i_0,j_0}(\mathcal{Q}) = (I - M(M^T M)^{-1} M^T) n_{\mu_0,i_0,j_0}
\]

is the projection of $n_{\mu_0,i_0,j_0}$ over the subspace orthogonal to the columns of $M$, that is $M^T n_{\mu_0,i_0,j_0} = 0$. We note that $\Gamma_0$ always contains $(\mu_0, i_0, j_0)$ and when $\Gamma_0 = \{ (\mu_0, i_0, j_0) \}$, vector $n_{\mu_0,i_0,j_0}$ is defined equal to $n_{\mu_0,i_0,j_0}$. When $\Gamma_0 \supset \{ (\mu_0, i_0, j_0) \}$, $M$ has $|\Gamma_0| - 1$ rows and $|\Gamma_0| - 1$ columns. From formulas (15)–(18) we have that each column of $M$ has at most six nonvanishing entries, so that $M^T M$ is easy to compute, moreover, $(M^T M)^{-1}$ has a low computational cost since its order, i.e., $|\Gamma_0| - 1$, is usually small; in all the numerical experiments shown in Section 5, $|\Gamma_0|$ has never exceeded 10. We note that vector $n_{\mu_0,i_0,j_0}(\mathcal{Q})$ is a slight modification of $n_{\mu_0,i_0,j_0}(\mathcal{Q})$, along which oriented areas $A_{\mu,i,j}$, $(\mu, i, j) \in \Gamma_0 \setminus \{ (\mu_0, i_0, j_0) \}$ do not change, since $n_{\mu_0,i_0,j_0}(\mathcal{Q})$ is orthogonal to $\nabla A_{\mu,i,j}(\mathcal{Q})$, $(\mu, i, j) \in \Gamma_0 \setminus \{ (\mu_0, i_0, j_0) \}$. Actually, this is the direction used to change the internal vertices of the grid $\mathcal{Q}$ under consideration. We have the final version of the proposed algorithm.
Algorithm 3. Let $\mathcal{Q}^{(0)}$ be a given initial quadrilateral grid. Let $p, \lambda, \gamma$, with $0 < p, \lambda, \gamma < 1$ be given parameters, let $\tau_0, \tau_1, \tau_2, \tau_3, \tau_4$ be given tolerances. Compute the quadrilateral grid $\mathcal{Q}$ as follows:

1. Let $\mathcal{Q}^{(0)}$ be a given initial quadrilateral grid.
2. Set $v = 0$, and reset $\mathcal{Q}^{(0)} = \mathcal{Q}^{(0)}_{ul, wA}$.
3. If $\mathcal{A}_{\min}(\mathcal{Q}^{(v)}) \geq \tau_3 \mathcal{A}$, then go to step (7).
4. Compute vector $w_{\mu_0, i_0, j_0}(\mathcal{Q}^{(v)}) \in \mathbb{R}^{2(N-1)(M-1)}$ using formulas (15)–(22).
5. If $\|w_{\mu_0, i_0, j_0}(\mathcal{Q}^{(v)})\| \leq \tau_4 \mathcal{A}_{\min}$, then go to step (7).
6. Compute $u_{i,j}^{(v+1)} = u_{i,j}^{(v)} + \gamma w_{\mu_0, i_0, j_0}(\mathcal{Q}^{(v)})/\|w_{\mu_0, i_0, j_0}(\mathcal{Q}^{(v)})\|$, $(i, j) \in I^*$, increase $v$ by one and go to step (3).
7. Return grid $\mathcal{Q} = \mathcal{Q}^{(v)}$.

Note that tolerance $\tau_4$ is usually chosen equal to tolerance $\tau_0$ of Algorithm 1. Moreover, when tolerance $\tau_3$ of termination criterion (3) is equal to tolerance $\tau_1$ of Algorithm 2 and this last algorithm does not fail, we have that Algorithm 3 reduces to Algorithm 2. On the contrary, when Algorithm 2 fails, we expect to deal with a quite difficult instance of the grid generation problem; in these cases $\tau_3 = \tau_1$ is usually inappropriate, so $\tau_3 < \tau_1$ is a usual choice. Moreover, when Algorithm 2 fails, the grid $\mathcal{Q}^{(v)}_{ul, wA}$ is used as the initial guess for an iterative procedure, where, in each step, the current grid is improved along the direction $w_{\mu_0, i_0, j_0}$ given by formula (22). Step (5) gives a failure criterion, in fact vectors $w_{\mu_0, i_0, j_0}$ having too small components are usually ineffective; similar results can be obtained by using a failure criterion on the maximum number of iterations. Finally, we note that $\Gamma_0$ gives the working set of the active set strategy proposed in Algorithm 3; the constraints implicitly taken into account are: $\mathcal{A}_{\mu,i,j}(\mathcal{Q}) \geq \tau_3 \mathcal{A}$, $\mu = 1, 2, 3, 4$, $(i, j) \in I_1 \cap I_2$.

5. Numerical experiments

We consider some numerical results obtained with the proposed algorithm. In these examples, the domains come from Rogue’s gallery, that is a well-known set of test problems for grid generation algorithms on planar domains, see [9] for details. In particular, these results are computed by using Algorithm 3 with the following parameters: $\tau_0 = \tau_4 = 0.0001$,
Fig. 6. The results obtained by using Algorithm 3: on the left, the domain $\mathcal{D}$, in the middle, the initial grid $Q^0$, on the right, the grid $Q$ obtained by using Algorithm 3. For each grid we also report some performance indices.

$\text{Horseshoe: } NM = 120, \theta_M = 160.89^\circ, \theta_m = 19.82^\circ, \sigma_\theta = 48.70^\circ, t = 1.28$

$\text{S: } NM = 80, \theta_M = 125.79^\circ, \theta_m = 55.60^\circ, \sigma_\theta = 14.90^\circ, t = 1.07$

$\text{Valley: } NM = 120, \theta_M = 138.25^\circ, \theta_m = 39.16^\circ, \sigma_\theta = 26.36^\circ, t = 1.53$

$\text{Chevron: } NM = 70, \theta_M = 136.07^\circ, \theta_m = 43.93^\circ, \sigma_\theta = 39.98^\circ, t = 0.48$

$\tau_1 = 0.5, \tau_2 = 0.25, \tau_3 = 0.2, \lambda = 0.005, \gamma = 0.3l_{\min}, p = 0.5$. The initial guess $\mathcal{D}^{(0)}$ is a grid with all the internal vertices $u_{i,j}^{(0)}, (i,j) \in I^0$ equal to a given point. This choice is the most simple one; a better initial guess can be obtained by the algebraic methods [9], and it can reduce the number of iterations in Algorithm 3. The results are reported in Figs. 5–8, here, for each example, we have: on the left, the domain $\Omega$, in the middle, the initial grid $\mathcal{D}^{(0)}$, and on the right, the grid $\mathcal{D}$ computed by using Algorithm 3. Moreover, for each example, we report the number of quadrilaterals $NM$ as well as some performance indices: $\theta_m$, i.e., the minimum internal angle of the quadrilaterals in $\mathcal{D}$, $\theta_M$, i.e., the maximum internal angle of the quadrilaterals in $\mathcal{D}$, $\sigma_\theta$, i.e., the standard deviation of the internal angles of the quadrilaterals in $\mathcal{D}$ from $90^\circ$, and $t$, i.e., the elapsed time, in seconds, for the computation of grid $\mathcal{D}$. In Table 1 we have the number of iterations in each algorithm for each example. We note that the results reported in Figs. 5 and 6 are practically computed by using only Algorithm 2; more precisely, for these grids Algorithm 3 reduces to Algorithm 2. Instead, for grids shown in Figs. 7 and 8, Algorithm 2 terminates with failure; more precisely, for these examples Algorithm 2 computes a folded grid, with the exception of the Swan example, where it computes only a low-quality grid. Figs. 5–8 show satisfactory results, in fact all the grids $\mathcal{D}$ computed by Algorithm 3 are made of convex quadrilaterals (see index $\theta_M$). Note that particularly fulfilling results are those obtained for SS domain, which is commonly considered a difficult domain, see [6] for details. Finally, we note that the computational time of Algorithm 3 strictly depends on the difficulty of the domain under consideration. This is due to the fact that Algorithm 3 reduces to Algorithm 2 for no hard domains, and Algorithm 2 reduces to Algorithm 1 with $w_L = 1, w_A = 0$ for convex domains or for particular domains, as we can see from Table 1.
6. Conclusions

We proposed an algorithm for the computation of quadrilateral grids on planar domains. This algorithm is based on a steepest descent approach and on a technique resembling the well-known active set strategy for constrained optimization problems. The steepest descent approach is used to solve a variational formulation of the grid generation problem that is similar to ones proposed in [9,1]. The main contribution of the present paper is the active set strategy that improves the robustness of the whole algorithm. We provided the numerical results obtained by this algorithm on a large number of test problems. These results show that the proposed algorithm is an efficient and robust procedure to compute a numerical solution of the grid generation problem.

The computational cost of the algorithm is linear in the number of the grid vertices and in the number of the iterations performed, so the proposed algorithm can quickly compute small and easy grids and with an appropriate computational
time, large and harder grids. Actually, in the present version, the proposed algorithm is able to solve problems having a difficulty similar to the ones considered in Section 5; however, it can be easily generalized to harder planar domains by using block decomposition techniques, see [4] for details. Moreover, this algorithm is a promising starting point for the numerical solution of more involved grid generation problems such as, for example, the generation of hexahedral grids on three-dimensional domains, and the generation of adaptive grids.

Appendix A.

We provide an explicit derivation of formula (10). Let \( l, m : \mathbb{R} \to \mathbb{R}, w = (w^1, w^2)^T : \mathbb{R} \to \mathbb{R}^2 \) be smooth functions, we denote with \( \text{div} w = w^1_{\xi_1} + w^2_{\xi_2} \) the divergence of \( w \), and we use the following basic vectorial relations:

\[
(\nabla l)^T \nabla m = l_{\xi_1} m_{\xi_1} + l_{\xi_2} m_{\xi_2} = (\nabla m)^T \nabla l,
\]

\[
(\nabla l)^T \nabla m = l_{\xi_1} m_{\xi_1} - l_{\xi_2} m_{\xi_2} = -(\nabla l)^T \nabla m,
\]

\[
(\nabla l)^T \nabla m = -m \Delta l + \text{div}(m \nabla l),
\]

\[
\nabla l \cdot \nabla m = l \text{div } w - \text{div}(lw),
\]

\[
\text{div}(m \nabla l) = -(\nabla m)^T A \nabla l,
\]

moreover the Green’s Theorem gives

\[
\int_{\mathcal{R}} \text{div } w = \int_{\partial \mathcal{R}} -w^2 \, d\xi_1 + w^1 \, d\xi_2,
\]

so that, if \( w|_{\partial \mathcal{R}} \equiv 0 \) we have

\[
\int_{\mathcal{R}} \text{div } w = 0.
\]

Let \( u = (u^1, u^2)^T, \nu = (\nu^1, \nu^2)^T : \mathbb{R} \to \mathbb{R}^2 \) be two smooth functions, such that \( \nu|_{\partial \mathcal{R}} \equiv 0 \), and let \( t \in \mathbb{R} \). By using (A.1) and (A.2) we have

\[
(\nabla (u^h + t \nu))^T \nabla (u^h + t \nu) = (\nabla u^h)^T \nabla u^h + 2t (\nabla \nu^h)^T \nabla \nu^h + o(t), h = 1, 2,
\]

Table 1
The number of iterations in each algorithm for each example

<table>
<thead>
<tr>
<th>Domain</th>
<th>NM</th>
<th>Algorithm 1</th>
<th>Algorithm 2</th>
<th>Algorithm 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airfoil</td>
<td>160</td>
<td>2677</td>
<td>769</td>
<td></td>
</tr>
<tr>
<td>Annulus</td>
<td>50</td>
<td>828</td>
<td>285</td>
<td>–</td>
</tr>
<tr>
<td>C</td>
<td>75</td>
<td>969</td>
<td>1618</td>
<td>–</td>
</tr>
<tr>
<td>Horseshoe</td>
<td>70</td>
<td>1316</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Shell</td>
<td>80</td>
<td>1316</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Valley</td>
<td>120</td>
<td>1545</td>
<td>398</td>
<td>–</td>
</tr>
<tr>
<td>Chevron</td>
<td>120</td>
<td>1545</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Swan</td>
<td>100</td>
<td>1545</td>
<td>398</td>
<td>–</td>
</tr>
<tr>
<td>Backstep</td>
<td>40</td>
<td>704</td>
<td>1391</td>
<td>9</td>
</tr>
<tr>
<td>Plow</td>
<td>56</td>
<td>1052</td>
<td>575</td>
<td>14</td>
</tr>
<tr>
<td>Tie</td>
<td>49</td>
<td>1445</td>
<td>1820</td>
<td>3</td>
</tr>
<tr>
<td>SS</td>
<td>60</td>
<td>650</td>
<td>213</td>
<td>2104</td>
</tr>
<tr>
<td>SS</td>
<td>240</td>
<td>1910</td>
<td>779</td>
<td>12115</td>
</tr>
</tbody>
</table>
\[
((\nabla(u^1 + tv^1))^T A \nabla(u^2 + tv^2))^2
\]
\[
= \left[((\nabla u^1)^T A \nabla u^2 + (\nabla v^1)^T A \nabla u^2)^2 + o(t)\right]^2
\]
\[
= (\nabla u^1)^T A \nabla u^2)^2 + 2t(\nabla u^1)^T A \nabla u^2
\times ((\nabla v^1)^T A \nabla u^2 + (\nabla u^1)^T A \nabla v^2) + o(t)
\]
\[
= (\nabla u^1)^T A \nabla u^2)^2 + 2t(\nabla u^1)^T A \nabla u^2
\times ((A \nabla u^2)^T A \nabla v^1 - (A \nabla u^1)^T A \nabla v^2) + o(t),
\]
(A.9)

where
\[
\lim_{t \to 0} \frac{o(t)}{t} = 0.
\]

From (A.8) and (5) we have
\[
I_L(u + tv) = I_L(u) + t \int_R ((\nabla u^1)^T A \nabla v^1 + (\nabla u^2)^T A \nabla v^2) \, d\xi^1 \, d\xi^2 + o(t),
\]
(A.10)

from (A.9) and (6) we have
\[
I_A(u + tv) = I_A(u) + t \int_R ((\nabla u^1)^T A \nabla u^2
\times ((A \nabla u^2)^T A \nabla v^1 - (A \nabla u^1)^T A \nabla v^2)) \, d\xi^1 \, d\xi^2 + o(t).
\]
(A.11)

Thus, from (7), (A.10), (A.11), we have
\[
\lim_{t \to 0^+} \frac{I_{W_L,W_A}(u + tv) - I_{W_L,W_A}(u)}{t}
\]
\[
= \lim_{t \to 0^+} \frac{w_L I_L(u + tv) + w_A I_A(u + tv) - w_L I_L(u) - w_A I_A(u)}{t}
\]
\[
= w_L \int_R ((\nabla u^1)^T A \nabla v^1 + (\nabla u^2)^T A \nabla v^2) \, d\xi^1 \, d\xi^2
\]
\[
+ w_A \int_R ((\nabla u^1)^T A \nabla u^2((A \nabla u^2)^T A \nabla v^1 - (A \nabla u^1)^T A \nabla v^2)) \, d\xi^1 \, d\xi^2
\]
\[
= \int_R (w_L(\nabla u^1)^T A \nabla v^1 + w_A m(\nabla u^1)^T A \nabla v^1) \, d\xi^1 \, d\xi^2
\]
\[
+ \int_R (w_L(\nabla u^2)^T A \nabla v^2 - w_A m(\nabla u^1)^T A \nabla v^2) \, d\xi^1 \, d\xi^2,
\]
(A.12)

where \( m = (\nabla u^1)^T A \nabla u^2. \)

For \( h = 1, 2 \) we have
\[
\int_R (\nabla u^h)^T A \nabla v^h \, d\xi^1 \, d\xi^2 = \int_R (-v^h \Delta u^h + \text{div}(v^h \nabla u^h)) \, d\xi^1 \, d\xi^2
\]
\[
= -\int_R \Delta u^h v^h \, d\xi^1 \, d\xi^2,
\]
(A.13)

where the first identity is obtained by (A.3) and the second one by (A.7).
For \( h, k = 1, 2 \) we have
\[
\int_R m(A \nabla u^h)^T \nabla v^k \, d\xi^1 \, d\xi^2 \\
= \int_R v^k \text{div}(m A \nabla u^h) \, d\xi^1 \, d\xi^2 - \int_R \text{div}(v^k m A \nabla u^h) \, d\xi^1 \, d\xi^2 \\
= - \int_R v^k (\nabla m)^T A \nabla u^h \, d\xi^1 \, d\xi^2 ,
\]
where the first identity is obtained by (A.4) and the second one by (A.5) and (A.7).

This concludes the derivation of formula (10).

Appendix B.

We provide a description of the finite difference quotients used in (12), and a detailed derivation of formula (13).

For \( v = 0, 1, \ldots, ^v, (i, j) \in I^v, G_{i,j}^{(v)} \) denotes the approximation of \( u^{(v)}(\xi_{i,j}) \) and \( G_{\psi i,j}^{(v)} \) \( \psi = 1, 2 \) denotes the approximation of \( \nabla I_{w_L,w_A}(u^{(v)}(\xi_{i,j})) \) obtained by approximating the derivatives of \( u^{(v)} \) by the usual central finite differences, that is, for \((i, j) \in I^v, v = 0, 1, \ldots, ^v, \ h = 1, 2,\)
\[
\frac{u^{(v)}}{\psi h}(\xi_{i,j}) \approx \frac{u^{(v)}(\xi_{i,j} + \delta_h e^h) - u^{(v)}(\xi_{i,j} - \delta_h e^h)}{2 \delta_h} ,
\]
\[
\frac{u^{(v)}}{\psi \delta h}(\xi_{i,j}) \approx \frac{u^{(v)}(\xi_{i,j} + \delta_h e^h) + u^{(v)}(\xi_{i,j} - \delta_h e^h) - 2u^{(v)}(\xi_{i,j})}{\delta_h^2} ,
\]
\[
\frac{u^{(v)}}{\psi \delta^2}(\xi_{i,j}) \approx \frac{u^{(v)}_{i+1,j} + u^{(v)}_{i-1,j} - u^{(v)}_{i+1,j-1} - u^{(v)}_{i+1,j-1}}{4 \delta_1 \delta_2} ,
\]
where \( \delta_1, \delta_2 \) are the discretization steps.

When \( w_L = 1 \) and \( w_A = 0, \) formula (12) becomes
\[
u_{i,j}^{(v+1)} = v_{i,j}^{(v)} - \lambda G_{1,0,i,j}^{(v)}, \quad (i, j) \in I^v, \ v = 0, 1, \ldots, ^v - 1,
\]
where \( G_{1,0,i,j}^{(v)} \) is the approximation of
\[
\nabla I_{1,0}(u^{(v)}(\xi_{i,j})) = -(A(u^{(v)})^1, A(u^{(v)})^2)^T(\xi_{i,j}) \\
= \frac{u^{(v)}_{\psi \delta^1}(\xi_{i,j}) + u^{(v)}_{\psi \delta^2}(\xi_{i,j})}{\delta^1 \delta^2} ,
\]
where the first identity comes from (10), and \( (u^{(v)})^h, h = 1, 2 \) denotes the Cartesian components of \( u^{(v)}. \) For \((i, j) \in I^v, \ v = 0, 1, \ldots, ^v - 1, \) we have
\[
v_{i,j}^{(v+1)} = v_{i,j}^{(v)} - \lambda \left( \frac{u^{(v)}(\xi_{i,j} + \delta_1 e^1) + u^{(v)}(\xi_{i,j} - \delta_1 e^1) - 2u^{(v)}(\xi_{i,j})}{\delta_1^2} \right) \\
- \lambda \left( \frac{u^{(v)}(\xi_{i,j} + \delta_2 e^2) + u^{(v)}(\xi_{i,j} - \delta_2 e^2) - 2u^{(v)}(\xi_{i,j})}{\delta_2^2} \right) ,
\]
(B.6)
where we have used (B.2), (B.4) and (B.5). It is always possible to suppose $\delta_1 = \delta_2$, by considering a parameterization $u : \tilde{R} \to \Omega$ where $\tilde{R}$ is an ad hoc rectangle, moreover with an abuse of notation we denote with $\lambda$ the quantity $4 \lambda / \delta_1^2$. So that Eq. (B.6) becomes

$$u_{i,j}^{(v+1)} = u_{i,j}^{(v)} - \lambda \left[ \frac{u^{(v)}(\xi_{i+1,j}) - u^{(v)}(\xi_{i-1,j}) - 2u^{(v)}(\xi_{i,j})}{4} + \frac{u^{(v)}(\xi_{i,j+1}) - u^{(v)}(\xi_{i,j-1}) - 2u^{(v)}(\xi_{i,j})}{4} \right].$$

(B.7)

that is (13).

References