INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2000; **33**: 89–123

Algebraic–hyperbolic grid generation with precise control of intersection of angles

Karl-Heinz Brakhage[1] and Siegfried Müller*,[2]

Institut für Geometrie und Praktische Mathematik, RWTH Aachen, Templergraben 55, D-52056 Aachen, Germany

SUMMARY

A new approach to construct high quality meshes for flow calculations is investigated in the context of algebraic grid generation, where the angle of intersection between two alternate co-ordinate lines can be specified by the user. From an initial grid, new co-ordinate lines in *one* parameter direction are determined by solving a set of ordinary differential equations (ODE). These trajectories intersect the alternate co-ordinate lines of the initial grid in a *prescribed* angle. The resulting grid function is proven to be folding-free. The advantages with respect to grid quality achieved by prescribing the angles are gained at the expense that the boundary distribution can only be prescribed on three of four boundaries. In view of a *sparse* representation of the grid function, the points of intersection are interpolated by a B-spline surface. Thus, the grid can be accessed evaluating the generated parametric representation of the computational domain. Investigations of several test configurations have shown that useful grids with high resolution can be computed from the B-spline representation only depending on a *small number of locally chosen parameters*. Therefore, only a small amount of computational memory is required for storing these parameters and evaluating the grid function can be efficiently performed at reduced computational costs. This is particularly promising when the mesh has to be frequently changed or updated in time. Copyright © 2000 John Wiley & Sons, Ltd.

1. INTRODUCTION

Algebraic grid generation is a very efficient methodology by which a curvilinear grid can be computed (see e.g. [1]). However, there occur severe drawbacks when applied to the discretization of partial differential equations (PDE) related to grid foldings, lack of orthogonality and discontinuities in metric coefficients. These phenomena are caused by the limited possibilities of controlling grid properties. For example, Coons patches [2] are computed from boundary-dependent properties, e.g. point distribution, derivatives, curvature, etc., and appropriate blends [3] by which the grid properties in the interior are globally determined. Alternatively,

* Correspondence to: Institut für Geometrie und Praktische Mathematik, RWTH Aachen, Templergraben 55, D-52056 Aachen, Germany.
[1] E-mail: brakhage@igpm.rwth-aachen.de
[2] E-mail: mueller@igpm.rwth-aachen.de

Received August 1998
Revised July 1999

algebraic systems based on transfinite interpolation (see, e.g. [1,4]), where the interpolation functions can be interpreted as non-linear blend functions are more flexible with respect to local control of grid properties. Here, the quality of the resulting grid mainly depends on the choice of control points, e.g. deBoor points in B-spline representations, which have to be chosen judiciously. Despite these local techniques, for Navier–Stokes type of grids, i.e. thin boundary layers are to be resolved, a lack of robustness occurs in the interior of domains with complex boundary shape and curved surfaces respectively. Therefore, algebraically generated grids are mainly used in the context of elliptic grid generation as an initial guess for the iterative solution of the non-linear Poisson equations. Nevertheless, there is one useful feature inherent in the concept of algebraic grid generation, which makes reconsideration worthwhile. In general, the number of control points needed in the algebraic representation is *much smaller* than the resolution of the used grid itself. Hence, it is admissible to spend more computational time in the computation of an appropriate sparse control net. From this point of view, grid generation systems based on solving PDEs become feasible for the computation of the control points not yet known. Whenever the blend functions satisfy a uniformity condition, then the properties of the control net are also transferred to the grid itself [5,6].

Recently, Eiseman *et al.* [7] presented the concept of algebraic–elliptic grid generation, where the control net is computed by an elliptic generator. The combination of these control point curves and transfinite interpolation, where the blending functions of Eisman's control point form are local interpolation functions [5,6] yields a robust grid generator only depending on a small number of grid tuning parameters. However, the robustness essentially depends on the definition of appropriate control functions, which occur in the non-linear elliptic PDE of Poisson type to be solved. These are determined from the boundary point distribution (see e.g. [1,8]. Hence, there is no direct local control on the behavior in the interior of the domain.

Opposite to Eiseman's algebraic–elliptic concept, we follow a different approach based on hyperbolic systems for the distribution of the control points instead of an elliptic one. However, we only have to solve a set of ordinary differential equations (ODEs) instead of PDEs, where the resulting trajectories can be embedded in a hyperbolic system of inhomogeneous advection equations by means of the theory of characteristics. Hyperbolic systems are frequently investigated in the context of orthogonal grid generation (see e.g. [1,9,10]. There are basically two types of orthogonal generation systems, namely (1) construction of an orthogonal system from a non-orthogonal system, and (2) field solutions of PDEs. Although orthogonality is a useful property with respect to truncation error (see [1], p. 331), this demand restricts the application to two-dimensional problems. In three-dimensional cases, orthogonality is difficult to achieve except for domains that are bounded by boundary surfaces satisfying certain conditions. Moreover, demanding orthogonality in the neighborhood of convex-shaped boundaries leads to the attraction of co-ordinate lines in the interior and vice versa in the concave case. Hence, orthogonality is opposed to the smoothness of grid point distribution, which is a characteristic feature in elliptic systems.

In this concept, we combine the demands of orthogonality and smoothness. To this end, we *replace* overall orthogonality by the condition that two alternate co-ordinate lines intersect in a *user-specified angle*. Hence, the user has direct *local* control of grid properties by locally determining the angle of intersection. The underlying construction principle is based on determining appropriate trajectories with respect to a set of given control lines, which coincide

with co-ordinate lines. From the points of intersection we determine an interpolating B-spline function. Then we compute the grid by evaluating the B-spline. Here, we aim to use as few as possible control points related to the points of intersection. This guarantees a sparse representation of the B-spline function, which in general is much smaller than the used resolution of the final grid. Moreover, remeshing, e.g. required by deformation of the computational domain or mesh adaptation due to flow quantities, can be performed by redistribution of the control points only. We want to emphasize that the idea of a sparse representation does not depend on the grid generation system for computing the control points. Instead of the hyperbolic approach, we may also apply elliptic generation systems [7].

In the following we would like to comment on related approaches already existing in the literature. First of all, we remark that our construction of the grid mapping is based on ODE techniques similar to those applied in the context of *orthogonal* grid generation, well known for at least two decades, see References [9, p. 29; 11]. However, we derive an ODE by means of computer aided geometric design (CAGD) techniques that take into account the already given angle of intersection. This ODE represents a non-trivial extension of the well-known ODE arising from orthogonal grid generation. Of course, globally prescribing an angle of 90°, our ODE coincides with the latter one.

Moreover, the principal idea of our concept is to prescribe a function representing the intersection angle of two alternate co-ordinate lines. This function can be interpreted as a control function, hence there is a connection to References [12,13]. However, we emphasize that the control functions introduced in the forementioned papers are incorporated in the construction of an *initial grid* only in order to control the grid spacing in one direction. This strategy has already been introduced by Eiseman [5,6,10] in his investigations on control surfaces. In our concept, we have also influence on the grid spacing in the second direction, i.e. alternate to the control curves of the initial grid, and moreover, we control it *locally* by the angle of intersection. Furthermore, we prove *analytically* and not only by construction that the resulting grids are folding-free. This even holds in the case where the boundary curves are not perpendicular at the corners. The quality of the resulting grids is verified by several examples, which are comparable with results reported in Reference [14]. Last but not least, we emphasize that we are aiming at a sparse representation of the grid mapping which is essential in view of modern flow solvers based on multiscale techniques recently developed [15,16]. From this we conclude that there exists some relationship to the above references. However, there are fundamental conceptual differences.

Next, we discuss limitations inherent in our approach. It is well known that in the context of orthogonal grid generation systems based on ODE solvers, the boundary point distribution can only be described on three of four boundaries. Obviously, this is also a deficiency of our method, which is compensated by the gain in grid quality. Moreover, in contrast to (strictly) orthogonal strategies we have a certain influence on the boundary distribution on the fourth boundary by the function of intersection angles that is to be prescribed. Furthermore, there exist numerous configurations arising from engineering applications, e.g. external flow fields around airplanes, where it suffices to describe the boundary on three boundaries only.

In our investigations, we restrict ourselves to two-dimensional problems only. However, we would like to emphasize that our approach is *not* restricted to planar two-dimensional domains. As we will verify by a Bezier patch it can be directly applied to surfaces. In principle,

the underlying concept is extensible to three-dimensional form. For more details on this topic we refer the reader to the conclusion (Section 6).

The paper is organized as follows. First of all, we describe the construction principle by which one trajectory is determined. Then we embed this principle in the definition of an analytic grid function. Of course, there exists no explicit representation of this function, since the initial value problems (IVPs) are in general not explicitly solvable. However, we can investigate this function analytically and deduce some helpful features. The latter are approximately transferred to the finally constructed algebraic grid function based on interpolation with B-spline functions. In the end, we demonstrate the flexibility of our concept by means of several test configurations.

2. CONSTRUCTION PRINCIPLE

We consider a domain $\Omega \subset \mathbb{R}^2$ which is always supposed to be a closed and simply connected domain bounded by four boundary curves. Furthermore, we assume that there exists a function $\mathbf{x}$: $[0, 1]^2 \to \Omega$ by which the parameter domain $[0, 1]^2$ is mapped onto the computational domain Ω. For example, the function $\mathbf{x}$ can be constructed as a blend of the given boundary curves (see Reference [2]). For more details see Section 4. In general, the co-ordinate lines $\mathbf{x}(\,.\,, v)$ and $\mathbf{x}(u, .\,)$ will not be orthogonal to each other. In order to derive an orthogonal function $\tilde{\mathbf{x}}$: $[0, 1]^2 \to \Omega$ by the initial one, we will now present the basic principle of how to construct a new curve corresponding to the u-line $\tilde{\mathbf{x}}(\,.\,, v)$ such that it is perpendicular to the family of all v-lines $\{\mathbf{x}(u, .\,)\}_{u\in[0,1]}$. In order to achieve this, we make the following construction. Maintaining the v-lines $\mathbf{x}(u, .\,)$, we determine orthogonal lines $\mathbf{y}(u) = \mathbf{x}(u, f(u))$ by the orthogonality condition

$$\mathbf{y}'(u)\mathbf{x}_v(u, f(u)) = 0$$

which is equivalent to

$$(\mathbf{x}_u(u, f(u)) + \mathbf{x}_v(u, f(u))f'(u))\mathbf{x}_v(u, f(u)) = 0 \tag{1}$$

This is an explicit equation for f' that can be resolved to

$$f'(u) = -\frac{\mathbf{x}_u(u, f(u))\mathbf{x}_v(u, f(u))}{\mathbf{x}_v(u, f(u))\mathbf{x}_v(u, f(u))}$$

Demanding $f(0) = v$, we have to solve an IVP for every parameter $v\in[0, 1]$. Omitting the arguments u and $f(u)$ and introducing the inner products $g_{11} = \mathbf{x}_u\mathbf{x}_u$, $g_{12} = \mathbf{x}_u\mathbf{x}_v$ and $g_{22} = \mathbf{x}_v\mathbf{x}_v$ this ODE can be shortly written as

$$f' = -\frac{g_{12}}{g_{22}}$$

In the context of orthogonal grid generation, this equation is known in the literature (see References [9, p. 29; 11]).

Analogously to the orthogonal case, we next want to derive these equations for arbitrary cosine values c: $[0, 1]^2 \to (-1, 1)$ given by the angles between $\mathbf{y}'$ and $\mathbf{x}_u$. For compatibility reasons, we suppose that $c(u, 0)$ is determined by the angles between $\mathbf{x}_u(u, 0)$ and $\mathbf{x}_v(u, 0)$ and $c(u, 1)$ is determined by the angles between $\mathbf{x}_u(u, 1)$ and $\mathbf{x}_v(u, 1)$ at the boundaries $\mathbf{x}(u, 0)$ and $\mathbf{x}(u, 1)$. This will guarantee that we do not leave the domain (see Section 3.1). In this case the basic equation is given by

$$\frac{\mathbf{y}'}{\|\mathbf{y}'\|} \frac{\mathbf{x}_v}{\|\mathbf{x}_v\|} = c \in (-1, 1)$$

Incorporating the notation of the metric coefficients from above this yields

$$\frac{g_{12} + g_{22} f'}{\sqrt{g_{11} + 2g_{12} f' + g_{22}(f')^2}\sqrt{g_{22}}} = c \tag{2}$$

which in contrast to Equation (1) is an implicit equation for f'. However, the only admissible solution turns out to be

$$f' = -\frac{g_{12}}{g_{22}} + \frac{c}{g_{22}} \sqrt{\frac{g_{11} g_{22} - g_{12}^2}{1 - c^2}}$$

We immediately verify that the right-hand side is well-defined, since $|c| < 1$ is supposed to hold. Furthermore, the term $g_{11} g_{12} - g_{12}^2$ is non-negative because it is just the determinant of the matrix $J^T J$, where $J = (\mathbf{x}_u, \mathbf{x}_v)$ is the Jacobian of the initial function $\mathbf{x}$. Whenever J is supposed to be regular, then $J^T J$ is positive definite, which implies that the determinant must be positive.

3. ANALYTICAL CONSTRUCTION

In the following we investigate the new co-ordinate line $\mathbf{y}$ defined by the initial function $\mathbf{x}$ and the trajectories f. To this end, we embed these lines in a new function $\tilde{\mathbf{x}}$. We will see that $\tilde{\mathbf{x}}$ can be interpreted as a solution of inhomogeneous advection equations, hence $\tilde{\mathbf{x}}$ can be alternatively interpreted as field solution of PDEs. In the special case of orthogonality, i.e. $c \equiv 0$, the function $\tilde{\mathbf{x}}$ can also be derived from well-known techniques known in the context of orthogonal grid generation.

3.1. Analytical investigations

First of all, we investigate the solution of the implicit Equation (2) for f' depending on the initial data $f(0) = v$ in some detail. To this end, we have to impose certain constraints on the initial function $\mathbf{x}$: $[0, 1]^2 \to \Omega \subset \mathbb{R}^2$ as well as on the function c: $[0, 1]^2 \to (-1, 1)$ corresponding to the angles of intersection to be constructed. In detail these are

(A1) $\mathbf{x}$ is twice continuously differentiable

(A2) the Jacobian $J(u, v) := (\mathbf{x}_u(u, v), \mathbf{x}_v(u, v)), \quad u, v \in [0, 1]$, is regular

and

(B1) $|c(u, v)| \leq q < 1$

(B2) c is continuous and Lipschitz-continuous in the second argument

(B3) $c(u, 0) = \dfrac{g_{12}(u, 0)}{\sqrt{g_{12}(u, 0) g_{22}(u, 0)}}, \qquad c(u, 1) = \dfrac{g_{12}(u, 1)}{\sqrt{g_{11}(u, 1) g_{22}(u, 1)}}, \quad u \in [0, 1]$

These assumptions guarantee that the function G: $[0, 1]^2 \to \mathbb{R}$ defined by

$$G(u, v) := \frac{c(u, v)\sqrt{\det(J^T(u, v) J(u, v))/(1 - c^2(u, v))} - g_{12}(u, v)}{g_{22}(u, v)} \tag{3}$$

is also continuous and Lipschitz-continuous in the second argument since the regularity assumption (A2) implies that the continuous metric coefficients g_{11} and g_{22} respectively are uniformly bounded in the compact parameter domain $[0, 1]^2$. Hence, we can apply standard results from the theory of ODEs (see, e.g. Reference [17]), which guarantee that the solution f_v of the IVP

$$f_v(0) = v, \qquad f_v'(u) = G(u, f_v(u)), \quad u > 0 \tag{4}$$

uniquely exists and is continuously differentiable for all $v \in [0, 1]$ and at least locally for small u. The constraint in u is primarily caused by the domain of definition for G, which is restricted to the parameter domain $[0, 1]$. However, we can remove this restriction by the consistency assumption (B3). For this purpose, we have to verify that $f_v(u)$ stays inside the domain $[0, 1]$, i.e.

$$f_v([0, 1]) \subset [0, 1], \quad v \in [0, 1] \tag{5}$$

Firstly, we observe that for $v = 0$ and $v = 1$ respectively, the unique solution is given by the constant-value function $f_v(u) = v$, $u \in [0, 1]$. To this end, we check that $G(\,.\,, v)$ vanishes for $u \in [0, 1]$, which immediately follows by assumption (B3). Hence, all trajectories f_v, $v \in [0, 1]$, are bounded by the trajectories f_0 and f_1 respectively. Otherwise, an integral curve f_v is to leave the parameter domain $[0, 1]$, which implies that two trajectories intersect. However, this leads to a contradiction to the unique existence of the solution of the IVP (4). Consequently, intersection of two distinct trajectories is excluded, i.e.

$$v_1, v_2 \in [0, 1], \quad v_1 \neq v_2 \Rightarrow f_{v_1}([0, 1]) \cap f_{v_2}([0, 1]) = \varnothing \tag{6}$$

Moreover, we notice that to any point in the parameter domain $[0, 1]^2$ there uniquely exists a trajectory f_v passing through this point, i.e.

$$\forall(\bar{u}, \bar{v})\in[0, 1]^2, \quad \exists_1 v\in[0, 1]: f_v(\bar{u}) = \bar{v} \tag{7}$$

The construction of an appropriate trajectory is based on reversing the integration direction in Equation (4), i.e. we consider $\tilde{f}_{\bar{v}}$: $[0, \bar{u}]\to[0, 1]$ defined as the solution of the IVP

$$\tilde{f}_{\bar{v}}(0) = \bar{v}, \qquad \tilde{f}_{\bar{v}}'(u) = -G(\bar{u}-\tilde{u}, \tilde{f}_{\bar{v}}(u)), \quad \tilde{u}\in[0, \bar{u}] \tag{8}$$

Analogous to f_v we verify that $\tilde{f}_{\bar{v}}([0, \bar{u}])\subset[0, 1]$ holds. Choosing $v := \tilde{f}_{\bar{v}}(\bar{u})$ and $f_v(u) := \tilde{f}_{\bar{v}}(\bar{u}-u)$, $u\in[0, \bar{u}]$, this obviously yields $f_v(\bar{u}) = \tilde{f}_{\bar{v}}(0) = \bar{v}$.

From the previous observations (5) and (6) we immediately conclude that the function $\mathbf{h}$: $[0, 1]^2\to[0, 1]^2$ defined by $\mathbf{h}(u, v) := (u, f_v(u))$ is bijective. Now, we can introduce a new function $\tilde{\mathbf{x}}$: $[0, 1]^2\to\Omega$ defined by

$$\tilde{\mathbf{x}}(u, v) := \mathbf{x}(\mathbf{h}(u, v)) = \mathbf{x}(u, f_v(u)) \tag{9}$$

As a composition of two bijective functions, notice that (B3) implies the bijectivity of $\mathbf{x}$, this function is also bijective.

Obviously, $\tilde{\mathbf{x}}(u, v)$ is the intersection point of the curves $\mathbf{x}(u, .)$ and $\mathbf{x}(., f_v(.))$. Thus, by definition of $\tilde{\mathbf{x}}$, the curves $\mathbf{x}(u, .)$ are only reparameterized except for $u = 0$, where the initial parameterization is preserved, i.e. for the boundary curves the following identities hold:

$$\tilde{\mathbf{x}}(., 0)\equiv\mathbf{x}(., 0), \quad \tilde{\mathbf{x}}(., 1)\equiv\mathbf{x}(., 1), \quad \tilde{\mathbf{x}}(0, .)\equiv\mathbf{x}(0, .), \quad \tilde{\mathbf{x}}(1, .)\equiv\mathbf{x}(1, f.(1))$$

This implies that a boundary point distribution can only be described on three of four boundary faces of Ω. This is a characteristic feature of orthogonal grid generation based on non-orthogonal grids.

In order to examine the smoothness of $\tilde{\mathbf{x}}$ it will be important to know how the solution f_v depends on the initial data $v\in[0, 1]$. Here, we can apply standard results from the theory of ODEs. For this purpose, we additionally have to assume that

(B4) c is continuously differentiable with respect to v.

This assumption, as well as (A1), imply that y: $[0, 1]^2\to\mathbb{R}$ defined by $y(u, v) := \partial G(u, v)/\partial v$ is continuous and uniformly bounded for all $(u, v)\in[0, 1]^2$. From this we conclude that the solution f_v of (4) is also continuously differentiable for all $u, v\in[0, 1]$. Hence, z: $[0, 1]^2\to\mathbb{R}$ defined by $z(u, v) := \partial f_v(u)/\partial v$ is well defined. Now, we are able to apply well-known results from the theory of ODEs (see, e.g. Reference [18], p. 408). However, we want to emphasize that in our case it is sufficient to consider $v\in[0, 1]$ instead of $v\in\mathbb{R}$, since (5) holds. By this we conclude that the solution of (4) is continuously differentiable for all $u, v\in[0, 1]$ and z is continuously differentiable with respect to u. Moreover, z is the solution of the IVP

$$\frac{\mathrm{d}z(u, v)}{\mathrm{d}u} = y(u, v)z(u, v), \quad z(0, v) = 1 \tag{10}$$

This is a linear homogeneous ODE, where the solution is given by

$$z(u, v) = \exp\left(\int_0^u y(w, v)\,\mathrm{d}w\right), \quad u \in [0, 1] \tag{11}$$

From the previous considerations we conclude that $\tilde{\mathbf{x}}$ is continuously differentiable and the derivatives can be written as

$$\begin{aligned}
\tilde{\mathbf{x}}_u(u, v) &= \mathbf{x}_u(u, f_v(u)) + \mathbf{x}_v(u, f_v(u))G(u, f_v(u)) \\
\tilde{\mathbf{x}}_v(u, v) &= \mathbf{x}_v(u, f_v(u))z(u, v)
\end{aligned} \tag{12}$$

where we apply (3) and (4). Furthermore, if we additionally assume that c is continuously differentiable with respect to u and twice continuously differentiable with respect to v respectively, then the second derivatives of $\tilde{\mathbf{x}}$ exist and are continuous too, and are given by

$$\begin{aligned}
\tilde{\mathbf{x}}_{uu}(u, v) &= (\mathbf{x}_{uu} + 2\mathbf{x}_{uv}G + \mathbf{x}_{vv}G^2 + \mathbf{x}_v(G_u + yG))\big|_{(u, f_v(u))} \\
\tilde{\mathbf{x}}_{uv}(u, v) &= \tilde{\mathbf{x}}_{vu}(u, v) = (\mathbf{x}_{uv} + \mathbf{x}_{vv}G)\big|_{(u, f_v(u))}z(u, v) + \mathbf{x}_v(u, f_v(u))z_u(u, v) \\
\tilde{\mathbf{x}}_{vv}(u, v) &= \mathbf{x}_{vv}(u, f_v(u))z^2(u, v) + \mathbf{x}_v(u, f_v(u))z_v(u, v)
\end{aligned}$$

Now, we are able to verify that two alternate co-ordinate lines determined by the new grid function $\tilde{\mathbf{x}}$ intersect with the angle implicitly given by c. To this end, we substitute the derivatives $\tilde{\mathbf{x}}_u$ and $\tilde{\mathbf{x}}_v$ according to (12) and get

$$\frac{\tilde{\mathbf{x}}_u}{\|\tilde{\mathbf{x}}_u\|}\frac{\tilde{\mathbf{x}}_v}{\|\tilde{\mathbf{x}}_v\|} = \frac{g_{11} + g_{22}G}{\sqrt{g_{11} + 2g_{12}G + g_{22}G^2}\sqrt{g_{22}}}\,\mathrm{sign}(z)$$

Here, we omit the arguments for simplification of representation. The positivity of z and a straightforward calculation yield

$$\frac{\tilde{\mathbf{x}}_u(u, v)}{\|\tilde{\mathbf{x}}_u(u, v)\|}\frac{\tilde{\mathbf{x}}_v(u, v)}{\|\tilde{\mathbf{x}}_v(u, v)\|} = c(u, f_v(u)) \tag{13}$$

Finally, we want to point out that whenever the identity

$$c(u, w) = \frac{g_{12}(u, w)}{\sqrt{g_{11}(u, w)g_{22}(u, w)}}, \quad w := f_v(u) \tag{14}$$

holds, then the solution of (4) remains constant, i.e. $f_v(u) = v$ since $G(u, f_v(u)) = 0$. Hence, the curve $\mathbf{x}(\,.\,, v)$ is preserved, i.e. $\tilde{\mathbf{x}}(\,.\,, v) = \mathbf{x}(\,.\,, v)$. In the following section, we will see that the function $\mathbf{x}$ can be split up along the integral curve f_v, if the angle is prescribed such that (14) is satisfied.

3.2. Advection equations

It is well known that the solution of ODEs corresponds to the solution of advection equations. In the following, we want to discuss how to interpret the function $\tilde{\mathbf{x}}$ as the solution of an inhomogeneous advection equation, where the characteristics are just the integral curves f_v. To this end, we consider the advection equation

$$\mathbf{w}_u(u, v) + G(u, v)\mathbf{w}_v(u, v) = \mathbf{x}_u(u, v) + \mathbf{x}_v(u, v)G(u, v), \quad u, v \in [0, 1]$$

$$\mathbf{w}(0, v) = \mathbf{x}(0, v), \quad v \in [0, 1] \tag{15}$$

The solution $\mathbf{w}$ of this inhomogeneous advection equation can be constructed according to the standard theory of characteristics (see, e.g. Reference [19], Chapter 2). Here, the characteristics are defined by the IVP

$$\frac{\mathrm{d}v(u)}{\mathrm{d}u} = G(u, v(u)), \quad v(0) = v_0$$

Obviously, the standard results cannot be directly applied since we have to make sure that the characteristics f_v stay inside of the domain [0, 1]. Otherwise, the right-hand side G is no longer well defined and the solution of (4) no longer exists as soon as the characteristic leaves this domain. Here, we notice that the curves $v(\,.\,)$ and $f_{v_0}(\,.\,)$ coincide, i.e. $v(u) = f_{v_0}(u)$, $u \in [0, 1]$. Hence, the characteristics uniquely exist for all $u \in [0, 1]$ and all initial values $v_0 \in [0, 1]$. In addition, given any $u, v \in [0, 1]$ then there also uniquely exists $v_0 \in [0, 1]$ such that $f_{v_0}(u) = v$, which follows by (7). Consequently, the solution of Equation (15) can be uniquely determined along the characteristic from the solution of the IVP

$$\frac{\mathrm{d}\mathbf{w}(u, f_{v_0}(u))}{\mathrm{d}u} = \mathbf{x}_u(u, f_{v_0}(u)) + \mathbf{x}_v(u, f_{v_0}(u))G(u, f_{v_0}(u)) = \tilde{\mathbf{x}}_u(u, v_0)$$

$$\mathbf{w}(0, f_{v_0}(0)) = \mathbf{w}(0, v_0) = \mathbf{x}(0, v_0) = \tilde{\mathbf{x}}(0, v_0)$$

where the right-hand side corresponds to the derivative $\tilde{\mathbf{x}}_u$ given by (12). By this relation, $\mathbf{w}$ and $\tilde{\mathbf{x}}$ are connected according to

$$\mathbf{w}(u, v) = \mathbf{x}(u, f_{v_0}(u)) = \tilde{\mathbf{x}}(u, v_0) \quad \text{with } f_{v_0}(u) = v$$

The advection equation (15) can be interpreted as Eulerian formulation of the underlying problem to construct $\tilde{\mathbf{x}}$, whereas defining $\tilde{\mathbf{x}}$ via the characteristics given by (4) is the

Lagrangian counterpart. However, we want to emphasize that constructing $\tilde{\mathbf{x}}$ by solving (15) is of no practical use since in any case we have to determine the solution of the characteristics f_v. Nevertheless, the Eulerian point of view is interesting since by the theory of characteristics discontinuities normal to the characteristic are admissible. Thus, the function c is allowed to have lines of discontinuity that coincide with a characteristic curve. This implies that the solution of the advection equation (15) is only piecewise smooth and, consequently, the function $\tilde{\mathbf{x}}$ can be split up into two independent domains if c is judiciously chosen on this curve, i.e. Equation (14) is satisfied on the characteristic.

3.3. Orthogonal grid generation

In the context of orthogonal grid generation there exist related concepts based on the construction of trajectories. We want to mention that for $c \equiv 0$, the above grid generation system based on the integral curves (4) has already been investigated by Watford [11]. However, the derivation is performed in a different and more complicated way. For a summary see Reference [9, p. 29]. In the end of Section 3, we finally want to present some relations to hyperbolic grid generation systems in the context of orthogonal grid generation, i.e. we assume $c \equiv 0$ in the following.

A well-known hyperbolic grid generation system has been derived by Steger and Chaussee [20], which is based on two equations

$$\bar{\mathbf{x}}_u \bar{\mathbf{x}}_v = 0, \qquad \bar{D} := \det(\bar{\mathbf{x}}_u, \bar{\mathbf{x}}_v) = \bar{x}_u \bar{y}_v - \bar{x}_v \bar{y}_u = \bar{V} \tag{16}$$

Here, the first equation guarantees orthogonality and grid-foldings are excluded by the second equation whenever $\bar{V}$ is supposed to be a non-vanishing continuous function. Choosing $\bar{V}$ judiciously, the function $\tilde{\mathbf{x}}$ can also derived by (16). For this purpose, we put

$$\bar{V}(u, v) := \tilde{D}(u, v) = D(u, f_v(u)) z(u, v)$$

where $D = \det(\mathbf{x}_u, \mathbf{x}_v) = x_u y_v - x_v y_u$ and $\tilde{D} = \det(\tilde{\mathbf{x}}_u, \tilde{\mathbf{x}}_v) = \tilde{x}_u \tilde{y}_v - \tilde{x}_v \tilde{y}_u$ denote the determinants of the Jacobian corresponding to the initial function $\mathbf{x}$ and the new grid function $\tilde{\mathbf{x}}$ respectively. Substituting the derivatives according to (12), it is a simple task to check that the conditions (16) are satisfied by $\tilde{\mathbf{x}}$.

Another frequently applied hyperbolic grid generation system is based on the equations (see [9])

$$\bar{\mathbf{x}}_u = F \bar{\mathbf{x}}_v^{\perp}, \qquad \bar{\mathbf{x}}(0, v) = \mathbf{x}_{0v}(v) \tag{17}$$

Here, $\bar{\mathbf{x}}_v^{\perp} := (\bar{y}_v, -\bar{x}_v)^T / \|\bar{\mathbf{x}}_v\|$ denotes one of the two orthonormal vectors to $\bar{\mathbf{x}}_v$. Furthermore, the scalar function $F = F(u, v)$ has to be judiciously chosen such that the first-order system (17) is hyperbolic. Otherwise, the system will not be well posed. In order to preserve the orientation of the co-ordinates, the sign of F is taken to be positive. Here, the function $\tilde{\mathbf{x}}$ is also a solution of (17), whenever F is chosen as

$$F(u, v) := \frac{\tilde{D}(u, v)}{\|\tilde{\mathbf{x}}_v(u, v)\|} = \frac{D(u, f_v(u))}{\|\mathbf{x}_v(u, f_v(u))\|}$$

where the determinant corresponding to the initial grid function has to be positive, i.e. $D > 0$. Since the grid map $\tilde{\mathbf{x}}$ is orthogonal, which follows by (13), a scalar F exists such that $\tilde{\mathbf{x}}_u = F\tilde{\mathbf{x}}_v^{\perp}$. Multiplying this relation by $\tilde{\mathbf{x}}_v^{\perp}$ yields a scalar equation for F. Substituting (12) immediately yields the representation of F, which is positive if $D > 0$ holds.

4. NUMERICAL CONSTRUCTION

In general, there exists no explicit representation of the solutions f_v of the IVP (4). Thus, the mapping $\tilde{\mathbf{x}}$ is only given implicitly. However, in practice we do not need the solutions f_v for all $v \in [0, 1]$ in any point $u \in [0, 1]$, but for certain given knots u_i, $i = 0, \ldots, N+1$ and v_j, $j = 0, \ldots, M+1$ respectively. By the boundary point distribution, we are then able to compute a net of control points $\tilde{\mathbf{x}}(u_i, f_{v_j}(u_i))$, where we apply an ODE solver to (4) in order to compute an approximation for $f_{v_j}(u_i)$. Since we know from (6) that the integral curves do not intersect, we can guarantee that the grid of the control points is also folding-free whenever the step size in the ODE solver is chosen sufficiently small. Here, an ODE solver with step size control is useful.

We now want to explain how to compute an algebraic grid function based on the approximation of the trajectories f_v. Here, we aim to realize this function with *as few parameters as possible*, i.e. a sparse net of control points is desirable, where we have control of grid properties by specifying the angles of intersection judiciously. The resulting algorithm is composed of five steps that are now described in detail.

Step 1

The starting point is a bounded domain $\Omega \subset \mathbb{R}^2$ where the boundary is composed of four parts, i.e. $\partial\Omega = \Gamma_{u0} \cup \Gamma_{u1} \cup \Gamma_{0v} \cup \Gamma_{1v}$. These boundaries are supposed to be represented by one-parametric, twice continuously differentiable curves

$$\mathbf{x}_{u0}\colon [0, 1] \to \Gamma_{u0}, \quad \mathbf{x}_{u1}\colon [0, 1] \to \Gamma_{u1}, \quad \mathbf{x}_{0v}\colon [0, 1] \to \Gamma_{0v}, \quad \mathbf{x}_{1v}\colon [0, 1] \to \Gamma_{1v}$$

Furthermore, the domain Ω is supposed to be simply connected, i.e. the boundary curves form a closed curve

$$\mathbf{x}_{00} := \mathbf{x}_{u0}(0) = \mathbf{x}_{0v}(0), \qquad \mathbf{x}_{10} := \mathbf{x}_{u0}(1) = \mathbf{x}_{1v}(0)$$

$$\mathbf{x}_{01} := \mathbf{x}_{u1}(0) = \mathbf{x}_{0v}(1), \qquad \mathbf{x}_{11} := \mathbf{x}_{u1}(1) = \mathbf{x}_{1v}(1)$$

Step 2

By the given boundary curves, we determine a twice continuously differentiable function $\mathbf{x}$: $[0, 1]^2 \to \Omega$ such that the Jacobian $J = (\mathbf{x}_u, \mathbf{x}_v)$ is regular. For compatibility reasons, we have to make sure that the following conditions hold:

$$\mathbf{x}(u, 0) = \mathbf{x}_{u0}(u), \qquad \mathbf{x}(u, 1) = \mathbf{x}_{u1}(u), \quad u \in [0, 1]$$

$$\mathbf{x}(0, v) = \mathbf{x}_{0v}(u), \qquad \mathbf{x}(1, v) = \mathbf{x}_{1v}(u), \quad v \in [0, 1]$$

Currently, the mapping $\mathbf{x}$ is computed as a Coons patch with linear blends. From CAGD, the linear blend for this boundary curves is well known. Here we write it in the following form:

$$\mathbf{x}(u, v) = (1-u, u)\begin{pmatrix} \mathbf{x}_{0v}(v) \\ \mathbf{x}_{1v}(v) \end{pmatrix} + (\mathbf{x}_{u0}(u), \mathbf{x}_{u1}(u))\begin{pmatrix} 1-v \\ v \end{pmatrix} - (1-u, u)\begin{pmatrix} \mathbf{x}_{00} & \mathbf{x}_{01} \\ \mathbf{x}_{10} & \mathbf{x}_{11} \end{pmatrix}\begin{pmatrix} 1-v \\ v \end{pmatrix} \quad (18)$$

If we additionally prescribe derivatives on the boundaries, e.g. orthogonality at the body, we have to modify our blend.

We would like to emphasize that the regularity assumption is no severe drawback. Of course, in the case of an interior corner angle larger than 180° the linear blend may correspond to a non-regular Jacobian. In this case we can apply a cubic blend with a judiciously chosen parameterization of the boundary. An example is presented in Section 5.

Step 3

The core ingredient of the algorithm is the definition of the function c: $[0, 1]^2 \to [-q, q]$, $0 < q < 1$, of cosine values corresponding to angles of intersection between two alternate co-ordinate lines. For compatibility reasons, we have to impose condition (B3), which guarantees that the integral curves f_v do not leave the parameter domain [0, 1] and, moreover, the boundary curves $\mathbf{x}(\,.\,, 0)$ and $\mathbf{x}(\,.\,, 1)$ respectively are preserved. In general, c is supposed to be continuously differentiable. However, if (14) is fulfiled then c is supposed to be continuously differentiable. However, if (14) is fulfiled then c is permitted to be discontinuous across the curve $(\,.\,, f_v(\,.\,))$.

In our numerical investigations, B-spline surfaces have been proven to be a flexible tool which enables local control on the resulting function. Hence, we suppose that the function c can be represented by a two-dimensional B-spline (surface)

$$c(u, v) = \sum_{i=0}^{M_c + k - 2} \sum_{j=0}^{N_c + k - 2} Q_{ij} N_i^k(u) N_j^k(v) \quad (19)$$

where the control points Q_{ij} as well as the knot vectors

$$T_c = (0 = t_0 = \cdots = t_{k-1} < \cdots < t_{M_c + k - 1} = \cdots = t_{M_c + 2(k-1)} = 1)$$

$$S_c = (0 = s_0 = \cdots = s_{k-1} < \cdots < s_{N_c + k - 1} = \cdots = s_{N_c + 2(k-1)} = 1) \quad (20)$$

have to be judiciously chosen by the user, i.e. depending on the configuration at hand. The order k of the B-spline functions N_i^k is chosen as 4 in our applications. For more analytical and algorithmical details on B-splines, see References [21] and [22] respectively.

In our test configurations we determine the control points in two steps. Firstly, we compute an interpolatory B-spline from the interpolatory conditions

$$c(u_i, v_j) = c_I(u_i, v_j), \quad i = 0, \ldots, M_c, \quad j = 0, \ldots, N_c \tag{21}$$

where c_I denotes the cosine function corresponding to the initial mapping determined by step 2, i.e.

$$c_I(u, v) := \frac{\mathbf{x}_u(u, v)}{\|\mathbf{x}_u(u, v)\|} \frac{\mathbf{x}_v(u, v)}{\|\mathbf{x}_v(u, v)\|}, \quad u, v \in [0, 1] \tag{22}$$

and the vanishing-curvature conditions

$$\begin{aligned} c_{uu}(u_0, v_j) &= c_{uu}(u_{M_c}, v_j) = 0, \quad j = 0, \ldots, N_c \\ c_{vv}(u_i, v_0) &= c_{vv}(u_i, v_{N_c}) = 0, \quad i = 0, \ldots, M_c \end{aligned} \tag{23}$$

as well as the vanishing-twist condition

$$c_{uv}(u_0, v_0) = c_{uv}(u_0, v_{N_c}) = c_{uv}(u_{M_c}, v_0) = c_{uv}(u_{M_c}, v_{N_c}) = 0 \tag{24}$$

where the parameters u_i, v_j have to be judiciously chosen in [0, 1]. By the conditions (21), (23) and (24), a linear system of equations for the control points Q_{ij} is determined, which can be written as

$$\mathbf{A}_u \mathbf{Q} \mathbf{A}_v^T = \mathbf{X} \tag{25}$$

where $\mathbf{A}_u$, $\mathbf{A}_v$ are tri-diagonal matrices, $\mathbf{Q}$ is the matrix of unknown control points and $\mathbf{X}$ is the matrix that is composed of the values to be interpolated. Solving this system requires an effort proportional to $M_c N_c$.

Next, we modify the resulting control points such that c vanishes close to the boundary Γ_{0v}. To this end, we put

$$Q_{ij} = 0 \quad (i, j) \in S \tag{26}$$

for a set $S \subset \{(i, j)\colon\ i = 0, \ldots, M_c,\ j = 0, \ldots, N_c\}$. Alternatively, we might have solved (21) with $c(u_i, v_j) = 0$ $(i, j) \in S$. However, this results in an oscillatory behavior of the B-spline. We would like to emphasize that the presented strategy on how to modify the control points strictly depends on the test configuration at hand. In view of an automatic grid generation system more sophisticated strategies have to be developed.

Finally, the modified control points are inserted in the B-spline representation (19) of c. Hence, c can be interpreted as a modification of c_I determined by the initial blend.

Step 4

In preparation of the approximate construction of the function $\tilde{\mathbf{x}}$ we determine a boundary point distribution in u- and v-direction of the parameter domain. For example, we can do this with respect to curvature of the boundary curves $\mathbf{x}_{u0}$ or $\mathbf{x}_{u1}$ and $\mathbf{x}_{0v}$ or $\mathbf{x}_{1v}$ respectively, such that

$$0 = u_0 < \cdots < u_{M_x} = 1, \qquad 0 = v_0 < \cdots < v_{N_x} = 1$$

where the numbers M_x and N_x should be small. Then, the solution f_{v_j} of (4) is approximately solved by a standard ODE solver. By this we compute the points of intersection $\tilde{\mathbf{x}}_{ij}$, $i = 0, \ldots, M_x$, $j = 0, \ldots, N_x$, where the co-ordinate line $\mathbf{x}(u_i, .)$ and the curve $\mathbf{x}(., f_{v_j}(.))$ are crossing.

Step 5

Finally, we apply a B-spline interpolation to the control points $\tilde{\mathbf{x}}_{i,j}$, $i = 0, \ldots, M_x$, $j = 0, \ldots, N_x$, which yields an approximation $\bar{\mathbf{x}}$ of the function $\tilde{\mathbf{x}}$ defined by (9). Specifically, we suppose that the final grid function $\bar{\mathbf{x}}$ can be written as

$$\bar{\mathbf{x}}(u, v) = \sum_{i=0}^{M_x + k - 2} \sum_{j=0}^{N_x + k - 2} \mathbf{P}_{ij} N_i^k(u) N_j^k(v) \tag{27}$$

where the control points $\mathbf{P}_{ij}$, frequently called deBoor points, are computed from the interpolatory conditions

$$\bar{\mathbf{x}} = (u_i, v_j) = \tilde{\mathbf{x}}_{ij}, \quad i = 0, \ldots, M_x; \quad j = 0, \ldots, N_x \tag{28}$$

and the interpolatory conditions for the derivatives at the boundaries

$$\bar{\mathbf{x}}_u(u_l, v_j) = \tilde{\mathbf{x}}_u(u_l, v_j), \quad l = 0, M_x; \quad j = 0, \ldots, N_x$$

$$\bar{\mathbf{x}}_v(u_i, v_l) = \tilde{\mathbf{x}}_v(u_i, v_l), \quad l = 0, N_x; \quad i = 0, \ldots, M_x \tag{29}$$

as well as the vanishing-twist conditions

$$\bar{\mathbf{x}}_{uv}(u_0, v_0) = \bar{\mathbf{x}}_{uv}(u_0, v_{N_x}) = \bar{\mathbf{x}}_{uv}(u_{M_x}, v_0) = \bar{\mathbf{x}}_{uv}(u_{M_x}, v_{N_x}) = \mathbf{0} \tag{30}$$

The knot vectors T_x, S_x are analogously defined to (20) where we put $t_{i+k-1} := u_i$, $i = 0, \ldots, M_x$ and $s_{j+k-1} := v_j$, $j = 0, \ldots, N_x$ respectively.

This is the same sparse linear system of equations for the control points P_{ij} as in (25) with $\mathbf{P}$ instead of $\mathbf{Q}$.

5. NUMERICAL EXAMPLES

So far, we derived a grid generator from analytical considerations and outlined the numerical construction. We now will consider numerous test configurations by which we investigate its capabilities. Here we do not only consider planar problems but we show also that it can be applied to surfaces.

5.1. Planar geometries

We present six planar configurations. These are determined by the points at the four edges and the boundary curves

(T1) *Convex-shaped boundary:*

$$\mathbf{x}_{00} = (5, 0)^T, \qquad \mathbf{x}_{10} = (5, 3)^T, \qquad \mathbf{x}_{01} = (0, 0)^T, \qquad \mathbf{x}_{11} = (0, 3)^T$$

$$\mathbf{x}_{u0}(u) = \mathbf{x}_{00} + u(\mathbf{x}_{10} - \mathbf{x}_{00}), \qquad \mathbf{x}_{u1}(u) = \mathbf{x}_{01} + u(\mathbf{x}_{11} - \mathbf{x}_{01})$$

$$\mathbf{x}_{1v}(v) = \mathbf{x}_{10} + v(\mathbf{x}_{11} - \mathbf{x}_{10}), \qquad \mathbf{x}_{0v}(v) = (5(1 - v),\ 0.75(\cos(2\pi v) - 1))^T$$

(T2) *Concave-shaped boundary:*

$$\mathbf{x}_{00} = (5, 0)^T, \qquad \mathbf{x}_{10} = (5, 3)^T, \qquad \mathbf{x}_{01} = (0, 0)^T, \qquad \mathbf{x}_{11} = (0, 3)^T$$

$$\mathbf{x}_{u0}(u) = \mathbf{x}_{00} + u(\mathbf{x}_{10} - \mathbf{x}_{00}), \qquad \mathbf{x}_{u1}(u) = \mathbf{x}_{01} + u(\mathbf{x}_{11} - \mathbf{x}_{01})$$

$$\mathbf{x}_{1v}(v) = \mathbf{x}_{10} + v(\mathbf{x}_{11} - \mathbf{x}_{10}), \qquad \mathbf{x}_{0v}(v) = (5(1 - v),\ -0.75(\cos(2\pi v) - 1))^T$$

(T3) *Parabolic-shaped boundary:*

$$\mathbf{x}_{00} = (5, 0)^T, \qquad \mathbf{x}_{10} = (6, 3)^T, \qquad \mathbf{x}_{01} = (1, 3)^T, \qquad \mathbf{x}_{11} = (0, 0)^T$$

$$\mathbf{x}_{u0}(u) = \mathbf{x}_{00} + u(\mathbf{x}_{10} - \mathbf{x}_{00}), \qquad \mathbf{x}_{u1}(u) = \mathbf{x}_{11} + u(\mathbf{x}_{01} - \mathbf{x}_{11})$$

$$\mathbf{x}_{1v}(v) = \mathbf{x}_{10} + v(\mathbf{x}_{01} - \mathbf{x}_{10}), \qquad \mathbf{x}_{0v}(v) = (5(1 - v),\ 4(1 - v)v)^T$$

(T4) *Four-foil region:*

$$\mathbf{x}_{00} = (1, -1)^T, \qquad \mathbf{x}_{10} = (1, 1)^T, \qquad \mathbf{x}_{01} = (-1, -1)^T, \qquad \mathbf{x}_{11} = (-1, 1)^T$$

$$\mathbf{x}_{u0}(u) = (1 + \cos((u - 0.5)\pi),\ \sin((u - 0.5)\pi))^T$$

$$\mathbf{x}_{u1}(u) = (-1 - \cos((u - 0.5)\pi),\ \sin((u - 0.5)\pi))^T$$

$$\mathbf{x}_{1v}(v) = (\cos(\pi v),\ 1 + \sin(\pi v))^T, \qquad \mathbf{x}_{0v}(v) = (\cos(\pi v),\ -1 - \sin(\pi v))^T$$

(T5) *NACA-0015 profile:*

$$\mathbf{x}_{00} = (1, 0.001575)^T, \qquad \mathbf{x}_{10} = (2, 0)^T, \qquad \mathbf{x}_{11} = (-1, 0)^T, \qquad \mathbf{x}_{01} = (0, 0)^T$$

$$\mathbf{x}_{u0}(u) = \mathbf{x}_{00} + u(\mathbf{x}_{10} - \mathbf{x}_{00}), \qquad \mathbf{x}_{u1}(u) = \mathbf{x}_{01} + u(\mathbf{x}_{11} - \mathbf{x}_{01})$$

$$\mathbf{x}_{1v}(v) = ((1 + 3\cos(\pi v))/2,\ 3\sin(\pi v)/2)^T$$

$$\mathbf{x}_{0v}(v) = (V^2,\ 0.15(1.4845V - 0.63V^2 - 1.758V^4 + 1.4215V^6 - 0.5075V^8))^T$$

with $V = \cos(0.5\pi v)$

(T6) *Lune-shaped region:*

$$\mathbf{x}_{00} = (5, 0)^T, \qquad \mathbf{x}_{10} = (5, 3 \times 10^{-6})^T, \qquad \mathbf{x}_{11} = (0, 3 \times 10^{-6})^T, \qquad \mathbf{x}_{01} = (0, 0)^T$$

$$\mathbf{x}_{u0}(u) = \mathbf{x}_{00} + u(\mathbf{x}_{10} - \mathbf{x}_{00}), \qquad \mathbf{x}_{u1}(u) = \mathbf{x}_{01} + u(\mathbf{x}_{11} - \mathbf{x}_{01})$$

$$\mathbf{x}_{1v}(v) = (5(1 - v),\ 5 \times 10^{-6} + 5(1 - v)v)^T$$

$$\mathbf{x}_{0v}(v) = (5(1 - v),\ -5(1 - v^2)v)^T$$

depending on the parameters $u, v \in [0, 1]$. By these settings, we compute the initial function $\mathbf{x}$ as a Coons patch with linear blends according to (18). Obviously, $\mathbf{x}$ is twice continuously differentiable and a straightforward calculation yields the regularity of its Jacobian.

By the initial grid function we compute an interpolatory B-spline function in the form (19) defined by Equations (21), (23) and (24) representing the cosine function c_I according to (22). Then we modify the control points of the resulting B-spline where we put them to zero along four layers next to the boundary curve, i.e., we perform (26) with $S = \{0, \ldots, 4\} \times \{1, \ldots, N_c - 1\}$ (T1, T2, T3, T5) and $S = (\{0, \ldots, 4\} \cup \{M_c - 3, \ldots, M_c\}) \times \{1, \ldots, N_c - 1\}$ (T4, T6) respectively.

In the following we discuss the characteristic features of the different test configurations. For test case (T1) we present *all* steps of the construction as outlined in Section 4, i.e. only here we compute an approximate B-spline for the presentation of the grid function. For the other configurations, we consider the exact grid mapping, by which we verify the *quality* of the resulting grid functions rather than *its sparse representation*. For the plots of the grid functions we use 21×21 (T1) and 27×27 (T2, T3, T4, T5, T6), parameter points which are equidistantly distributed except for (T1) and (T4). Furthermore, the cosine function corresponding to the initial blend is presented by 58×58 points in order to obtain a smooth plot. We emphasize that the number of control points generally is much smaller.

Test case T1

In Figure 1 the computational domain as well as some curves of the initial function $\mathbf{x}$ are plotted. In addition, the corresponding cosine values c_j are presented in Figure 2. We notice that c_I vanishes at the boundaries Γ_{u0}, Γ_{u1} and Γ_{1v} but not at Γ_{0v} i.e. $\mathbf{x}$ is orthogonal at three out of four boundaries.

If we now choose $c \equiv 0$, then we get an overall orthogonal grid function $\tilde{\mathbf{x}}(u, v) := \mathbf{x}(u, f_v(u))$. The corresponding trajectories f_v are shown in Figure 4 plotted in the $u - f_v(u)$ plane. As we can learn from Figure 3, there occurs a strong variation in the metric coefficients, which in grid generation is referred to as discontinuity in the metric coefficients, i.e. co-ordinate lines are

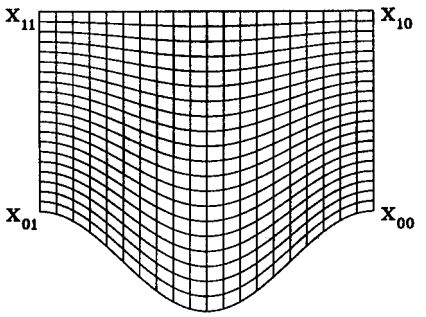

Figure 1. T1: Initial blend.

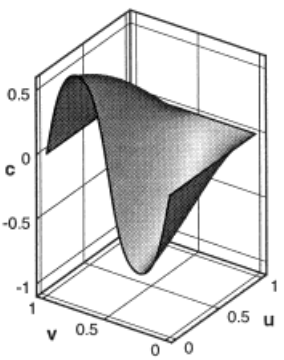

Figure 2. T1: c_I for initial blend.

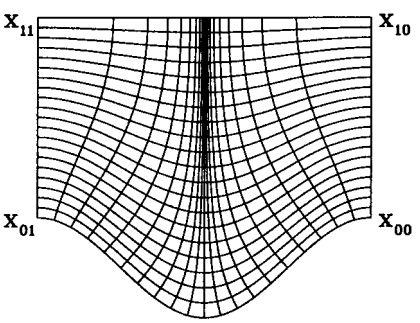

Figure 3. T1: Orthogonal grid function.

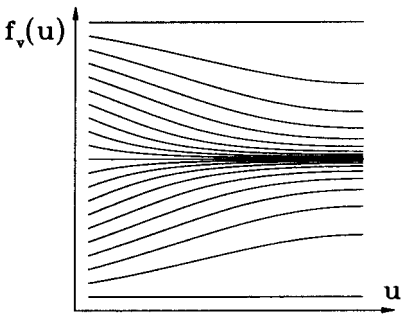

Figure 4. T1: Trajectories to $c \equiv 0$.

attracted to the line $v = 0.5$. This is a characteristic feature of orthogonal grid generation, which always arises in the presence of convex-shaped boundaries. In order to smooth the metric coefficients we have to weaken the orthogonality demand in the interior of the domain in order to avoid attraction of curves, whereas orthogonality at the boundaries is still desirable. For this purpose we modify the cosine function c_I determined by the initial blend. Putting some layers of control points corresponding to the B-spline representation to zero ensures orthogonality near to the boundary Γ_{0v}. From the new function c we now compute the trajectories f_v, which are presented in Figure 6. We notice that the trajectories are essentially straight lines with a stronger deviation for small u values, i.e. in comparison with the initial blend the new co-ordinate lines will only deviate near to the boundary Γ_{0v}. This behavior is reflected in Figure 5. Moreover, the new function $\tilde{\mathbf{x}}$ is orthogonal close to the boundary Γ_{0v}.

However, we do not compute $\tilde{\mathbf{x}}$ explicitly but approximate this function by a B-spline $\bar{\mathbf{x}}$ according to (27). Here, the main objective is to realize this function by as few as possible control points in order to get a sparse representation, which is important in view of complex three-dimensional configurations. In the present test case we choose $M_x = N_x = 10$. The net of control points is presented in Figure 7. From the function $\bar{\mathbf{x}}$ we are now able to compute the final grid just by cheap function evaluations. In Figure 8, we show a grid of Navier–Stokes

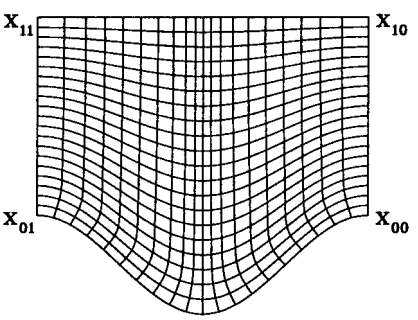

Figure 5. T1: Modified grid function.

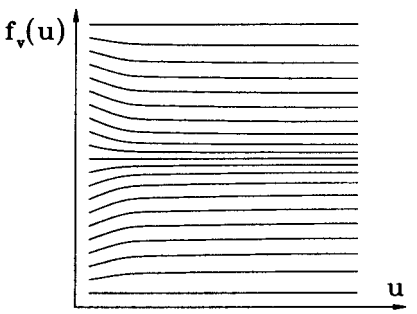

Figure 6. T1: Trajectories to modified c.

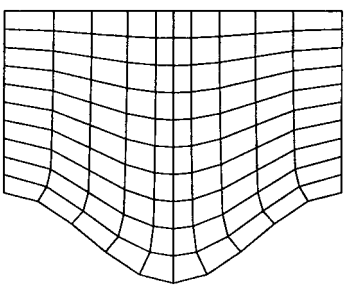

Figure 7. T1: Sparse net of control points.

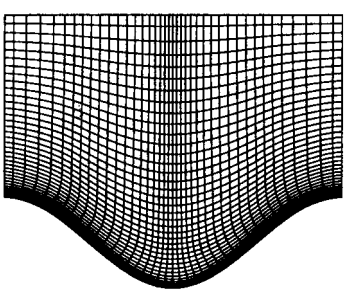

Figure 8. T1: Grid of Navier–Stokes type.

type, i.e. we attract the grid lines to the boundary Γ_{0v} by introducing a parameter function $\mathbf{s}(u, v) := (u^2, v)^T$ where we use 41×41 parameter values. In contrast to standard grid generation systems based on the solution of PDEs, we do not have to solve systems corresponding to the number of points in the final grid, here: about 10000, but to the number of control points, here: about 100, in the B-spline representation (27), which is much smaller. This results in a significant reduction of computational time when generating an appropriate grid where, in addition, we still have control of desirable grid qualities.

Test case T2

This configuration is similar to the previous one (Figure 9). Due to the concave shape the grid lines arising close to the local maximum are known to diverge in the context of orthogonal grid generators. This is opposite to (T1) where they are attracting close to the local minimum of the boundary curve. By means of this configuration we would like to demonstrate the influence of the number of control points M_c and N_v we use in the B-spline representation of the cosine functions c and c_I respectively. For this purpose we perform three computations starting from the initial blend and the corresponding cosine function c_I (Figures 9 and 10). We choose $M_c = N_c = 5$, 10, 20 for the B-spline representation of the cosine function c where we modify the control points of c_I according to (26). The resulting grid functions are plotted in Figures 11–13. We notice that with increasing number of control points for c the effect of contraction and divergence of grid lines close to the local extrema is diminished. This can be explained by the fact that we always put four layers of control points to zero controlling orthogonality close to the bottom boundary. The more control points we use, i.e. the larger M_c and N_c are, the closer these layers are located to the boundary and hence orthogonality is enforced only there. Reversely, the smaller this number becomes the longer the grid lines are orthogonal in the interior of the domain causing the forementioned effects.

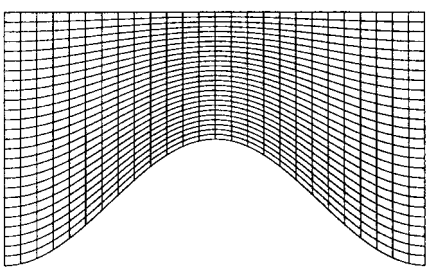

Figure 9. T2: Initial blend.

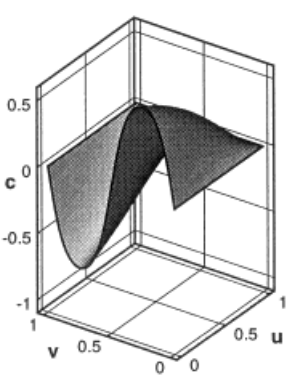

Figure 10. T2: c_I for initial blend.

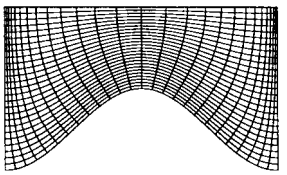

Figure 11. T2: Grid function $M_c = 5$, $N_c = 5$.

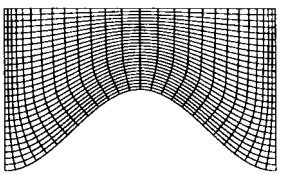

Figure 12. T2: Grid function $M_c = 10$, $N_c = 10$.

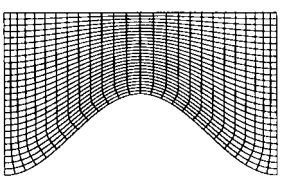

Figure 13. T2: Grid function $M_c = 20$, $N_c = 20$.

Test case T3

Up to now, we only considered configurations where boundary curves are perpendicular to each other at the four corners. Next, we verify that we also can handle configurations where this is violated. In general, orthogonal grid generators produce grid-foldings near to these kind of corners or even leave the domain. Since the initial grid (Figure 14) is determined by a linear blend, it is not orthogonal at *all* of the four boundaries. In particular, there exist geometry-induced singularities at the four corners, where orthogonality is violated at most as can be deduced from Figure 15.

Once again, we consider the influence of the control points used in the B-spline representation of the cosine function. To this end, we perform six different computations varying M_c and N_c respectively (Figures 16–21). We notice that with increasing M_c, i.e. the number in the u-direction, we deviate sooner from orthogonality in this parameter direction. Otherwise, grid lines are attracted to the left boundary, if we increase the number of control points N_c in the other direction.

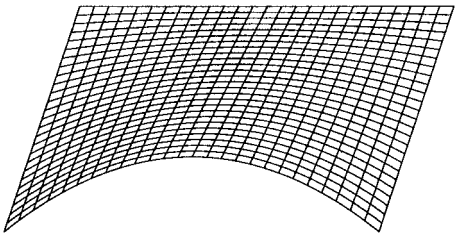

Figure 14. T3: Initial blend.

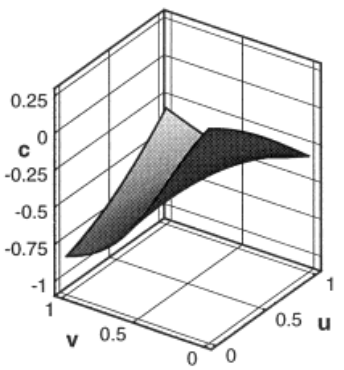

Figure 15. T3: c_I for initial blend.

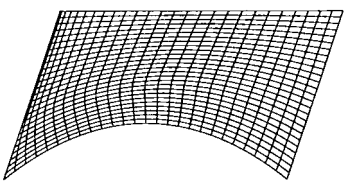

Figure 16. T3: Grid function $M_c = 10$, $N_c = 5$.

Test case T4

Another kind of singularity is characterized by domains where there exist corner angles larger than 180°. For this purpose, we consider the fourfold domain. We determine an initial grid by the Coons patch using a *bi-cubic* blend with vanishing cross-derivative at the four corners. We emphasize that only due to a judicious parameterization of the boundary curves no grid-foldings occur. In general, this is only guaranteed as long as the interior angles at the corners are less than 180°. As can be deduced from Figures 22 and 23, the cubic blend yields a

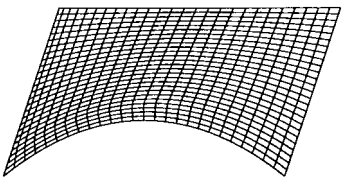

Figure 17. T3: Grid function $M_c = 20$, $N_c = 5$.

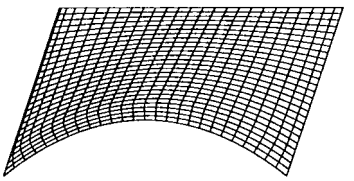

Figure 18. T3: Grid function $M_c = 20$, $N_c = 10$.

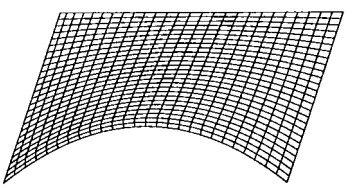

Figure 19. T3: Grid function $M_c = 40$, $N_c = 5$.

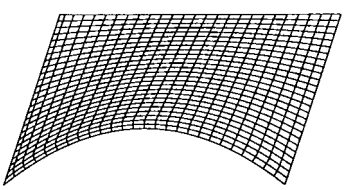

Figure 20. T3: Grid function $M_c = 40$, $N_c = 10$.

non-orthogonal grid where the strongest deviations from orthogonality occur close to the corners. Therefore, the final grid mapping only exhibits two orthogonal boundaries. In addition, we notice that the symmetry inherent in the linear blend is no longer preserved (Figure 24). Furthermore, we would like to mention that we apply a cubic parameter distribution for the plot of the grid function. Here the cubic polynomial is uniquely determined

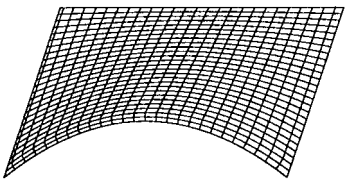

Figure 21. T3: Grid function $M_c = 40$, $N_c = 20$.

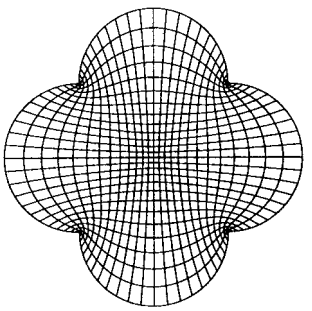

Figure 22. T4: Initial blend.

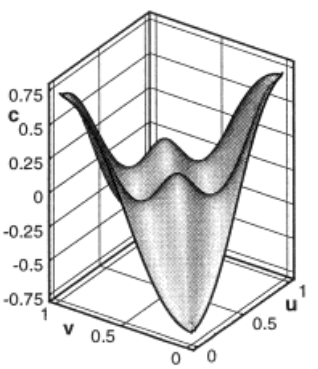

Figure 23. T4: c_I for initial blend.

by the interpolation conditions $u(0) = 0$, $u(1) = 1$ and the slopes $u^1(0) = 0$, $u^1(1) = 0.05$. We notice that the number of control points for the cosine function is rather high in comparison with the other presented configurations. This is due to the resolution close to the singularities at the corners. This number can be significantly reduced when modifying the cosine function by multiplying with a factor of 1.25, which can be interpreted as an overrelaxation by which

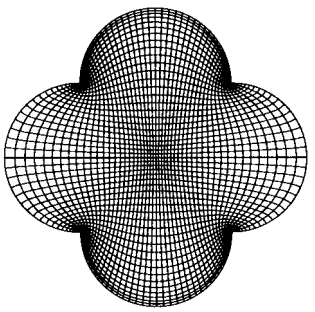

Figure 24. T4: Grid function $M_c = 45$, $N_c = 45$.

the co-ordinate lines are shifted closer to the boundaries. Then only 15 control points have to be put to zero according to (26). The resulting grid function is presented in Figure 25. This again demonstrates the essential influence of the cosine function on the grid function.

Test case T5

A typical application of orthogonal grid generators are external flow fields arising, for instance, around airfoils. To this end, we consider the symmetric NACA-0015, which is frequently applied. Notice that by the geometry a singularity is introduced at the trailing edge where the angle is not 90°. The initial blend is shown in Figure 26. The plot of the corresponding cosine function c_I indicates that we have orthogonality only at the boundary Γ_{u1} (Figure 28). Furthermore, we notice that at the trailing edge $\mathbf{x}_{00}$ orthogonality is significantly violated.

Finally, we obtain the grid function (Figure 27), where we use $M_c = 40$, $N_c = 10$ control points for the cosine function. We emphasize that orthogonality is only realized at the surface

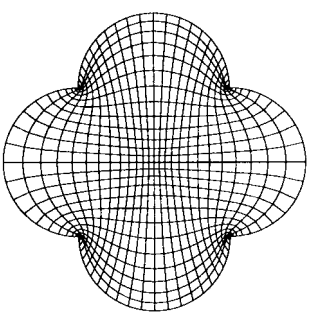

Figure 25. T4: Grid function $M_c = 15$, $N_c = 15$.

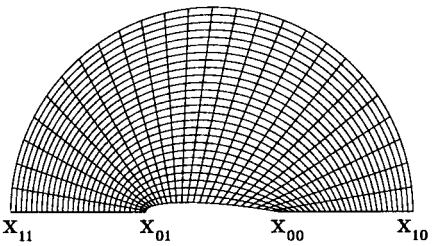

Figure 26. T5: Initial blend.

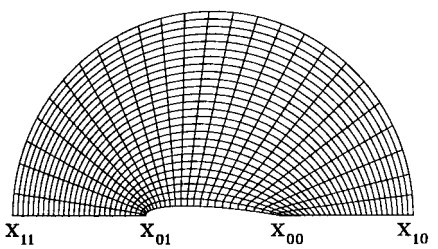

Figure 27. T5: Grid function $M_c = 40$, $N_c = 10$.

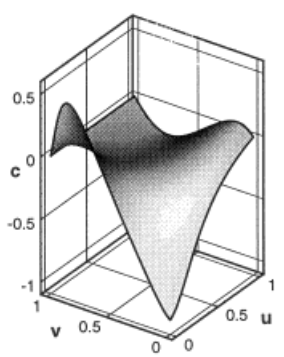

Figure 28. T5: c_I for initial blend.

of the profile, but not at the boundary Γ_{u0}. There, the angles are determined by the initial linear blend. In order to improve orthogonality at this edge, we have to use a different initial grid. For instance, we can use a cubic blend as used for T4 or we apply our grid generator once again where we determine the modified trajectories with respect to the other parameter direction starting from the grid function plotted in Figure 27.

Test case T6

Here we consider an example where the quadrilateral shape of the computational domain is degenerated, i.e. two neighboring corners coincide. By our strategy we are able to handle also these kind of singularities. In order to avoid modifications in our implementation we introduce an infinitesimal perturbation such that the corner points are only *almost* coinciding. Due to the singularity there occurs a significant deviation from orthogonality at the degenerated corners as can be deduced from the cosine function plotted in Figure 31 corresponding to the initial blend Figure 29. The resulting grid mapping determined by a cosine function with $M_c = N_c = 9$ control points is shown in Figure 30.

5.2. Surfaces

Up to now, the investigations have been deliberately restricted to planar two-dimensional configurations. However, we can directly apply the concept to curved surfaces in three-dimensional cases, i.e. $\mathbf{x}$: $[0, 1]^2 \rightarrow \Omega \subset \mathbb{R}^3$, where we choose $\Omega \subset \mathbb{R}^3$ instead of $\Omega \subset \mathbb{R}^2$ in the previous sections. In contrast to the planar case, the domain Ω is no longer specified by the boundary curves alone but, in addition, the interior shape has to be prescribed. In the following we consider one test configuration, referred to as (T7), where the surface is given by a Bezier patch

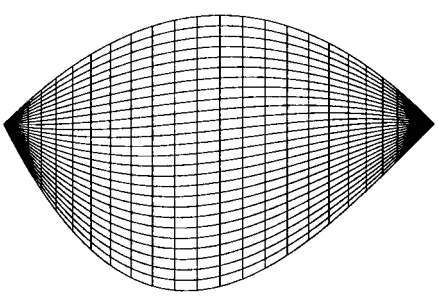

Figure 29. T6: Initial blend.

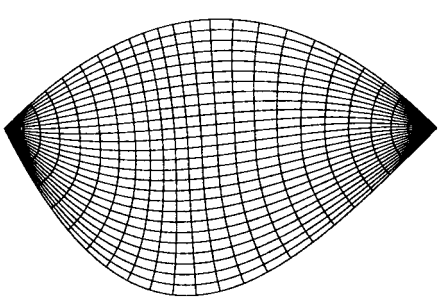

Figure 30. T6: Grid function $M_c = N_c = 9$.

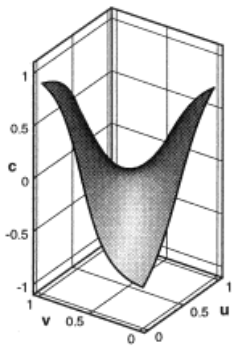

Figure 31. T6: c_I for initial blend.

$$\mathbf{x}(u, v) := \sum_{i=0}^{3} \sum_{j=0}^{3} \mathbf{P}_{ij} \binom{3}{j} (1-v)^{3-j} v^j \binom{3}{i} (1-u)^{3-i} u^i$$

which is uniquely determined by the control points $\mathbf{P}_{ij} \in \mathbb{R}^3$. For each component, it can be more compactly written in matrix–vector representation

$$x_i(u, v) = \mathbf{b}^T(u) \mathbf{P}_j \mathbf{b}(v), \quad i = 1, 2, 3$$

with the vector of Bernstein polynomials $\mathbf{b}(w) = ((1-w)^3,\ 3(1-w)^2 w,\ 3(1-w)w^2,\ w^3)^T$. Here we fix the matrices $\mathbf{P}_1$ and $\mathbf{P}_2$ corresponding to the control points by

$$\mathbf{P}_1 = \mathbf{P}_2 = \frac{1}{9} \begin{pmatrix} 9 & 3 & -3 & -9 \\ 3 & 1 & -1 & -3 \\ -3 & -1 & 1 & 3 \\ -9 & -3 & 3 & 9 \end{pmatrix}$$

and only vary the values in the third component

$$\text{(a)} \quad \mathbf{P}_3 = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 2 & 2 & 1 \\ 0 & 2 & 2 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}, \qquad \text{(b)} \quad \mathbf{P}_3 = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

$$\text{(c)} \quad \mathbf{P}_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

For all of these three settings we present the initial Bezier patch (Figures 32, 36 and 40), and the corresponding cosine function (Figures 33, 37 and 41). Again, the modified cosine function c_I is an interpolatory B-spline function according to step 3 in Section 4 where we perform (26) with $\mathrm{S} = (\{0, \ldots, 4\} \cup \{M_c - 3, \ldots, M_c\}) \times \{0, \ldots, N_c\}$. For each of the three configurations

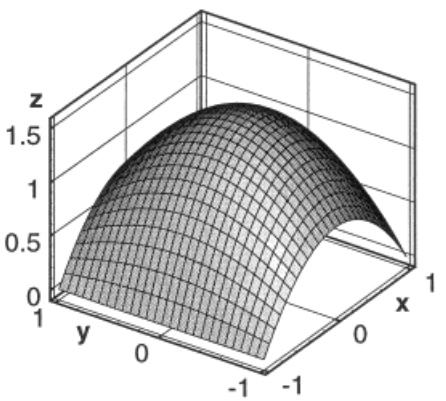

Figure 32. T7a: Initial Bezier patch.

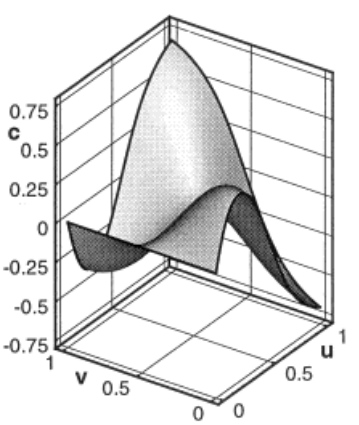

Figure 33. T7a: c_I for initial patch.

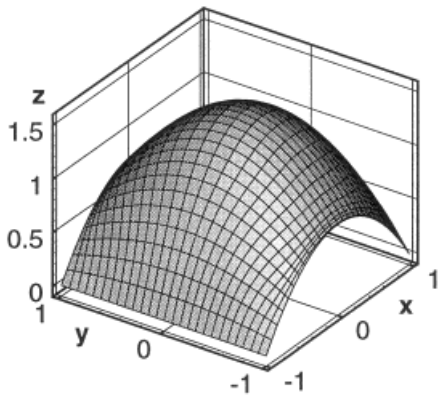

Figure 34. T7a: Grid function $M_c = 10$, $N_c = 10$

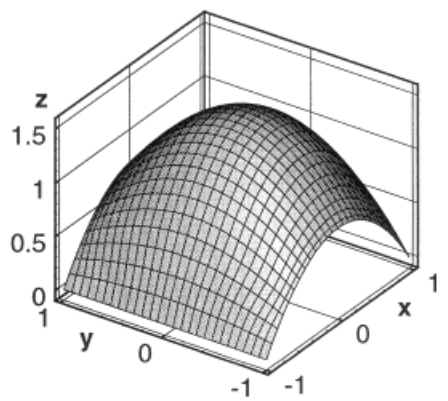

Figure 35. T7a: Grid function $M_c = 20$, $N_c = 10$.

we present two grids with 27×27 points computed from the resulting grid function, which depends on the number of control points for the cosine function. In particular, we use $M_c = 10$, $N_c = 10$ and $M_c = 20$, $N_c = 10$ respectively. The corresponding grids are shown in Figures 34, 38 and 42 and Figures 35, 39 and 43 respectively. The influence on the grid is similar to the planar case.

6. CONCLUSION AND OUTLOOK

An efficient tool is presented by which a curvilinear grid can be computed by evaluating functions from a sparse B-spline representation. The construction principle is basically derived

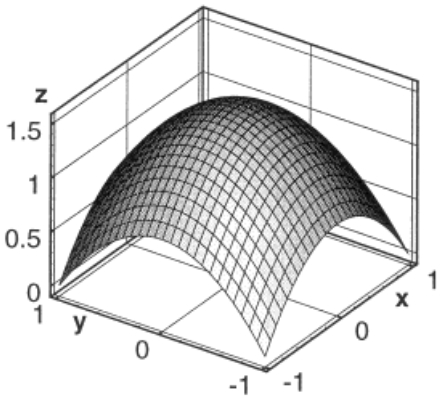

Figure 36. T7b: Initial Bezier patch.

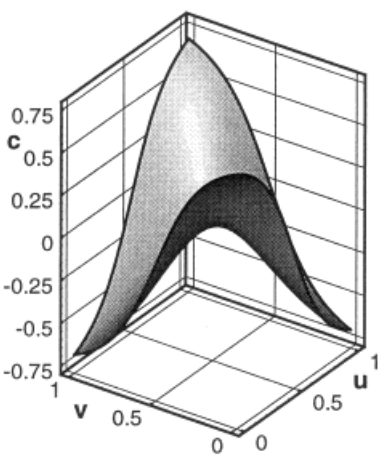

Figure 37. T7b: c_I for initial patch.

from a combination of algebraic and differential grid generation techniques. Here, the core ingredient is a function by which the angle of intersection between two alternate co-ordinate lines can be specified by the user. As first investigations have shown, the user needs local influence onto the setting of these angles in view of local control of grid properties. Here, B-splines turned out to be a powerful tool. However, there is no general strategy available up to now how to compute the control points automatically.

We emphasize that the efficiency of the presented grid generator is essentially based on three features, namely (i) the *fast computation*, (ii) the *sparse representation*, and (iii) the *fast evaluation* of the grid function. In particular, the number of control points by which the grid

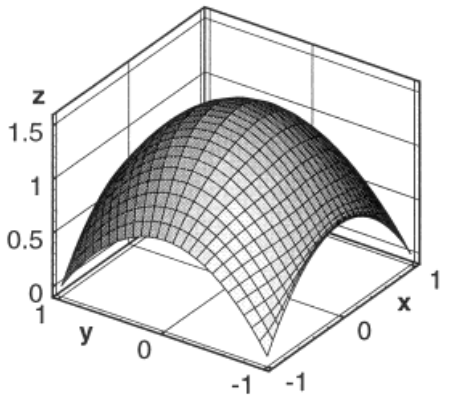

Figure 38. T7b: Grid function $M_c = 10$, $N_c = 10$.

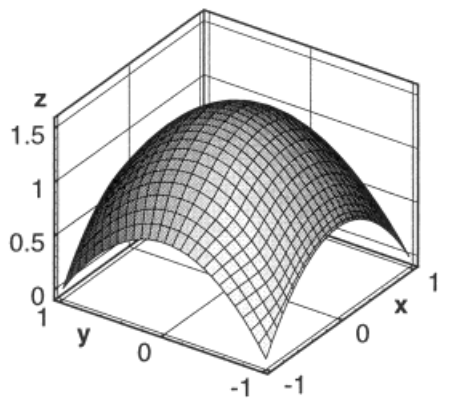

Figure 39. T7b: Grid function $M_c = 20$, $N_c = 10$.

function is determined is much smaller than the number of grid points for the computational grid. This becomes extremely significant when the flow solver at hand is based on local refining and recoarsening strategies. Furthermore, the computation of the grid function only needs a few CPU seconds (here, 1–4) on a 233 MHz PC. This is very promising in view of *time-dependent* grids that have to be modified, for instance, due to changes in the flow field or boundary movements caused by the interaction of aerodynamic and aeroelastic effects. Here, grid generators based on the solution of PDEs are prohibited due to the CPU requirements.

The present investigations have only been performed for two-dimensional domains in the plane and on surfaces. Nevertheless, we believe that the concept can also be extended to three-dimensional domains, i.e. $\mathbf{x}$: $[0, 1]^3 \to \Omega \subset \mathbb{R}^3$. Performing a dimensional splitting, i.e. we can apply the two-dimensional concept to each curvilinear co-ordinate surface separately, a

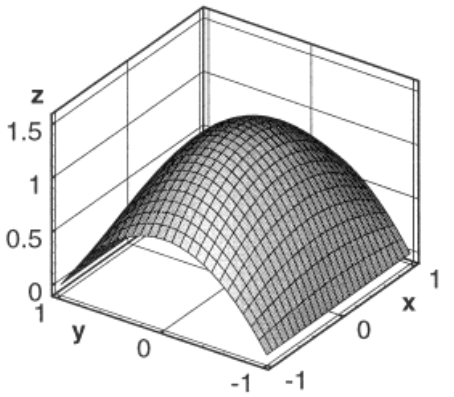

Figure 40. T7c: Initial Bezier patch.

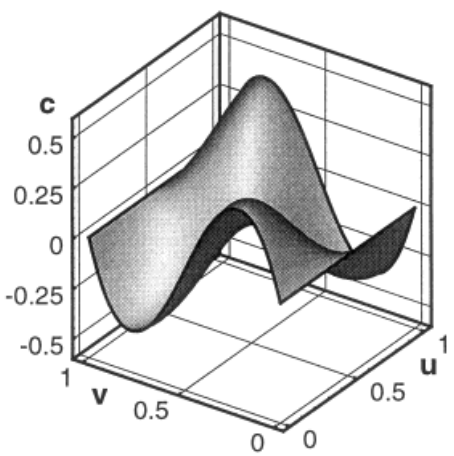

Figure 41. T7c: c_I for initial patch.

straightforward extension is possible. However, by this approach we are only able to control the angle of intersection in one direction but not in two directions simultaneously. Here, more sophisticated strategies have to be developed. In the case of an *orthogonal* grid generator, the resulting grids will be of poor quality for lack of smoothness. Once more, we emphasize that in the present concept the orthogonality condition is locally weakened, i.e. alternate co-ordinate lines are not necessarily perpendicular to each other. As we have verified by configuration (T1), in particular, the smoothness of the resulting grid function can be significantly improved by locally changing a few control points of the B-spline representation corresponding to the function of intersection angles determined by the initial grid mapping. In all of our test configurations the effect of non-smoothness can be avoided if we judiciously choose the cosine

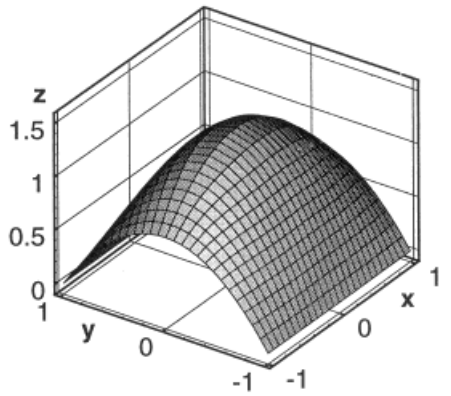

Figure 42. T7c: Grid function $M_c = 10$, $N_c = 10$.

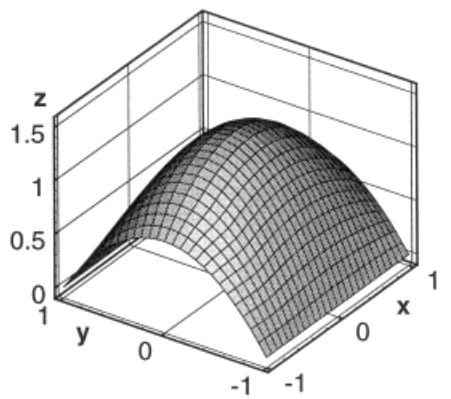

Figure 43. T7c: Grid function $M_c = 20$, $N_c = 10$.

function. Due to the *local* strategy we believe that we will be able to compute grids of reasonable quality even in the case of three-dimensional domains.

Finally, we would like to mention that in a forthcoming work we embed the presented strategy in a block structured concept based on B-spline representation of grid functions in each block separately and apply it to flow field simulations around airfoils.

ACKNOWLEDGMENTS

The authors would like to thank Professor Dr W. Dahmen for many encouraging discussions during the preparation of the present work, especially on conceptual aspects of grid generation systems and their impact on flow solvers. This work has been partially supported by the Deutsche Forschungsgemeinschaft in the SFB 401.

REFERENCES

1. Thompson JF, Warsi ZUA, Mastin CW. *Numerical Grid Generation: Foundations and Applications*. Elsevier/North Holland: Amsterdam, 1985.
2. Coons SA. Surface patches and B-spline curves. In *Computer Aided Geometric Design*, Barnhill RE, Riesenfeld RF (eds). Academic Press: New York, 1974; 1–16.
3. Gordon WJ, Hall ChA. Construction of curvilinear co-ordinate systems and applications to mesh generation. *International Journal for Numerical Methods in Engineering* 1973; **7**: 461–477.
4. Soni BK. Two- and three-dimensional grid generation for internal flow applications of computational fluid dynamics. In AIAA 7th Computational Fluid Dynamics Conference, AIAA Paper No. 855-1526, 1985; 351–359.
5. Eiseman PR. Coordinate generation with precise control over mesh properties. *Journal of Computational Physics* 1982; **47**: 331–351.
6. Eiseman PR. High level continuity for co-ordinate generation with precise controls. *Journal of Computational Physics* 1982; **47**: 352–374.
7. Eiseman PR, Lu N, Jiang M, Thompson JF. Algebraic–elliptic grid generation. In *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, Weatherill NP, Eiseman PR, Hauser J, Thompson JF (eds). Pineridge Press: Swansea, 1994; 37–48.
8. Steger JL, Sorenson RL. Automatic mesh-point clustering near a boundary in grid generation with elliptic partial differential equations. *Journal of Computational Physics* 1979; **33**: 405–410.
9. Thompson JF, Warsi ZUA, Mastin CW. Boundary-fitted co-ordinate systems for numerical solution of partial differential equations—a review. *Journal of Computational Physics* 1982; **47**: 1–108.
10. Eiseman PR. Orthogonal grid generation. *Journal of Applied Mathematics and Computations* 1982; **47**: 193–233.
11. Watford RW. A method for orthogonalizing meshes in plane regions and its application to discrete fluid mechanics problem. Report FS/78/43. Imperial College of Science and Technology, London, 1978.
12. Oh HJ, Kang IS. A non-iterative scheme for orthogonal grid generation with control function and specified boundary correspondence on three sides. *Journal of Computational Physics* 1994; **112**: 138–148.
13. Kang IS, Leal LG. Orthogonal grid generation in a 2D domain via the boundary integral technique. *Journal of Computational Physics* 1992; **102**: 78–87.
14. Duraiswami R, Prosperetti A. Orthogonal mapping in two dimensions. *Journal of Computational Physics* 1992; **98**: 254–268.
15. Gottschlich-Müller B. On multiscale concepts for multidimensional conservation laws. PhD thesis, RWTH Aachen, 1998.
16. Gottschlich-Müller B, Müller S. Adaptive finite volume schemes for conservation laws based on local multiresolution techniques. In *Hyperbolic Problems: Theory*, *Numerics*, *Applications*, Fey M, Jeltsch R (eds). Birkhauser: Basel (Proceedings of 7th International Conference on Hyperbolic Problems, Zurich, 9–13 February 1998), 1999.
17. Henrici P. *Discrete Variable Methods in Ordinary Differential Equations*. Wiley: New York, 1962.
18. Stoer J, Bulirsch R. *Introduction to Numerical Analysis*. Springer: New York, 1980.
19. Courant R, Hilbert D. *Methoden der Mathematischen Physik II* (2nd edn). Springer: Berlin, 1968.
20. Steger JL, Chaussee DS. Generation of body-fitted coordinates using hyperbolic partial differential equations. *SIAM Journal of Scientific Computing* 1980; **1**(4): 431–437.
21. De Boor CA. *A Practical Guide to Splines*. Springer: New York, 1978.
22. Bohm W, Farin G, Kahmann J. A survey of curve and surface methods in CAGD. *Computer Aided Geometric Design* 1984; **1**: 1–60.

