Algebraic–Hyperbolic Grid Generation with Precise Control of Intersection of Angles *

Karl–Heinz Brakhage, Siegfried Müller

Abstract

A new approach to construct high quality meshes for flow calculations is investigated in the context of algebraic grid generation. In particular, the angle of intersection between two alternate coordinate lines can be specified by the user. The construction principle is similar to well–known techniques which are used to transform a non–orthogonal grid into an orthogonal grid. From an initial grid, new coordinate lines in **one** parameter direction are determined by solving a set of ODEs. These trajectories intersect the alternate coordinate lines of the initial grid in a prescribed angle. Finally, the points of intersection are interpolated by a (planar) 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 few 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.

1 Introduction

Algebraic grid generation is a very efficient methodology by which a curvilinear grid can be computed (see e.g. [TZM85]). However, there occur severe drawbacks when applied to the discretization of PDEs 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 [Coo74] are computed from boundary–dependent properties, e.g. point distribution, derivatives, curvature, etc., and appropriate blends [GH73] by which the grid properties in the interior are globally determined. Alternatively, algebraic systems based on transfinite interpolation (see e.g. [Son85, TZM85]) where the interpolation functions can be interpreted as nonlinear 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 nonlinear Poisson equations. Nevertheless, there is one useful feature inherent in the concept

*The work has been partially supported by the Deutsche Forschungsgemeinschaft in the SFB 401.

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 (see [Eis82a, Eis82b]).

Recently, Eiseman et al. [ELJT94] presented the concept of algebraic–elliptic grid generation where the control net is computed by an elliptic grid generator. The combination of these control point curves and transfinite interpolation where the blending functions of Eiseman's control point form are local interpolation functions (see [Eis82a, Eis82b]) 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 nonlinear elliptic PDE of Poisson type to be solved. These are determined from the boundary point distribution (see e.g. [SS79, TZM85]). 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 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. [TZM82, TZM85, Eis82c]). 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 [TZM85], p. 331 ff), this demand restricts the application to 2D problems. In 3D orthogonality is difficult to achieve except for domains which are bounded by boundary surfaces satisfying certain conditions. Moreover, demanding orthogonality in the neighborhood of convex–shaped boundaries leads to the attraction of coordinate 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 our concept, we combine the demands of orthogonality and smoothness. To this end, we weaken overall orthogonality by the condition that two alternate coordinate 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 coordinate 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 (see e.g. [ELJT94]).

Thus, 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 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. At the end, we demonstrate the flexibility of our concept by some test configurations.

Here, we restrict out investigations to 2D grid generation. However, we want to emphasize that the underlying concept carries over in principle to 3D. For more details on this topic we refer the reader to the end of this work.

2 Construction Principle

We consider a domain $\Omega \subset \mathrm{IR}^2$ 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 e.g. [Coo74]). For more details see also Sec. 4. In general the coordinate 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 now will present the basic principle 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

$$\left(\mathbf{x}_u(u, f(u)) + \mathbf{x}_v(u, f(u))f'(u)\right)\mathbf{x}_v(u, f(u)) = 0. \qquad (1)$$

This is an explicit equation for f' which 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 initial value problem (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 ordinary differential equation (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 [Wat78], [TZM82], p. 29).

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 Sec. 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

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

which, in contrast to (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 $g_{11} g_{22} - g_{12}^2$ 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 coordinate lines $\mathbf{y}$ 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 $\mathbf{x} : [0,1]^2 \to \Omega \subset \mathrm{I\!R}^2$ 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 $J(u,v) := (\mathbf{x}_u(u,v) \,, \mathbf{x}_v(u,v))$, $u, v \in [0,1]$, is regular,

and

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

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

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

These assumptions guarantee that the function $G : [0,1]^2 \to \mathrm{I\!R}$ 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. [Hen62]) which guarantee that the solution f_v of the initial value problem (IVP)

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

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..

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

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.,

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

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.,

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

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

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

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

From the previous observations (5, 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 has also to be 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:

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

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. [SB80], p. 408 ff). 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

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

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

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

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, 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

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

Now, we are able to verify that two alternate coordinate 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 straight–forward calculation yields

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

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

$$\begin{aligned} &\mathbf{w}_u(u,v) + G(u,v)\mathbf{w}_v(u,v) = \mathbf{x}_u(u,v) + \mathbf{x}_v(u,v)\,G(u,v), \quad u,v \in [0,1] \\ &\mathbf{w}(0,v) = \mathbf{x}(0,v), \quad v \in [0,1]. \end{aligned} \tag{15}$$

The solution $\mathbf{w}$ of this inhomogeneous advection equation can be constructed according to the standard theory of characteristics (see e.g. [CH68], Chap. 2). Here, the characteristics are defined by the IVP

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

Obviously, the standard results can not 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 value $v_0 \in [0,1]$. 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 (15) can be uniquely determined along the characteristic from the solution of the IVP

$$\begin{aligned} \frac{d\,\mathbf{w}(u, f_{v_0}(u))}{d\,u} &= \mathbf{x}_u(u, f_{v_0}(u)) + \mathbf{x}_v(u, f_{v_0}(u))\,G(u, f_{v_0}(u)) = \tilde{\mathbf{x}}_u(u, v_0), \\ \mathbf{w}(0, f_{v_0}(0)) &= \mathbf{w}(0, v_0) = \mathbf{x}(0, v_0) = \tilde{\mathbf{x}}(0, v_0). \end{aligned}$$

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

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

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 which 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., (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 [Wat78]. However, the derivation is performed in a different and more complicated way. For a summary see [TZM82], p. 29 ff. In the end of Sec. 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 Chaussee and Steger [SC80] which is based on two equations

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

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

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

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 e.g. [TZM82])

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

Here, $\overline{\mathbf{x}}_v^{\perp} := (\overline{y}_v, -\overline{x}_v)^T / \|\overline{\mathbf{x}}_v\|$ denotes one of the two orthonormal vectors to $\overline{\mathbf{x}}_v$. 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 coordinates, 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 which are now described in detail.

Step 1: 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

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

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

$$\begin{array}{lllllllll} \mathbf{x}_{00} & := & \mathbf{x}_{u0}(0) & = & \mathbf{x}_{0v}(0), & \mathbf{x}_{10} & := & \mathbf{x}_{u0}(1) & = & \mathbf{x}_{1v}(0), \\ \mathbf{x}_{01} & := & \mathbf{x}_{u1}(0) & = & \mathbf{x}_{0v}(1), & \mathbf{x}_{11} & := & \mathbf{x}_{u1}(1) & = & \mathbf{x}_{1v}(1). \end{array}$$

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:

$$\begin{array}{rcll} \mathbf{x}(u,0) & = & \mathbf{x}_{u0}(u),\ \mathbf{x}(u,1) = \mathbf{x}_{u1}(u), & u \in [0,1] \\ \mathbf{x}(0,v) & = & \mathbf{x}_{0v}(u),\ \mathbf{x}(1,v) = \mathbf{x}_{1v}(u), & v \in [0,1]. \end{array}$$

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

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

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

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 coordinate 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 fulfilled 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 as two–dimensional B–spline (surface)

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

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

$$\begin{array}{rcl} T_c & = & (0 = t_0 = \ldots = t_{k-1} < \ldots < t_{M_c+k-1} = \ldots = t_{M_c+2(k-1)} = 1), \\ S_c & = & (0 = s_0 = \ldots = s_{k-1} < \ldots < s_{N_c+k-1} = \ldots = s_{N_c+2(k-1)} = 1) \end{array} \tag{20}$$

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 [Boo78] and [BFK84], respectively.

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

$$0 = u_0 < \ldots < u_{M_x} = 1, \quad 0 = v_0 < \ldots < v_{N_x} = 1$$

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 $\tilde{\mathbf{x}}_{i,j},\ i = 0, \ldots, M_x,\ j = 0, \ldots, N_x$, where the coordinate 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 $\overline{\mathbf{x}}$ of the function $\tilde{\mathbf{x}}$ defined by (9). Specifically, we suppose that the final grid function $\overline{\mathbf{x}}$ can be written as

$$\overline{\mathbf{x}}(u, v) = \sum_{i=0}^{M_x+k-2} \sum_{j=0}^{N_x+k-2} \mathbf{P}_{ij} N_i^k(u) N_j^k(v) \tag{21}$$

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

$$\overline{\mathbf{x}}(u_i, v_j) = \tilde{\mathbf{x}}_{i,j}, \quad i = 0, \ldots, M_x,\ j = 0, \ldots, N_x \tag{22}$$

and the interpolatory conditions for the derivatives at the boundaries

$$\begin{aligned} \overline{\mathbf{x}}_u(u_l, v_j) &= \tilde{\mathbf{x}}_u(u_l, v_j), \quad l = 0, M_x,\ j = 0, \ldots, N_x, \\ \overline{\mathbf{x}}_v(u_i, v_l) &= \tilde{\mathbf{x}}_v(u_i, v_l), \quad l = 0, N_x,\ i = 0, \ldots, M_x \end{aligned} \tag{23}$$

as well as the vanishing–twist conditions

$$\overline{\mathbf{x}}_{uv}(u_0, v_0) = \overline{\mathbf{x}}_{uv}(u_0, v_{N_x}) = \overline{\mathbf{x}}_{uv}(u_{M_x}, v_0) = \overline{\mathbf{x}}_{uv}(u_{M_x}, v_{N_x}) = \mathbf{0}. \tag{24}$$

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. The conditions (22,23,24) build a linear system of equations for the control points which can be written as (see [Bra])

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

where $\mathbf{A}_u, \mathbf{A}_v$ are tri–diagonal matrices, $\mathbf{P}$ the matrix of searched control points and $\mathbf{X}$ the matrix which is composed of the values to be interpolated. Solving this system requires an effort proportional to $M_x\, N_x$.

5 Numerical Example

As a test configuration we consider a domain bounded by three straight lines and one convex–shaped curve given by the edge points

$$\mathbf{x}_{00} = \begin{pmatrix} 5 \\ 0 \end{pmatrix}, \quad \mathbf{x}_{10} = \begin{pmatrix} 5 \\ 3 \end{pmatrix}, \quad \mathbf{x}_{01} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \mathbf{x}_{11} = \begin{pmatrix} 0 \\ 3 \end{pmatrix}$$

and the four boundary curves

$$\mathbf{x}_{u0} = \mathbf{x}_{00} + u(\mathbf{x}_{10} - \mathbf{x}_{00}),\ \mathbf{x}_{u1} = \mathbf{x}_{01} + u(\mathbf{x}_{11} - \mathbf{x}_{01}),\ \mathbf{x}_{1v} = \mathbf{x}_{10} + u(\mathbf{x}_{11} - \mathbf{x}_{10}),$$

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

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 straight–forward calculation yields the regularity of its Jacobian. In Fig. 1 the computational domain as well as some curves of the initial function $\mathbf{x}$ are plotted. In addition, the corresponding cosine values

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

are presented in Fig. 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.

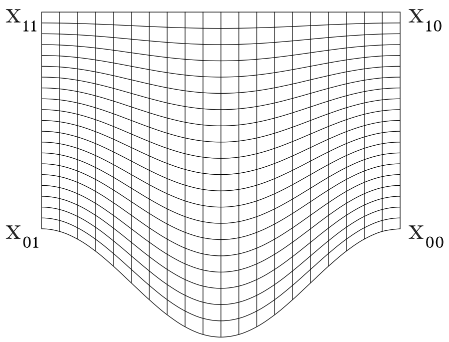

Figure 1: Initial blend

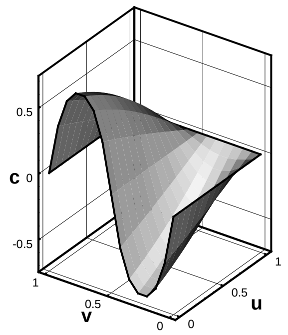

Figure 2: c for initial blend

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 Fig. 4 plotted in the u-$f_v(u)$ plane. As we can learn from Fig. 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., coordinate lines are 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.

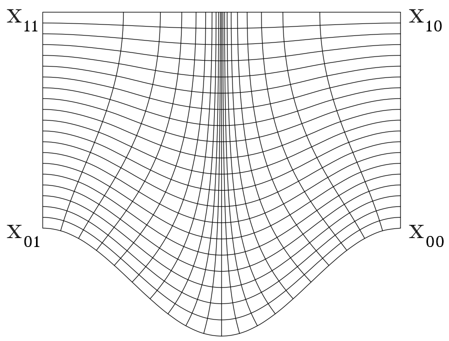

Figure 3: Orthogonal grid function

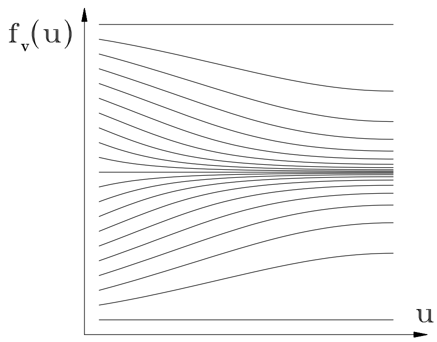

Figure 4: Trajectories to $c \equiv 0$

Hence, we have to proceed differently. Of course, orthogonality at the boundaries is still desirable, however, we have to weaken the orthogonality demand in the interior of the domain in order to avoid attraction of curves. For this purpose, c could be determined by a globally defined analytical function. However, this results in unsatisfactory grids. Therefore, we need a tool by which we can locally control the angles. Here, B–spline surfaces have been proven to be an appropriate tool by which the user has local control on the behavior of the resulting function. Hence, we suppose that the function c can be represented in the form (19). In the present test configuration we determine the control points in two steps. Firstly, we compute an interpolatory B–spline from the interpolatory conditions

$$c(u_i, v_j) = c_I(u_i, v_j), \quad i = 0, \ldots, M_c,\ j = 0, \ldots, N_c \tag{26}$$

and the vanishing–curvature conditions

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

as well as the vanishing–twist condition

$$c_{uv}(u_0, v_0) = c_{uv}(u_0, v_{N_c}) = c_{uv}(u_{M_c}, v_0) = c_{uv}(u_{M_c}, v_{N_c}) = 0 \tag{27}$$

where the parameters $u_i,\ v_j$ are uniformly distributed in [0,1] with $M_c = N_c = 14$. This is the same sparse linear system of equations for the control points Q_{ij} as in (25) with $\mathbf{Q}$ instead of $\mathbf{P}$. Next, we modify the resulting control points such that c vanishes near to the boundary Γ_{0v}. To this end, we put

$$Q_{i,j} = 0, \quad i = 0, 1, 2,\ j = 0, \ldots, N_c.$$

Alternatively, we could have solved (26) with $c(u_i, v_j) = 0,\ i = 0, 1, 2,\ j = 0, \ldots, N_c$. However, this results in an oscillatory behavior of the B–spline.

We would like to emphasize that the presented strategy 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.

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. From the new function c we now compute the trajectories f_v which are presented in Fig. 6. We notice that the trajectories are essentially straight lines with a stronger deviation for small u values, i.e., in comparison to the initial blend the new coordinate lines will only deviate near to the boundary Γ_{0v}. This behavior is reflected in Fig. 5. Moreover, the new function $\tilde{\mathbf{x}}$ is orthogonal close to the boundary Γ_{0v}.

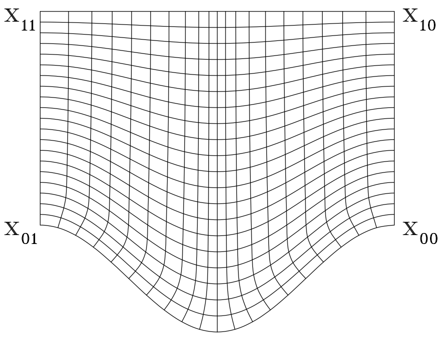

Figure 5: Modified grid function

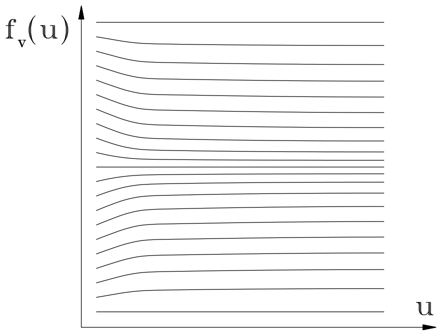

Figure 6: Trajectories to modified c

However, we do not compute $\tilde{\mathbf{x}}$ explicitly but approximate this function by a B–spline $\bar{\mathbf{x}}$ according to (21). 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 3D configurations. In the present test case we choose $M_x = N_x = 10$. The net of control points is presented in Fig. 7. From the function $\overline{\mathbf{x}}$ we are now able to compute the final grid, shown in Fig. 8, just by cheap function evaluations. Here, we choose a grid of Navier–Stokes type, i.e., we attract the grid lines to the boundary Γ_{0v} by introducing a parameter function $\mathbf{s}(u,v) := (u^2, v)^T$. Opposite to standard grid generation systems based on the solutions 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 (21) which is much smaller. This results in an significant reduction of computational time when generating an appropriate grid where, in addition, we still have control of desirable grid qualities.

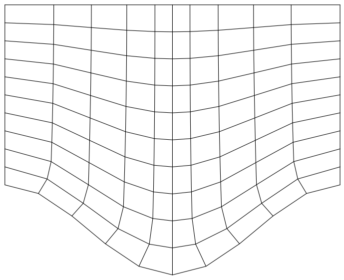

Figure 7: Sparse net of control points

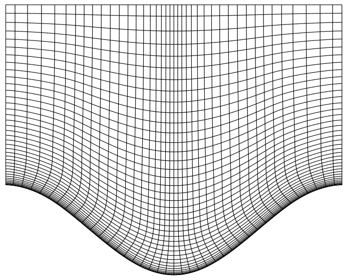

Figure 8: Grid of Navier–Stokes type

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 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 coordinate 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.

The present investigations are restricted to (planar) 2D configurations. However, we can directly apply the concept to curved surfaces in 3D, i.e., $\mathbf{x} : [0,1]^2 \to \Omega \subset \mathrm{IR}^3$, where we have to choose $\Omega \subset \mathrm{IR}^3$ instead of $\Omega \subset \mathrm{IR}^2$ in the previous sections. We can use this when extending the strategy to 3D grid generation, i.e., $\mathbf{x} : [0,1]^3 \to \Omega \subset \mathrm{IR}^3$. Performing a dimensional splitting, i.e., we apply the 2D concept to each curvilinear coordinate surface separately, a straight–forward 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.

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 (see [BM]).

7 Acknowledgment

The authors would like to thank Prof. 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.

References

[BFK84] W. Böhm, G. Farin, and J. Kahmann. A survey of curve and surface methods in CAGD. *Computer Aided Geometric Design*, 1:1–60, 1984.

[BM] K. H. Brakhage and S. Müller. A Grid Generation Concept based on B–Splines and Coons Patches. In preparation.

[Boo78] C.A. De Boor. *A Practical Guide to Splines.* Springer–Verlag, New York, 1978.

[Bra] K. H. Brakhage. B–Splines — An Algorithmic Approach. In preparation.

[CH68] R. Courant and D. Hilbert. *Methoden der Mathematischen Physik II.* Springer, Berlin–Heidelberg–New York, 1968. 2nd edition.

[Coo74] S. A. Coons. Surface patches and b–spline curves. In R. E. Barnhill and R. F. Riesenfeld, editors, *Computer Aided Geometric Design*, pages 1–16. Academic Press, New York, 1974.

[Eis82a] P.R. Eiseman. Coordinate generation with precise control over mesh properties. *J. Comp. Phys.*, 47:331–351, 1982.

[Eis82b] P.R. Eiseman. High level continuity for coordinate generation with precise controls. *J. Comp. Phys.*, 47:352–374, 1982.

[Eis82c] P.R. Eiseman. Orthogonal grid generation. *J. Comp. Appl. Math.*, 47:193–233, 1982.

[ELJT94] P.R. Eiseman, N. Lu, M. Jiang, and J.F. Thompson. Algebraic–elliptic grid generation. In N.P. Weatherill, P.R. Eiseman, J. Häuser, and J.F. Thompson, editors, *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, pages 37–48. Pineridge Press, Swansea, 1994.

[GH73] W. J. Gordon and Ch. A. Hall. Construction of curvilinear co–ordinate systems and applications to mesh generation. *Int. J. for Numerical Methods in Engineering*, 7:461–477, 1973.

[Hen62] P. Henrici. *Discrete Variable Methods in Ordinary Differential Equations.* John–Wiley, New York, 1962.

[SB80] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis.* Springer, New York, 1980.

[SC80] J.L. Steger and D.S. Chaussee. Generation of body–fitted coordinates using hyperbolic partial differential equations. *SIAM J. Sci. Comput.*, 1(4), 1980.

[Son85] B.K. Soni. Two– and three–dimensional grid generation for internal flow applications of computational fluid dynamics. In *AIAA 7th Computational Fluid Dynamics Conference, AIAA paper no. 85–1526*, pages 351–359, 1985.

[SS79] J. L. Steger and R. L. Sorenson. Automatic mesh–point clustering near a boundary in grid generation with elliptic partial differential equations. *J. Comp. Phys.*, 33:405–410, 1979.

[TZM82] J.F. Thompson, Z.U.A.Warsi, and C.W. Mastin. Boundary–fitted coordinate systems for numerical solution of partial differential equations — a review. *J. Comp. Phys.*, 47:1–108, 1982.

[TZM85] J.F. Thompson, Z.U.A.Warsi, and C.W. Mastin. *Numerical Grid Generation: Foundations and Applications.* Elsevier, North Holland, 1985.

[Wat78] R. W. Watford. A Method for Orthogonalizing Meshes in Plane Regions and Its Application to Discrete Fluid Mechanics Problem. Report FS/78/43, Imp. College of Science and Tech., London, 1978.

Karl–Heinz Brakhage
Institut für Geometrie und Praktische Mathematik,
RWTH Aachen, Templergraben 55,
D-52056 Aachen, Germany

E–mail address: brakhage@igpm.rwth-aachen.de

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

E–mail address: mueller@igpm.rwth-aachen.de