A Unified Approach to the Modeling of Airplane Wings and Numerical Grid Generation using B-Spline Representations

Karl-Heinz Brakhage, Wolfgang Dahmen and Philipp Lamby

Abstract In this article we summarize the development of a unified platform for treating the entire range of geometric preprocessing tasks that preceded the wind tunnel readings and the numerical simulations performed in the collaborative research center SFB 401. In particular, this includes the automated generation of the CAD models which were used for manufacturing multi-parted wing-fuselage configurations as well as the generation of the numerical grids for the corresponding, adaptive numerical simulations.

AMS Subject Classifications: 65D07, 65D17, 65D10, 76N15

Key words: B-splines, approximation, fairing, offset curves, grid generation

1 Introduction and Overview

The numerical simulation of fluid-structure-interaction, especially when employing high level models such as the compressible Navier-Stokes equations on the fluid side, still pose enormous challenges that cannot be met solely by increased computing power. The processes are inherently nonstationary and the computational domain varies in time. Moreover, stiff components in the coupled fluid-structure problem require implicit time integration and hence the repeated solution of possible extremely large nonlinear systems of equations. To deal with problems of such complexity calls, on one hand, for adaptive spatial and temporal discretizations in the fluid and structure solvers, in order to keep the systems as small as possible in the first place. On the other hand, such adaptive meshes need to be frequently adapted

Karl-Heinz Brakhage, Wolfgang Dahmen
Institut für Geometrie und Praktische Mathematik, RWTH Aachen,
D-52056 Aachen, Germany, e-mail: brakhage{dahmen}@igpm.rwth-aachen.de
Philipp Lamby
University of South Carolina, Industrial Mathematics Institute,
Columbia SC, 29208, USA, e-mail: lamby@webmail.sc.edu

to varying domain geometries. Therefore, a long term central objective has been the development of an integrated concept that closely intertwines adaptive discretizations with a tailormade grid generation in a way that the above tasks are supported to a possibly large extent. Moreover, manufacturing high quality models for corresponding wind tunnel experiments suggests taking related CAD tasks into account. The project B2 of the SFB 401 has been therefore concerned with the development of suitable grid generation concepts as well as with the generation of CAD models. More precisely, the work in this project can be divided into three major blocks:

1. In order to support adaptive discretizations based on hierarchies of nested grids, as needed by the multiscale algorithms designed in project A4, a *parametric* grid generation concept has been developed and integrated into the finite volume solver QUADFLOW.
2. A system, including a graphical interface, for generating parametric grids employing B-spline techniques has been implemented. In particular, properly adapted versions of many classical grid generation algorithms have been integrated into the B-spline concept as tools for generating suitable control nets for the B-spline mappings. Furthermore new methods, in particular for the generation of offset grids, have been invented.
3. In the context of the HiReNASD-project CAD models for the multi-parted wing-fuselage configuration have been generated, which are both highly parameterizable and high-quality, such that they can be used directly for both the manufacturing process and the grid generation.

The central objective of project B2 has been to develop a unified framework for addressing all the geometric processing tasks related to the above three packages. In particular, geometry representations needed to be found that support both the manufacturing process and the grid generation already at the modeling stage. This goal has been achieved by extensive use of B-spline techniques. In the following three sections we shall recall the motivation behind this concept, sketch briefly the evolution of the project during the whole funding period of the research center and highlight some of the latest developments.

1.1 Parametric Grids

Some basic features of QUADFLOW and corresponding demands on the underlying grid generation concept are indicated in Figure 1 which shows a transonic inviscid fluid flow around a BAC 3-11 profile. The computation starts with an initially very coarse grid. After some cycles of adaptive flow computation one arrives at a final adapted grid meeting a desired target accuracy. The grid adaptation is a quadtree-type *h*-refinement based on an initial block partition of the computational domain. Each block hosts a logically Cartesian grid. Grids on adjacent blocks need not match though which simplifies the process. The key task of the grid generation module is to provide hierarchies of grids at arbitrary levels of resolution that are structured

blockwise, so that locally refined portions of such hierarchies can be efficiently activated by suitable adaptation criteria.

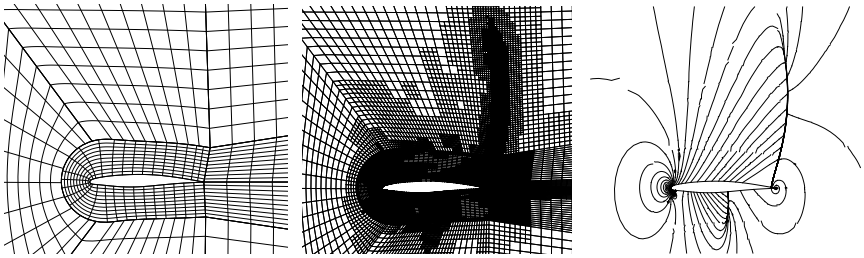

Fig. 1 Initial coarse grid, final adapted grid and pressure distribution for flow around BAC 3-11 profile, Mach 0.85.

The adaptation strategy employed in QUADFLOW requires the underlying grid hierarchy to be nested. This means that a cell $V_{j,k}$ on level j with index k is the union of cells $V_{j+1,r}$, $r \in \mathscr{M}_{j,k}$ on the next finer level so that the sum of the volumes of the fine grid cells is the same as the volume of the coarse grid cell: $\sum_{r\in\mathscr{M}_{j,k}} |V_{j+1,r}| = |V_{j,k}|$. Here a problem arises because obviously in curvilinear domains standard polygonal grid hierarchies are *not* nested.

In order to facilitate the efficient generation of nested grids of curvilinear type we do *not* use discrete grid models. Instead we provide as input for the flow solver parametric mappings that represent coordinate systems in the single blocks. By default these mappings are realized by tensor product B-splines which ensure fast point evaluation independent of the number of knots. Then grids at arbitrary level of resolution can be constructed easily by function evaluation and grid nestedness is automatically achieved if one considers a grid cell to be the geometric image of the corresponding cell in parameter space. Hence, within the frame of our grid generation concept a multi-block grid is a representation of the flow domain by B-spline tensor product patches. In 3D, of course, the blocks are trivariate B-spline tensor product volumes bounded by B-splines surfaces.

The integration of the parametric grid generation concept into the QUADFLOW solver started at the beginning of the second funding period, when the first prototypes of the multiscale adaptation algorithm developed in project A4 and the finite volume solver implemented in project B3 became available. Since then the parametric system has been used and tested extensively. In the course of these investigations it turned out though, that in the curvilinear setting, special care has to be taken in order to fulfill the *geometric conservation laws* which can be considered as necessary consistency conditions for the accuracy and stability of the finite volume solver. For a detailed discussion of this topic we refer to [17].

1.2 B-Spline Grid Generation

Of course, the concepts described above require a possibly automatized and robust generation of B-spline mappings to begin with. In principle this task can be related to classical block-structured grid generation by viewing the B-spline control nets, i.e. the coefficients in B-spline representations, as coarse versions of grids, since these control points indeed have a geometric significance. Hence, in principle, a host of algorithms and codes could be taken from the relevant literature to serve that purpose. The most notable ones are algebraic grid generation techniques using transfinite interpolation and elliptic grid generation systems. Therefore, already in the first funding period, these algorithms were thoroughly evaluated regarding their usability and, if approved, were added to the software package that was build in this project. As mentioned before these algorithms could be used to generate control nets, or to generate first classical grids and convert them subsequently into B-spline representations. This required the development of tailormade fast interpolation and approximation algorithms. A detailed documentation of this stage of the project can be found in [12].

On the other hand, the superior flexibility and accuracy of the B-spline representations motivated the invention of completely new grid generation methods, like for example *algebraic-hyperbolic* grid generation as described in [6], and later the generation of offset blocks, see [10].

Since complex grids cannot be represented by a single tensor-product mapping, a multiblock-grid data structure has to be defined. This became a major part of the work during the middle of the funding period of the research center. Finally a multiblock manager was implemented that mimics ideas from [21] and [19] and adapts them to the parametric setting, see [17].

At a later stage of the QUADFLOW development, when the solver became able to compute on time-varying grids, robust and efficient grid deformation algorithms became an important issue. A first set of such algorithms based on algebraic perturbation methods was presented in [11]. These methods were later expanded by a deformation algorithm based on radial basis functions, which will be discussed later in Section 4.

Only at about this time, i.e. at a relatively late stage of the project, when the configurations and block-decompositions became too complex to be assembled based on sketches and scripts, a real, interactive graphical interface was implemented. In this context the need of revising some of the initial approximation algorithms became apparent since they did not perform well enough in a complex application. In particular, this was the case when for constructive reasons irregular B-spline knot sequences came into play. As a remedy at that stage of the work we chose to discretize an elliptic grid generation system by means of *collocation* instead by finite differences, which naturally ties into the B-spline representations and offers significantly more accuracy. These new results will be presented in Section 3.

1.3 Generation of Wing Models

In the final stage of the SFB another problem came into focus.

Usually the CAD-models which are used for the manufacturing of wind tunnel models are generated by commercial programs that provide hundreds of geometric entities to construct the various geometric features. Every program has its own way to represent and manipulate data. Conversion from one program to the other is accompanied by approximation processes which usually cause errors and loss of information, in particular, with regard to the topology. For instance, in CAD-models one finds very frequently trimmed surfaces, which destroy the logically Cartesian topology of the surface representation and therefore work very badly together with block-structured grids.

Hence, after getting the CAD-data, the grid generator is usually committed to fix and repair the geometry before he can start with the genuine grid generation task. This geometry post-processing can take weeks of work for a complex configuration and is therefore a major bottleneck in the modeling and grid generation pipeline.

In this project CAD-models for an airplane wing have been constructed that do not suffer from such deficiencies. The complete wing is constructed by exactly fitting, untrimmed B-spline patches. The wing itself corresponds to a three parted, back-swept BAC 3-11 aerofoil cruise configuration of scale 1:28 with rounded tip. To diminish wind tunnel influences a half-body is placed between the wing and the wind tunnel wall. Optionally the outer part of the wing can be replaced by a part with a winglet. This model is conveniently parameterizable and offers the choice to vary design parameters like bending radii, angles and top views easily.

However, in order to fulfill several non-standard constraints, that stemmed from the design demands and the manufacturing needs, the algorithms for approximation and fairing, that can be found in literature, had to be extended, as will be explained in Section 2.

Pure B-splines are ideal for communication between different software packages, because they are considered to be basic in CAD and are understood by basically every B-spline software. The basic data exchange between the modeling, grid generation and manufacturing software was carried out by IGES files. Concretely the milling machine employed hyperCAD/hyperMill from OpenMind, the inner technical constructions were planed with CATIA and for the visualization we used Rhino.

1.4 Outline of Paper

In the remainder of this article we mostly concentrate on work, that has been done in the last funding period. First, in Section 2 we present in some detail the construction of the wing model that was used for the HiReNASD wind tunnel readings described somewhere else in this book. Section 3 sets up a B-spline collocation method to realize an elliptic grid generation system. In some sense, this section serves to

demonstrate how classical grid generation theory and B-spline representation work together. Finally, in Section 4 we discuss an algorithm for grid deformation.

1.5 Notation

Throughout this paper we write B-spline curves in the form $\mathbf{x}(t) = \sum_{i=0}^{N} \mathbf{p}_i N_{i,p,T}(t)$ where $N_{i,p,T}(t)$ is the i-th normalized B-spline function of order p (degree $p-1$) corresponding to the generally non-uniform knot vector $T = (t_0, t_1, \ldots, t_{N+p})$ and $\mathbf{p}_i$ are the control points that determine the curve. In fact, since the B-splines form a local partition of unity the location of the control points already conveys a good geometric information on the actual position of the corresponding curve. Moreover, (a properly defined notion of) oscillation of the sequence of control points and of the corresponding control polygon is known to control also the oscillation of the B-spline curve. We usually assume that T is clamped and all interior knots are of multiplicity one, i.e., $t_0 = \ldots = t_{p-1} < t_p < \ldots < t_N < t_{N+1} = \ldots = t_{N+p}$. For the sake of simplicity we write $N_{i,p}$ instead of $N_{i,p,T}$ whenever it is clear from the context which knot vector is being referred to. Surfaces are represented by B-spline tensor products of the form

$$\sum_{i=0}^{N} \sum_{j=0}^{M} \mathbf{p}_{ij} N_{i,p}(u) N_{j,q}(v). \tag{1}$$

We shall always adopt the following convention. If $\mathbf{q}$ is a point, then $x(\mathbf{q}), y(\mathbf{q}), z(\mathbf{q})$ denote its x, y, or z-coordinate respectively. The orientation of our coordinate system is explained in Figures 4 and 5.

2 Geometry Description and Outline of the Modeling Process

We start the technical part of this paper with a description of the construction of the CAD-model for the wing model including the mounting unit shown in Figure 2. Both parts are represented exclusively by untrimmed B-spline patches. More specifically, as pointed out later, the main part of the wing is defined by a set of cross-sections which are connected by ruled surfaces. Hereby the number of cross-sections can be varied on demand. Additionally the simplified half of a fuselage has been designed in order to reduce the influence of the wind tunnel walls, see figure 3. This fuselage is represented by a curvature continuous periodic B-spline surface.

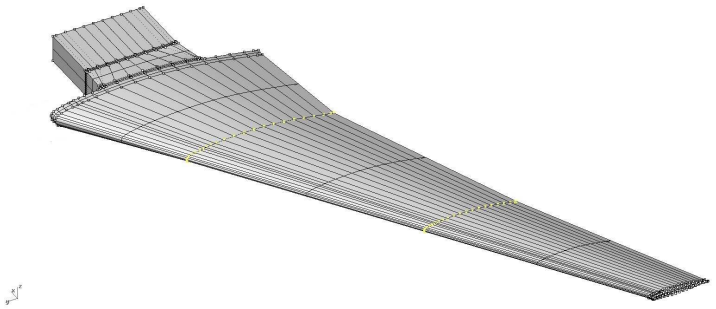

Fig. 2 View of the complete model with mounting unit.

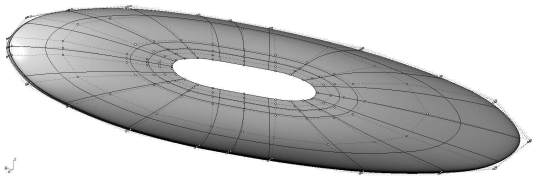

Fig. 3 Simplified fuselage – control points and knot-isolines

2.1 Cross Sections and Wing Construction

The first step of the modeling process is to find B-spline representations of the wing cross sections. Originally the reference cross section of the cruise configuration was described by 87 points given in the two ARGARD reports [18]. These points $\mathbf{x}_j$ are scaled in such a way that the wing depth, i.e, the distance from the nose to the trailing edge, attains unit length. The tolerance relative to the wing depth is $\varepsilon = 1.7 \cdot 10^{-4}$. Thus we have to guarantee that

$$\min_t \|\mathbf{x}(t) - \mathbf{x}_j\|_2 \leq \varepsilon \qquad \forall j.$$

The profile is represented by a closed curve with a discontinuous tangent only at the trailing edge which is located precisely at $(1,0)$. Moreover, the point of the profile corresponding to the nose of the wing is $(0,0)$ and has a vertical tangent there. Optionally the curvature of the cross section at this point can be prescribed as well. Since the commonly used approximation schemes do not allow one to meet such a variety of different approximation constraints, special approximation procedures had to be developed, which are described in [10].

The relative thickness of the reference profile is $rt = 11\%$. This thickness can be varied by scaling the control points of the spline in vertical direction. At the fuselage the thickness of the profile has to be 15%. This increase of thickness has to be achieved by changing the profile only in the lower part (see Figure 4). All these

computations are done in 2D space because they require only varying cross sections. Therefore we can fix the knot vector after computing the smooth 11% reference cross-section and keep the same knot vector for the other profiles, in particular, for the profile at the fuselage. This will later lead to an easy representation of the airfoil sections. The result is shown in Figure 4. We have used order $p = 4$ and a parametrization according to chord length.

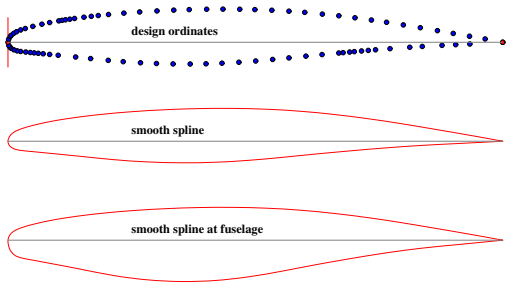

Fig. 4 Design ordinates, spline ($rt = 11\%$) and spline at fuselage ($rt = 15\%$). For later reference: the horizontal direction in this sketch corresponds to the y-coordinate of our 3D-coordinate system. The vertical direction corresponds to the z-coordinate.

The next step is to describe the top view sketch of the multi-parted back-swept wing. This can be done with the aid of an arbitrary 2D CAD program that can treat B-splines and export coordinates and lengths. We use WinCAG [3, 4]. For our purposes some special interfaces were added to that system. The only information we need from this step is the front position of the cross-sections A_i, their depth l_i and the relative position of R with respect to A_n ($= A_4$ here), compare Figure 5).

The scaled profiles are placed at the right position in 3D space. The corresponding 3D coordinates are generated from the 2D sketch. Using piecewise linear connections between the profiles, the multi-parted wing can be represented by tensor product B-spline patches of order $(p \times 2)$ which results in ruled surfaces between the cross sections A_i. The only exception is the lower part of the patch near the fuselage which consists of conics.

The sensors and cables have to be placed inside the wing. The necessary shell thickness of the aerofoil is roughly known from stress and eigenfrequencies computations (FE shell model considering webs) and is of variable size. Therefore a variable inner offset surface of the wing was computed. All detail constructions for the interior equipment have to remain inside this surface.

So far we have considered mainly 2D constructions. In the following sections we shall describe some genuinely 3D constructions.

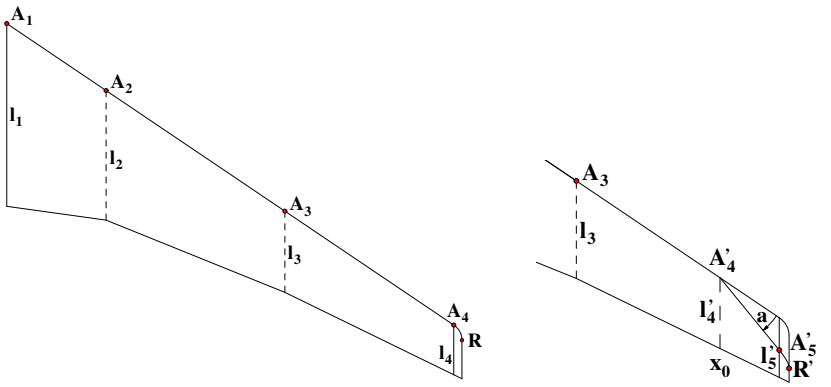

Fig. 5 Sheer Plan of the multi-parted back-swept wing – winglet modifications on right hand side, see Section 2.4. For later reference: The coordinate in span direction (i.e. the horizontal direction in the sketch) is the *x*-coordinate, the vertical direction is described by the *y*-coordinate.

2.2 Mounting Unit

The mounting unit is given by its top and front view plans which also indicate the rounded angles needed for the milling process. Only the top view of the fillet is given in the 2D sketch. To avoid gaps, the blend is not computed as a trimmed surface. For this reason the B-spline representing the cross-section at the fuselage has to be split up into five parts. This is done by knot insertion. The fillet and the mounting unit are then computed as one block. The exact blending radii and their centers are exported separately to the milling software. This part was manufactured with a cylindrical milling cutter. Between the wing and the blend to the mounting unit a cylindrical continuation can be added (see Figure 6) to pass the (material of the) fuselage.

The realization of GC1-continuity (tangent plane continuity) for the fillet is a little bit more difficult. Employing knot insertion we subdivide the profile spline of the wing into 5 parts (compare Figure 6). The partitions number 2 and 4 - the last one with inverted orientation - are the boundary curves of our fillet. Now we can apply the formulas from [15, Ch.7] to achieve the desired continuity properties. For the blending radii a pre-computation of the *x*-*y*-coordinates is done (fitting of the circular arc). We have still one degree of freedom left for the second row of control points. This can be used to design a fairly rapid transition from the wing to the inclined part. The same applies to the transition to the mounting unit. Figure 7 shows the four untrimmed surfaces forming the transition.

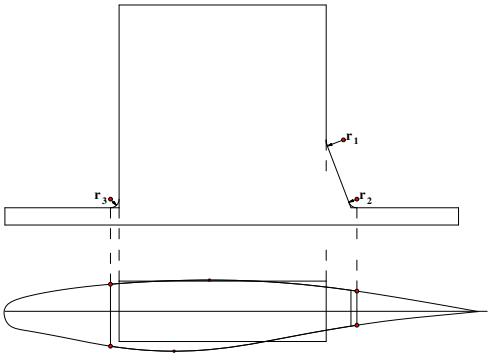

Fig. 6 Top and front view of mounting unit with continuation

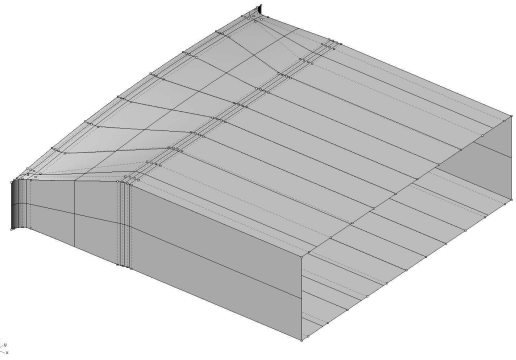

Fig. 7 Plot of the fillet with mounting unit.

2.3 The Wing Tip

The rounded wing tip is determined by the relative position of $\mathbf{R}$ to $\mathbf{A}_n$ (compare Figure 5) and the following construction. The engineers demand only GC1-continuity at the crossover from the wing given by the control points $\mathbf{p}_{ij}$ to the tip with control points $\mathbf{q}_{ik}$. To achieve this with a possibly small number of control points we build a B-spline-Bézier surface of order $p \times 3$. The control point matrix $(\mathbf{q}_{ik})$ will consist of three rows ($k = 0,1,2$) of control points each with $i = 0,\ldots,N$ points. Here $N+1$ is the number of control points in the spline representations of each cross-section.

As first row of control points for the tip we take the one from the last cross section of the airfoil: $\mathbf{q}_{i0} = \mathbf{p}_{iN}$. We place the second row of control points on the intersection of the lines through pairs of two corresponding control points from the last two airfoil cross sections with the plane through the point $\mathbf{R}$ parallel to the cross-section A_4, i.e. $\mathbf{q}_{i1} = \mathbf{p}_{iN} + \alpha\,(\mathbf{p}_{iN} - \mathbf{p}_{i,N-1})$. Since the outer part of the wing

is a conic with parallel cross-sections this construction guarantees the desired GC1-continuity. For the third and last row we first project the points of the second row into the *x*-*y*-plane and then transform the *y*-coordinates of these points linearly onto the interval $y(\mathbf{q}_{i0}), y(\mathbf{R})$ in order to round of the wing at the outer end of the leading edge:

$$\mathbf{q}_{i2} = (x(\mathbf{q}_{i1}), y(\mathbf{q}_{01}) + \frac{y(\mathbf{R}) - y(\mathbf{q}_{01})}{y(\mathbf{P}) - y(\mathbf{q}_{01})}(y(\mathbf{q}_{i1}) - y(\mathbf{q}_{01}), 0),$$

where $\mathbf{P}$ is the nose of the profile defined by the spline with control points $\mathbf{q}_{i1}$. The surface defined by this procedure is a GC1 continuation of the wing and is automatically computed due to the above choice of parameters. In this form it is well suited for manufacturing, but for grid generation it has to be re-parameterized, see [10]. A plot of the wing tip and its control points is given in Figure 8.

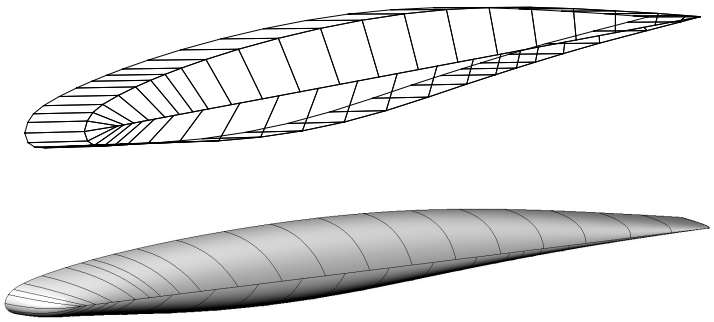

Fig. 8 Wing tip – Control points (top), knot-isolines (bottom).

2.4 Winglet Construction

A future series of wind tunnel readings will be concerned with wing configurations where a winglet is added to the airfoil, see Figure 9. For this reason we have developed algorithms for automatic winglet constructions. In a first step the sheer plan (compare Figure 5) has to be extended. We mark the bending position $\mathbf{x}_0$ and introduce an additional dihedral angle a. From this the new positions $\mathbf{A}'_4$ and $\mathbf{A}'_5$ and the wing chords l'_4 and l'_5 are determined. Finally $\mathbf{R}'$ takes over the role of $\mathbf{R}$ in the construction of the wing tip.

The basic idea of the 3D construction is to do all computations directly on the control points. Remember that the sheer plan lies in the *xy*-plane and the *x*-coordinate corresponds to the span width. In this plane the control points of our surfaces have (x_{ij}, y_{ij})-coordinates and certain heights z_{ij} (positive or negative). In

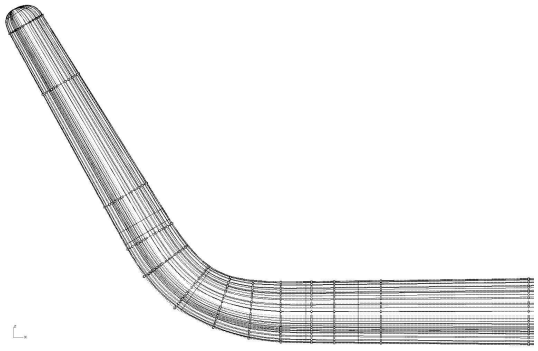

Fig. 9 Side view of winglet construction – bending angle 60°.

the following constructions the y values of the control points will not be changed. Therefore the bending process can be described by the sketch 10 which shows a projection in y-direction. The sketch shows, how the original middle axis of the straight

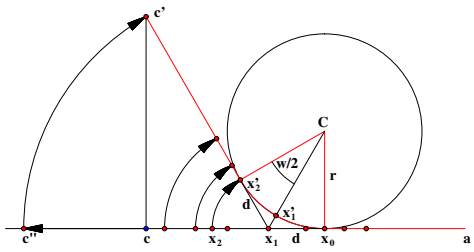

Fig. 10 Principle of winglet construction.

wing, i.e. the plane $y = 0$ containing the leading and trailing edge of the wing, is transformed into a curved middle axis for the wing with winglet. The adjustable parameters of this construction are the bending position $\mathbf{x}_0$, the radius r and the angle w.

Given these quantities, the center $\mathbf{C}$ and the points $\mathbf{x}_0(=\mathbf{x}_0')$, $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_1'$ and $\mathbf{x}_2'$ are determined according to the specifications in the sketch. We want to keep the wing span given by $\mathbf{c}$. Thus we first have to prolong the straight wing from $\mathbf{c}$ to $\mathbf{c}''$. Since the original representation of the straight wing as piecewise linear spline prohibits the generation of a curved structure, we replace it by a spline of order $p$using a degree raising algorithm. Then we determine the spline parameters $v_0 < v_1 < v_2$

corresponding to the points $\mathbf{x}_0$, $\mathbf{x}_1$ and $\mathbf{x}_2$. Starting from $\mathbf{a}$ we want to keep the middle axis straight up to as closely as possible to $\mathbf{x}_0$. Hence we insert some knots ($< v_1$) around v_0. For the same reason we add some knots ($> v_1$) around v_2. This gives us a couple of control points lying between $\mathbf{a}$ and $\mathbf{c}''$ to provide the necessary flexibility for modeling the bending region. The control points on the left of $\mathbf{x}_2$ are mapped by a rotation around center $\mathbf{x}_1$ onto the line $\mathbf{c}'\mathbf{x}_2'$. The control points between $\mathbf{x}_2$ and $\mathbf{x}_0$ are mapped onto the circular arc connecting $\mathbf{x}_2'$ and $\mathbf{x}_0$ according to their distances. The control points on the right of $\mathbf{x}_0$ remain unchanged. Altogether this describes a piecewise defined function $\mathbf{x} = (x,0) \mapsto \mathbf{x}' = (x',z')$. With $\mathbf{n} = (n_x, n_z)$ we can denote the unit normal to the image curve at the point (x',z')

Actually we are not interested primarily in the deformation of the middle axis but rather in the transformation of the surface points. To define this we assume that the cross sections are rigid and remain perpendicular to the middle axis, so that finally any surface control point $\mathbf{p} = (x,y,z)$ is mapped to $(x' + zn_x, y, z' + zn_z)$.

Figure 11 shows the final result with parameters $a = 10^\circ$, $\mathbf{x}_0 = 1194$mm (the whole wing span is 1312mm with, and 1286mm without tip) and $r = 25$mm.

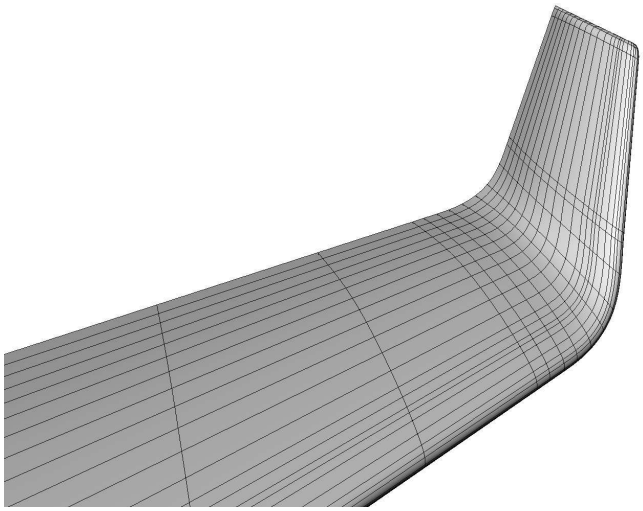

Fig. 11 Winglet with dihedral angle $a = 10^\circ$ and bending angle $w = 60^\circ$.

An IGES file containing the configuration that was actually manufactured and used for the wind tunnel experiments can be downloaded from [5].

3 Elliptic Grid Generation

In this section, which can be considered as a follow-up to [9], we show how an elliptic grid generation system can be integrated into our B-spline based system using collocation methods.

3.1 Spekreijse's Grid Generation System

Among the various elliptic grid generation methods that are described in the literature we have chosen Spekreijse's approach which can be very briefly summarized as follows. Let $\hat{\mathbf{x}}(\mathbf{s})$ be a harmonic mapping from the d-dimensional parameter space $\mathscr{P}$ onto the physical domain $\mathscr{D}$ and $\mathbf{s}(\xi)$ be a so-called control mapping from the computational domain $\mathscr{C}$ onto the parameter domain $\mathscr{P}$. Then the composite mapping

$$\mathbf{x}(\xi) = \hat{\mathbf{x}}(\mathbf{s}(\xi)) : \quad \mathscr{C} \longrightarrow \mathscr{D} \tag{2}$$

fulfills a differential equation of the form

$$L(\mathbf{x}) = \sum_{i,j=1}^{d} g_{ij} \frac{\partial^2 \mathbf{x}}{\partial \xi_i \xi_j} + \sum_{k=1}^{d} P_k \frac{\partial \mathbf{x}}{\partial \xi_k} = 0. \tag{3}$$

Here, denoting by $J = \det \mathbf{x}'(\xi)$ the Jacobian of the composite mapping, we have

$$P_k = \sum_{i,j=1}^{d} J^2 g^{ij} P_{ij}^k, \tag{4}$$

where the g_{ij} and g^{ij} are the covariant and contravariant metric tensors defined by

$$g_{ij} = \frac{\partial \mathbf{x}}{\partial \xi_i} \cdot \frac{\partial \mathbf{x}}{\partial \xi_j}, \quad \sum_{k=1}^{d} g_{ik} g^{kj} = \delta_{ij}, \tag{5}$$

the P_{ij}^k are the components of the vector

$$\mathbf{P}_{ij} = -T^{-1} \frac{\partial^2 \mathbf{s}}{\partial \xi_i \partial \xi_j}, \tag{6}$$

and $T = \mathbf{s}'(\xi)$ is the Jacobian matrix of the control mapping. This PDE has to be solved numerically, and in the present context it is a natural idea to use a B-spline collocation method to do so because this way the solution will be readily represented in the desired format.

3.2 B-Spline Collocation

The general idea of collocation is to seek a function that satisfies the differential equation at certain points, the collocation points. In a way collocation is similar to interpolation, but in contrast to interpolation we do not match function values but certain combination of function and derivative values. In order to simplify the notation we concentrate on the bivariate case from now on and denote the Cartesian coordinates of the computational domain, the unit square, with $\xi = (u,v)$ and of the parameter domain with $\mathbf{s} = (s,t)$. Hence, we search a function of the form (1) which fulfills

$$L\mathbf{x}(\hat{u}_i, \hat{v}_j) = 0, \quad i = 1,\ldots,N-1, \quad j = 1,\ldots,M-1, \tag{7}$$

at certain collocation points $\hat{u}_i$, $\hat{v}_j$, yet to be chosen. Moreover, Dirichlet boundary conditions are imposed on the control points $p_{0,j}$, $p_{N,j}$, $j = 0,\ldots,M$, and $p_{i,0}$, $p_{i,M}$, $i = 0,\ldots,N$. Among various available collocation schemes, we prefer a scheme that works for splines with arbitrary knot sequences and uses the *Greville abscissae* defined by

$$\check{u}_i = \frac{1}{p} \sum_{k=i+1}^{i+p} u_k, \tag{8}$$

as collocation points. This choice is motivated by the Schoenberg-Whitney Theorem, see [13], which says that the interpolation problem $x(\hat{u}_i) = f_i$ is well posed if, and only if, every $\hat{u}_i$ lies in the support of the $i-th$ B-spline function, i.e., if $N_i(\hat{u}_i) > 0$. As one can easily verify, the Greville abscissae always give a set of as many distinct points, as the spline has control points and they fulfill the conditions of the Schoenberg-Whitney Theorem.

The Schoenberg-Whitney Theorem is also the reason why the strategy to use a standard finite difference code followed by an interpolation algorithm in order to generate elliptic B-spline grids sometimes fails to produce good results. In fact, a typical finite difference code is based on the assumption that the grid points $\mathbf{x}_{i,j}$ are numerical approximations of regularly spaced values $\mathbf{x}(ih_u, jh_v)$. However, depending on the structure of the underlying spline it could become necessary to work with unevenly spaced grid points in order to fulfill the stipulations of the Schoenberg-Whitney Theorem during the interpolation process.

3.3 Application Example

The afore-mentioned collocation scheme has been implemented and tested for planar grids, surfaces and volume grids. In order to solve the PDE we just follow the standard approach and use a fixed point iteration, freezing the metric coefficients in Equation (3) in order to get a linear system in every single iteration. Then we apply the collocation scheme to the linearized equations.

As a first application we present the grid in a block that is taken from a grid for a dual-bell configuration, see Figure 12. The boundaries are approximatively parameterized by arc length, so that we can use the identity mapping as control mapping. Hence, all control functions P_{ij}^k are zero and the resulting grid mapping is harmonic. However, the spacing of the control points displayed in the upper plot is rather irregular. This irregularity stems from an adaptive B-spline approximation algorithm which tries to resolve the different features of the nozzle contour and from the necessity to mutually insert the knots which are not present in the representation of the opposite boundary in order to build a tensor product. However, the resulting numerical grid, which is computed by evaluation of the B-spline function can be seen to have the desired smoothness properties.

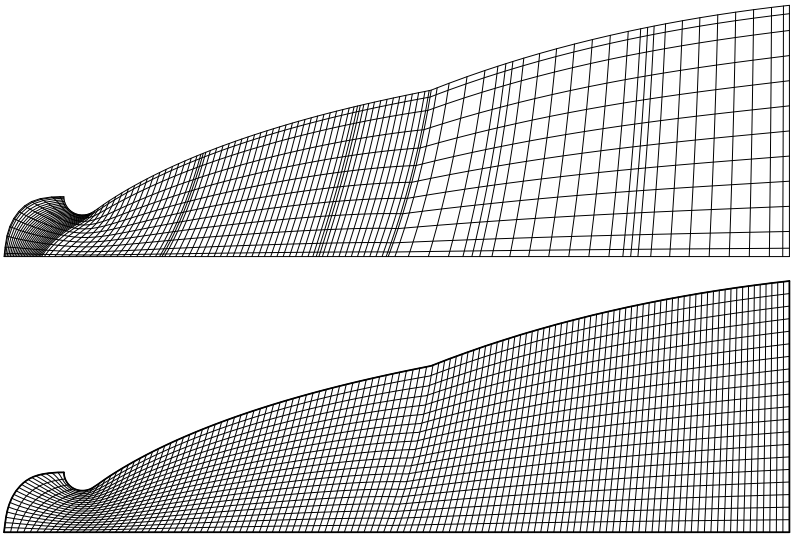

Fig. 12 Control points and harmonic mesh for the dual bell.

However, as is well known, harmonic grid generation systems have the tendency to push away the grid lines from concave boundaries and the grids are in general not orthogonal at the boundaries. This is clearly demonstrated in Figure 13.

3.4 Boundary Orthogonality

There remains the problem to determine in Spekreijse's approach suitable control functions in order to incorporate desired features into the grids. In order to find a control mapping that ensures boundary orthogonality Spekreijse proposes to pro-

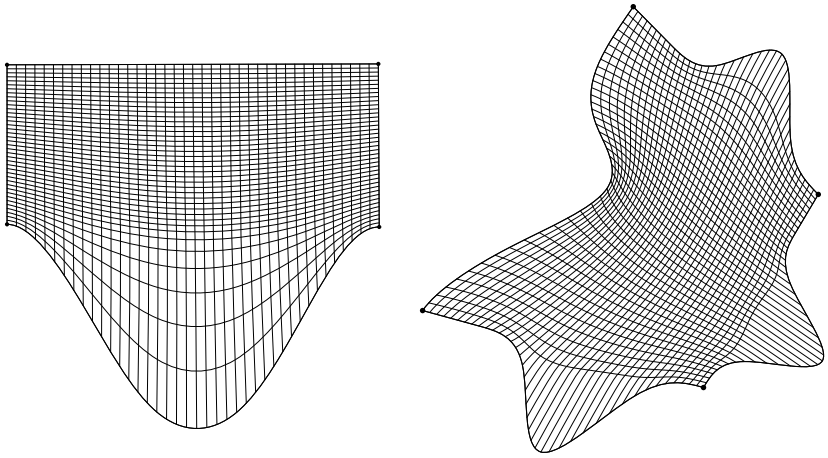

Fig. 13 Discretization with 40×40 control points

ceed as follows. The main idea is to compose a harmonic mapping with a control mapping that are both orthogonal at the boundary. The inverse of the harmonic mapping $\hat{s}(\mathbf{x})$ we search for fulfills mixed Dirichlet and Neumann boundary conditions, in particular $\partial\hat{s}/\partial\mathbf{n} = 0$ at the boundaries $\mathbf{x}(u,0)$ and $\mathbf{x}(u,1)$ and $\partial\hat{t}/\partial\mathbf{n} = 0$ at the boundaries $\mathbf{x}(0,v)$ and $\mathbf{x}(1,v)$ of the physical domain. At this point let us assume that a folding-free grid $\mathbf{x}(\xi)$ is already available. This grid may be, for instance, a transfinite interpolant or the solution of the purely harmonic grid generation system. We use this grid to discretize the above problem on a uniform mesh.

In the first step we use this given grid to transform the Laplace equation into the computational domain. The components (s,t) of the composed mapping $\mathbf{s}(\xi) = \hat{\mathbf{s}}(\mathbf{x}(\xi))$ fulfill individually a partial differential equation, which for brevity we only present for the s-component:

$$\operatorname{div}(A \operatorname{grad} s) = 0 \tag{9}$$

where

$$A = J \begin{pmatrix} g^{11} & g^{12} \\ g^{12} & g^{22} \end{pmatrix} = \frac{1}{J} \begin{pmatrix} g_{22} & -g_{12} \\ -g_{12} & g_{11} \end{pmatrix}. \tag{10}$$

The Neumann boundary condition $\partial s/\partial\mathbf{n} = 0$ transforms to

$$(\operatorname{grad} s, A\mathbf{n}) = 0 \tag{11}$$

at the corresponding boundary in $\mathscr{C}$. The solution of this problem gives us a one-to-one mapping of the boundaries $\partial\mathscr{C} \longrightarrow \partial\mathscr{P}$. In the second step we complete this boundary mapping to a suitable control mapping that fulfills the *orthogonality conditions* $\partial t/\partial u = 0$ along the boundaries $u = 0$ and $u = 1$ and $\partial s/\partial v = 0$ along

the boundaries $v = 0$ and $v = 1$ using an algebraic grid generation method. For the details of this method we refer to [20].

Whereas the discretization of Equation (3) by collocation is straightforward it is in this case equally convenient to discretize Equation (9) with a finite volume method. For this we observe that for any control volume Ω in the computational domain the equation

$$\int_{\partial\Omega} (\operatorname{grad} s, A\mathbf{n})\, d\sigma = 0 \tag{12}$$

holds. Of course, as control volumes we will choose rectangles of the form $[u_i, u_{i+1}] \times [v_j, v_{j+1}]$. Again we want to represent the control mapping as tensor product B-spline. Therefore, the integral over the boundary of the control volume is composed of segments of the form

$$\begin{aligned}
&\int_{u_i}^{u_{i+1}} (\operatorname{grad} s, A\mathbf{n})\, d\sigma = \int_{u_i}^{u_{i+1}} \left(\begin{pmatrix} s_u \\ s_v \end{pmatrix}, A \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right) d\sigma \\
&= \sum_{i,j} p_{ij} \left[-N_j(v) \int_{u_i}^{u_{i+1}} N_i'(u) \frac{g_{12}}{J}\, du + N_j'(v) \int_{u_1}^{u_2} N_i(u) \frac{g_{11}}{J}\, du \right]
\end{aligned}$$

and

$$\begin{aligned}
&\int_{v_j}^{v_{j+1}} (\operatorname{grad} s, A\mathbf{n})\, d\sigma = \int_{v_j}^{v_{j+1}} \left(\begin{pmatrix} s_u \\ s_v \end{pmatrix}, A \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right) d\sigma \\
&= \sum_{i,j} p_{ij} \left[-N_i(u) \int_{v_j}^{v_{j+1}} N_j'(v) \frac{g_{12}}{J}\, dv + N_i'(u) \int_{v_j}^{v_{j+1}} N_j(v) \frac{g_{22}}{J}\, du \right].
\end{aligned}$$

The integrals in the brackets enter the matrix of the discretized problem and can cheaply be evaluated by quadrature formulas. In order to obtain as many equations as control points we center the control volumes around the Greville abscissae by choosing

$$\begin{aligned}
u_i &= \frac{1}{2}(\breve{u}_{i-1} + \breve{u}_i), \quad i = 1, \ldots, N-1, \quad u_0 = 0, \quad u_N = 1, \\
v_i &= \frac{1}{2}(\breve{v}_{j-1} + \breve{v}_j), \quad i = 1, \ldots, N-1, \quad v_0 = 0, \quad v_M = 1.
\end{aligned} \tag{13}$$

Due to Equation (11), the discretization at boundary grid points is also straightforward. Figure 3.4 shows the control map and the smooth evaluation of the resulting orthogonal grid. It can very well be observed how the new boundary parameterization of the grid serves to enforce orthogonality along the boundary. However, since we still do not have control over the boundary spacing, the grid is rather distorted. This will be addresses in the next section.

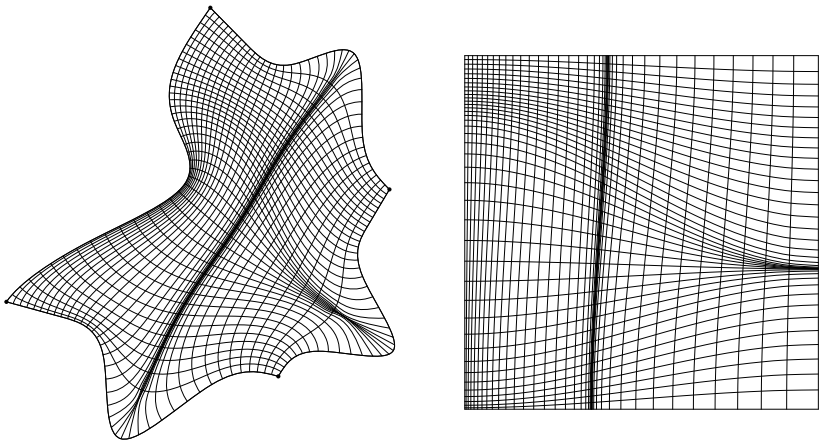

Fig. 14 Grid, orthogonal near the boundaries, and its control mapping

3.5 Complete Boundary Control

In order to achieve control of both the angles and the grid spacing near the boundary we need to look for control mappings that do not only have the right boundary parameterization and fulfill the orthogonality conditions but also take prescribed values $\partial t/\partial v$ along the boundaries $u = 0$ and $u = 1$ and $\partial s/\partial u$ along the boundaries $v = 0$ and $v = 1$. In principle, it is possible to construct such grids with algebraic methods, namely with so-called cubic Coons-Patches. However, algebraic methods easily tend to produce grid-folding. Hence we take up again the ideas of Spekreijse who uses solutions of the biharmonic equation to incorporate these additional boundary conditions into the control mapping.

To set up this method, we assume that we have computed already a grid $\mathbf{x}(u,v)$ which is orthogonal at the boundaries and its corresponding control mapping $\mathbf{s}(u,v)$. In a parametric grid the spacing of the first meshline in physical space is, within second order accuracy, proportional to cross derivatives, i.e., to

$$\|\frac{\partial \mathbf{x}}{\partial u}(u,v)\| \text{ and } \|\frac{\partial \mathbf{x}}{\partial v}(u,v)\|$$

on the boundaries $u = 0$, $u = 1$ or $v = 0$, $v = 1$ respectively. In the given grid $\mathbf{x}(u,v)$, these quantities can be evaluated and compared with the desired values, lets say $f(u,v)$. Since we assume that the control mapping the $\mathbf{s}(u,v)$ is already orthogonal at the boundary and because the harmonic transformation does not change if we do not change the boundary distribution in the parameter space, we can conclude that the norm of the above cross derivatives are proportional to the cross derivatives of the control mapping. That means, to achieve the desired spacing in the physical domain, the cross derivatives of the improved control mapping $\tilde{\mathbf{s}} = (\tilde{s},\tilde{t})$ should take

the values

$$\frac{\partial \tilde{t}}{\partial v}(u,v) = \frac{\partial t}{\partial v}(u,v)\frac{f(u,v)}{||\frac{\partial \mathbf{x}}{\partial v}(u,v)||}. \tag{14}$$

$$\frac{\partial \tilde{s}}{\partial u}(u,v) = \frac{\partial s}{\partial u}(u,v)\frac{f(u,v)}{||\frac{\partial \mathbf{x}}{\partial u}(u,v)||} \tag{15}$$

Hence, our new control mapping for complete boundary control is found by solving the biharmonic equation

$$\Delta\Delta\tilde{\mathbf{s}} = 0 \tag{16}$$

subject to the Dirichlet boundary conditions

$$\tilde{\mathbf{s}}(u,v) = \mathbf{s}(u,v) \tag{17}$$

and the Neumann conditions (14) and (15). In addition, as before, $\partial\tilde{t}/\partial u = 0$ has to hold along the boundaries $u = 0$ and $u = 1$ and $\partial\tilde{s}/\partial v = 0$ along the boundaries $v = 0$ and $v = 1$.

In this case, instead of employing a B-spline collocation method to solve the biharmonic equation, we apply a finite-difference algorithm, see [2]. The result is subsequently interpolated by a B-spline mapping. In this case there are no problems with irregular knot-sequences, because the control mapping does not enter the construction of the physical grid, so that we can just use uniform discretizations in the parameter space. Figure 3.5 shows the grid for the above test configuration which is generated in the way just described with control of the first spacing. In particular, notice the excellent smoothness of the control mapping compared to Figure 3.4.

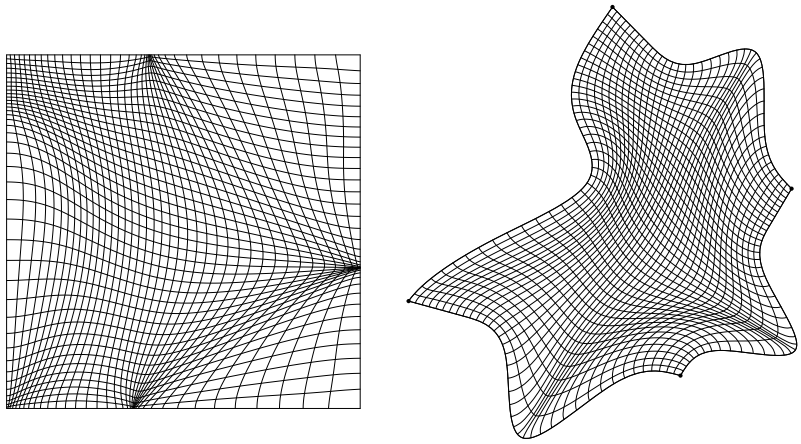

Fig. 15 Final grid with complete boundary control.

Generally, from our practical experiences we conclude that the collocation scheme proposed above offers a stable and reliable way to realize elliptic grid generation methods directly in terms of B-spline representations. Due to the higher approximation order of splines compared with finite differences, it might even be computationally more efficient than the standard discretization method.

4 Deforming Grids

In general, when solving unsteady problems where the mesh has to conform to the instantaneous shape of a deforming body, the grid has to be updated in every time step. Therefore the grid deformation process should be cheap, ideally taking only a small fraction of the overall CPU time required by the flow solver.

In practice it turns out that multi-block grids offer several advantages over unstructured methods when it comes to grid deformation. As long as the deformations are moderate, it is usually possible to keep the topology and the connectivity of the grid unchanged, whereas moving unstructured grids often require some kind of remeshing. Another point is that transfinite interpolation can be used to compute the displacement of the interior grid points in a block or face from previously computed displacements of the grid points on the boundaries. Therefore the main problem is the computation of the displacement of the vertices and the edges of the blocks, i.e., of the so-called *framework*. For this task more sophisticated and expensive methods have to be used. Fortunately this is feasible since the amount of points belonging to the framework is negligible compared to the overall number of grid points.

4.1 Deforming the Framework

Our strategy for deforming the framework is based on interpolation with *radial basis functions*. This method is rather general and can be applied to arbitrary types of grids, both structured and unstructured. The main idea is to reduce the deformation problem to a scattered data interpolation problem. To this end, suppose that on the configuration surface and possibly on the far field boundaries the displacements $\mathbf{d}_k$ of some specifically chosen points $\mathbf{x}_k$, $k = 1, \ldots, N$, are prescribed. The objective is now to find a smooth function $\mathbf{d}(\mathbf{x})$ which assigns a displacement to each point in the physical space and interpolates the given data: $\mathbf{d}(\mathbf{x}_k) = \mathbf{d}_k$ for all k.

Once this problem has been solved, the function $\mathbf{d}(\mathbf{x})$ is used to determine the displacements of the grid elements constituting the multiblock framework, for the technical details see [17]. Typically the data sites $\mathbf{x}_k$ are the vertices of the multiblock grid which lie on the configuration surface plus possibly a small number of additional specific points, for instance, the leading and trailing edge of a profile. Spekreijse et. al. [22] estimate that even for a complex fighter configuration no more

than 100 data points are necessary to make the method described in the current section work.

However, the data sites $\mathbf{x}_k$ bear no regular structure. A standard numerical method to address such a *scattered data interpolation problem* is interpolation by radial basis functions (RBFs). The general ansatz is to interpolate the given data by a function of the form

$$\mathbf{d}(\mathbf{x}) = \sum_{k=1}^{N} \mathbf{a}_k \Phi(\mathbf{x} - \mathbf{x}_k) + \mathbf{p}(\mathbf{x}) \tag{18}$$

where $\mathbf{p}$ is a polynomial of a low, fixed maximum order $\mathrm{ord}(p) \leq m$ and the function Φ actually depends only on the distance $||\mathbf{x} - \mathbf{x}_k||$. That means there exists a univariate function $\phi : \mathbb{R}_+ \longrightarrow \mathbb{R}$ such that $\Phi(\mathbf{x} - \mathbf{y}) = \phi(||\mathbf{x} - \mathbf{y}||)$. The data sites $\mathbf{x}_k$ are usually called *centers*. The coefficients $\mathbf{a}_k$ and the polynomial $\mathbf{p}$ are determined by the interpolation conditions

$$\mathbf{d}(\mathbf{x}_k) = \mathbf{d}_k, \quad k = 1, \ldots, N, \tag{19}$$

and the additional requirement

$$\sum_{k=1}^{N} \mathbf{a}_k \mathbf{q}(\mathbf{x}_k) = 0 \tag{20}$$

for all polynomials $\mathbf{q}$ of order $\mathrm{ord}(q) \leq m$.

In this general ansatz we now have to choose a concrete radial basis function. One popular choice is to take the fundamental solutions of the biharmonic equation. In 2D the fundamental solution is the so-called thin plate spline

$$\Phi(||\mathbf{x} - \mathbf{y}||) = \frac{-1}{8\pi} ||\mathbf{x} - \mathbf{y}||^2 \log(||\mathbf{x} - \mathbf{y}||). \tag{21}$$

In 3D the fundamental solution of the biharmonic equation is essentially just the distance function itself:

$$\Phi(||\mathbf{x} - \mathbf{y}||) = \frac{||\mathbf{x} - \mathbf{y}||}{8\pi}. \tag{22}$$

The theory of interpolation with radial basis functions, which can be found for example in [23], requires that the order of $\mathbf{q}$ must be at least $m = 1$, and indeed this minimal choice is the only one considered in [16] and [22]. However, we have a slight preference for the choice $m = 2$. This is motivated by the following lemma.

Lemma 1. *If $m \geq 2$ then $\mathbf{d}(\mathbf{x})$ reproduces each affine transformation exactly, i.e., if $\mathbf{d}_k = R\mathbf{x}_k + \mathbf{v}$ with $R \in \mathbb{R}^{d \times d}$ and $\mathbf{v} \in \mathbb{R}^d$ then $\mathbf{d}(\mathbf{x}) = R\mathbf{x} + \mathbf{v}$ for all $\mathbf{x} \in \mathbb{R}^d$.*

A proof can be found in [17]. Reproduction of affine functions has the useful implication, that if the configuration performs a rigid body transformation, a rotation or translation, this movement is exactly carried over to the whole grid.

4.2 Example: High Lift Configuration

As a practical example we apply this technique to the high lift configuration. The construction of the blocking shown in Figure 16 follows the strategy outlined in [12]. There are offset areas around the single elements, and we have constructed a Cartesian bounding box around the configuration such that a Cartesian far field could easily be added.

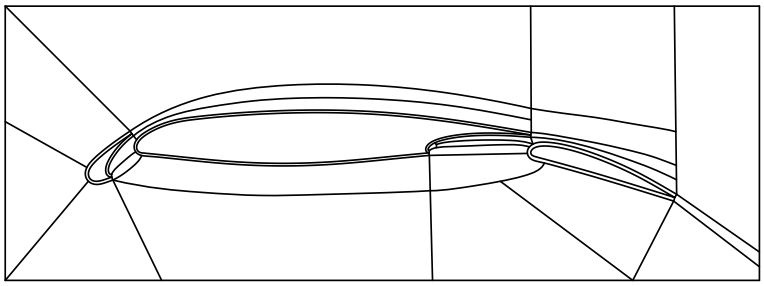

Fig. 16 Sample blocking for the three-element configuration.

The slat and the flap can be rotated around the centers that are indicated in Figure 17.

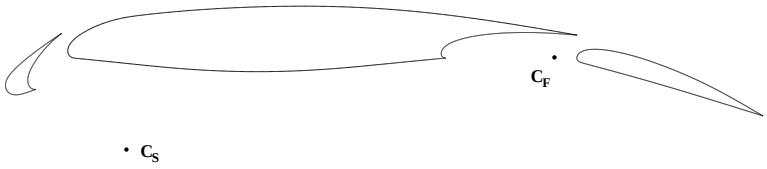

Fig. 17 Three element high lift configuration. The dots mark the rotation centers for the slat and flap riggings.

The offset areas are considered to be rigidly connected to their supporting element. On each offset curve 20 support points are chosen for the volume spline method. Furthermore on each edge of the bounding box 10 support points are defined. The bounding box, of course, is not allowed to move during the deformation.

The edges of the block structure between the offset curves and the bounding box are moved according to the radial basis interpolation function. An overall view of a deformed block is given in Figure 18.

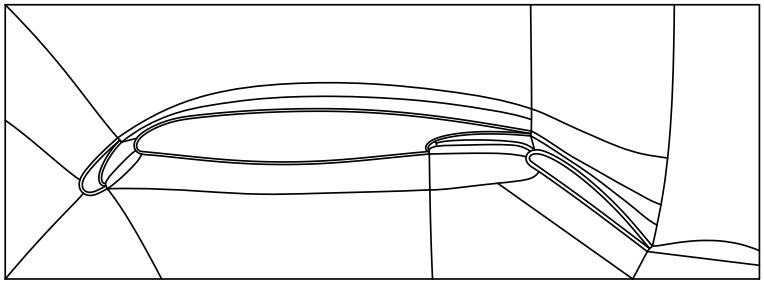

Fig. 18 Deformed configuration. Here the slat has been rotated 8° counter-clockwise and the flap 20° clockwise.

Figure 19 shows scaled up sections of this plot near the flap and the slat. It turns out, that the most problematic part of the configuration is the area between the slat and the main element. Here highly distorted grids may arise if one reduces the slat gap, i.e., if one rotates the slat clock-wise. The problem is that the angles between the offset curves and the edge connecting the offset curve around the main element with the offset area behind the slat become very large. The situation can be improved, if one allows the right hand vertex of this edge to slide freely on the offset curve.

This possibility is a special feature of the grid generation technique presented here. With discrete grids a similar strategy is in general not available because the number of grid points in each block is determined by the position of such a supported vertex, and therefore reattaching a boundary curve to another vertex would at least require a complete remeshing of the affected blocks or even violate global combinatorial constraints. However, to use this feature in a nonstationary calculation, it would be necessary to have a flow solver, that could handle changes in the cell connectivity during one time step. Therefore at the moment, QUADFLOW can use this feature only for steady state parameter studies, if for example one wants to examine the effects of varying flap and slat riggings.

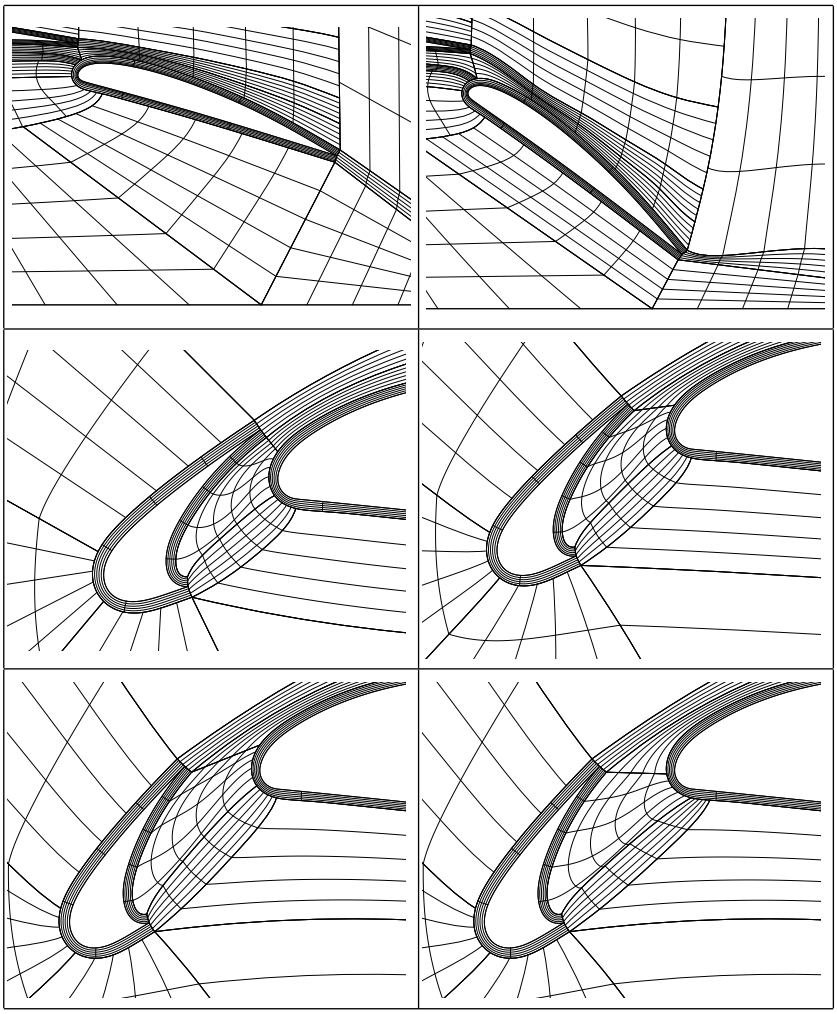

Fig. 19 Detailed viewed at the slat and flap. First row: grid around slat in initial configuration (left) and rotated 20° clock-wise. Second row: grid around flap in starting configuration (left and rotated counter-clockwise by 6°. Third row: same grid rotated 12 degrees counter-clockwise with vertices rigidly fixed to the offset curves (left) and vertices allowed to move on the offset curve (right).

5 Conclusion

In this paper we have considered several aspects of the B-spline based parametric grid generation system developed in project B2. Compared with classical block-structured grid generation such a system offers more flexibility and better integration to CAD-software. However it is well known that the main disadvantage of block-

structured grid generation is the fact, that there are hardly any automatic methods to generate the block-decompositions. While some steps in this direction has been taken this problem has not really been addressed in a systematic way in the current work. It would therefore be one important objective of further research. Another direction would be to combine parametric representations with unstructured grid technology or with subdivision schemes as known from CAGD. In particular, the latter approach seems to be promising because of its close connections with both geometry and adaptivity.

6 Acknowledgement

This work has been performed with funding by the Deutsche Forschungsgemeinschaft in the Collaborative Research Center SFB401 *Flow Modulation and Fluid-Structure Interaction at Airplane Wings* of the RWTH Aachen University, Aachen, Germany.

References

1. Ballmann, J. (Ed.): Flow Modulation and Fluid-Structure-Interaction at Airplane Wings. No. 84 in Numerical Notes on Fluid Mechanics. Springer (2003)
2. Bjorstad, P.E.: Numerical Solution of the Biharmonic Equation. Ph.D. dissertation, Stanford University (1980)
3. Brakhage, K.-H.: Ein menugesteuertes, intelligentes System zur zwei- und dreidimensionalen Computergeometrie. VDI Reihe 20 CAD/CAM, Nr. 26 Edition. VDI Verlag (1990)
4. Brakhage, K.-H.: Wincag-education software for geometry. In: 11^{th} International Conference on Engineering Computer Graphics and Descriptive Geometry. Guangzhou, China (2004)
5. Brakhage, K.-H.: `http://www.igpm.rwth-aachen.de/brakhage/SFBmodel`
6. Brakhage, K.-H., Müller, S.: Algebraic–hyperbolic grid generation with precise control of intersection of angles. J. Num. Meth. in Fluids **33** (1), 89–123 (2000)
7. Brakhage, K.-H., Lamby, P.: Generating airplane wings for numerical simulation and manufacturing. In: Soni, B., Häuser, J., Eiseman, P. (Eds.): Proceedings of the 9th International Conference on Numerical Grid Generation in Computational Field Simulations, San Jose, California, USA (2005)
8. Brakhage, K.-H., Lamby, P.: Modeling of Airplane Wings with Winglets. In: Soni, B., Häuser, J., Eiseman, P. (Eds.), Proceedings of the 10th International Conference on Numerical Grid Generation in Computational Field Simulations, FORTH, Crete, Greece (2007)
9. Lamby, P., Brakhage, K.-H.: Elliptic Grid Generation by B-Spline Collocation. In: Soni, B., Häuser, J., Eiseman, P. (Eds.): Proceedings of the 10th International Conference on Numerical Grid Generation in Computational Field Simulations, FORTH, Crete, Greece (2007)
10. Brakhage, K.-H., Lamby, P.: Application of B-Spline Techniques to the Modeling of Airplane Wings and Numerical Grid Generation. Computer Aided Geometric Design **2**5, 738-750 (2008)
11. Bramkamp, F., Lamby, P., Müller, S.: An adaptive multiscale finite volume solver for unsteady and steady state flow computations. Journal of Computational Physics **1**97, 460–490 (2004)
12. Bramkamp F., Gottschlich-Müller, B., Hesse, M., Lamby, P., Müller, S., Ballmann, J., Brakhage, K.-H., Dahmen, W.: *H*-adaptive Multiscale Schemes for the Compressible Navier–Stokes Equations — Polyhedral Discretization, Data Compression and Mesh Generation.

In Ballmann, J. (Ed.): Flow Modulation and Fluid-Structure-Interaction at Airplane Wings, No. 84 in Numerical Notes on Fluid Mechanics, 125–204. Springer-Verlag (2003)
13. deBoor, C.: A Practical Guide To Splines, Springer-Verlag (1978)
14. Farin, G., Rein, G., Sapidis, N., Worsey, A.: Fairing cubic B-spline curves. Computer Aided Geometric Design 4, 91–103 (1987)
15. Hoschek, J., Lasser, D.: Grundlagen der geometrischen Datenverarbeitung. 2nd Edition, Teubner Verlag, Stuttgart (1992)
16. Hounjet M.H.L., Meijer, J.J.: Evaluation of elastomechanical and aerodynamic data transfer methods for non-planar configurations in computational aeroelastic analysis. In: Proceedings International Forum on Aeroelasticity and Structural Dynamics, Manchester, UK (1995)
17. Lamby, P.: Parametric Grid Generation and Application to Adaptive Flow Simulations. PhD thesis, RWTH Aachen University (2007)
18. Moir, I.: Measurements on a two-dimensional aerofoil with high lift devices. AGARD-AR-303 Vol. I+II, DRA, Farnborough (1994)
19. Runborg, O.: A Multiblock Mesh Manager. In: Computational Mesh Adaptation, No. 69 of Numerical Notes on Fluid Mechanics. European Commision, Directorate General XII, Science, Research and Development, Vieweg (1999)
20. Spekreijse, S.: Elliptic Grid Generation Based on Laplace-Equations and Algebraic Transformations. Journal of Computational Physics **118**, 38–61 (1995)
21. Spekreijse, S., Boerstoel, J.W.: Multiblock Grid Generation. Part 2: Multiblock Aspects. In H.Deconinck (Ed.): Computational Fluid Dynamics. VKI Lecture Series 1996-06, 1–39. von Karman Institute for Fluid Dynamics (1996).
22. Spekreijse, S., Prananta, B., Kok, J.: A simple, robust and fast algorithm to compute deformations of multi-block structured grids. Tech. Rep. NLR-TP-2002-105, National Aerospace Laboratory NLR (2002)
23. Wendland, H.: Scattered Data Approximation. Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press (2005)