Computers and Fluids 292 (2025) 106588

Contents lists available at ScienceDirect

Computers and Fluids

journal homepage: www.elsevier.com/locate/compfluid

A differentiated geometry blade parameterization methodology for gas turbines

Christian Voß [a,*], Martin Siggel [a], Jan Backhaus [a], Georgios Goinis [a], Andreas Pahs [b]

[a] German Aerospace Center (DLR), Institute of Propulsion Technology, Linder Hoehe, Cologne, 51147, Germany
[b] German Aerospace Center (DLR), Institute of Propulsion Technology, Bunsenstrasse 10, Goettingen, 37073, Germany

ARTICLE INFO

Keywords:
Turbomachinery design
Gas turbine
NURBS
B-spline
CAD
Blade geometry
Parameterization
Algorithmic differentiation

ABSTRACT

The present paper is about the turbomachinery blade geometry parameterization, which has been developed at the DLR Institute of Propulsion Technology for the last two decades. It was implemented into the software BladeGen, which is used to design blades and vanes for axial and radial compressors as well as for fans, propellers and turbines. The parameterization uses typical geometric variables such as characteristic angles and lengths in combination with B-spline parameters like control points and knot vectors to generate a variable number of two-dimensional profile-sections, which are transformed by a conformal mapping into three-dimensional space and connected via NURBS surfaces to form blade surfaces and solids.

This publication presents the application philosophy and some parameterization and implementation details. The intuitive handling and some software enhancements such as algorithmic differentiation capabilities are also demonstrated by application examples.

1. Introduction

The parameterization of turbomachinery blades and airfoil profiles has been the subject of research for many decades and is, of course, still essential in day-to-day design work. NACA developed a first series of thoroughly tested 2D airfoils during the late 1920s and into the 1930s. In these early years, the profiles were mostly parameterized using a thickness distribution plotted on a camber line. Simple 3rd or 4th order polynomials over the normalized arc length were commonly used. Examples of such methods were published in [1] and earlier methods were included in [2]. In the following decades it became clear that these methods are not ideal for designing the suction side and pressure side independently of each other to achieve the desired performance in typical turbomachinery flows. Modern methods vary each surface (suction and pressure side) independently and use NURBS representations (see [3]) to ensure flexibility and usability.

Further requirements for the parameterization of turbomachinery blades result from the increased use of multidisciplinary optimization routines in recent years. Regardless of whether these optimization methods require gradients [4], or use surrogate models such as Gaussian processes [5,6], or combine both methods [7], they always require a stable, batch-capable parameterization that allows the essential geometry features to be targeted with as few parameters as possible.

Two main approaches to blade parameterization have become established: engineering parameters and free form deformation (FFD). While FFD allows a great deal of design freedom in the deformation of existing blades, the use of engineering parameters to control blade geometry has the great advantage that performance can be inferred from the design parameters, as the parameters are intuitive to the designer (see [8]). It also makes it easier to transfer design knowledge to new blades, and the number of parameters is smaller compared to FFD.

Recently, the design of compressor, fan and turbine blades has reached the level of custom design. Therefore, it is required that the design method must be able to optimize the blade shape at a very detailed level.

This paper presents a parametric blade geometry design method using engineering parameters that can satisfy the custom design of any type of turbomachinery blade and is highly suitable for gas turbine design (similar to [3]). With the BladeGen software development, started in 2003, the DLR institute of propulsion technology tackled the target of a new blade parameterization, using intuitive and CAD compatible NURBS description and satisfying the requests of human design engineers and automated optimization procedures at once. The developed software fulfills the objectives mentioned in [3]:

- Support any type of blade configuration.
- Use engineering parameters with an intuitive geometrical meaning.

* Corresponding author.
E-mail address: christian.voss@dlr.de (C. Voß).

https://doi.org/10.1016/j.compfluid.2025.106588
Received 13 November 2024; Received in revised form 24 January 2025; Accepted 16 February 2025
Available online 25 February 2025

- Produce smooth geometries with continuous curvature (G2 continuity).
- Retain compatibility with Computer-Aided Design (CAD) software for further analysis.
- Allow the designer to impose geometric constraints.
- Provide the sensitivity of the shape with respect to the design variables.
- Be computationally cheap in terms of execution time and memory usage.

However, unlike [3], a more direct parameterization of the suction side is used instead of working with thickness distributions. Starting with the first axial compressor airfoil designs and optimizations (see eg. [9]), the software has been used in a large number of projects and is still being continuously developed and improved until today. Over the years, the program has been gradually expanded with new functionalities such as blade fillets, cooled turbine blades, centrifugal compressor blades, wedge diffusers etc. Furthermore, new types of profile pressure side parameterization, specification of radial leading and trailing edge lines, linear transformations and blade cutting features have been added to meet the needs of a growing user community. The software is widely used at different DLR institutes to design fans, axial and radial compressors, turbines, ventilators etc. and the implemented interfaces to subsequent CFD and CSM calculations as well as the interface to automated optimization frameworks are well established (see [5]). In addition, the original software designed for Linux operating systems has been extended to Windows. C++ and Python interfaces have been added and nowadays BladeGen is an integral part of the DLR's preliminary design environment GTlab [10].

The present paper will give an overview of the BladeGen software, explain essential inputs, methods, mathematical basics, and presents the different blade parameterization schemes using specific design examples. Beside that, new developments like the Python interface and the algorithmically differentiated version of BladeGen are introduced.

2. Global design approach and coordinate systems

Aerodynamic turbomachinery engineers like to use descriptive blade parameters like angles and thicknesses, which may result from preliminary aerodynamic codes (meanline, throughflow, two-dimensional profile aerodynamics), to describe a blade surface. Thus, the presented parameterization uses such intuitive geometric parameters in combination with B-spline control points to create multiple profile sections and assemble them into a three-dimensional blade surface. BladeGen's design approach can roughly be subdivided into four parts:

1. The profiles are constructed as B-spline curves in a two-dimensional coordinate system (see Section 3).
2. The two-dimensional staggered profiles are transformed into the three-dimensional coordinate system (stacking see Section 4).
3. The three-dimensional profiles are skinned by a three-dimensional B-spline tensor product surface (see Section 4).
4. Special treatments like a generation of fillets, a cutting of some blade regions, and linear transformations are performed (see Section 6).

BladeGen allows the use of different coordinate systems depending on the application. By convention, x denotes the axis of rotation and thus coincides with the shaft of the turbomachinery component, (x, y, z) is the right hand rule cartesian coordinate system, $r = \sqrt{y^2 + z^2}$ is the radius and $\theta = \mathrm{atan}(y/z)$ is the circumferential angle used in cylindrical coordinates:

$$\begin{aligned} x &= x \\ y &= r\sin(\theta) \\ z &= r\cos(\theta) \end{aligned} \tag{1}$$

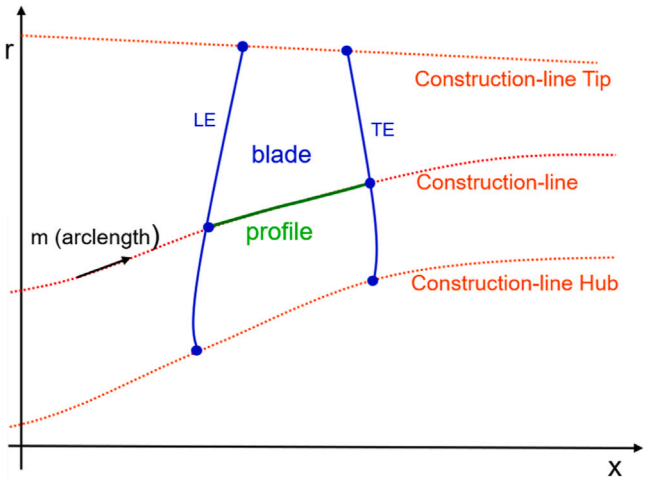


Fig. 1. Blade construction via profiles and definition of m'.

To reduce the number of design parameters, the BladeGen parameterization offers three different coordinate system options for constructing airfoils in a two-dimensional coordinate system. These are:

1. (m', θ) coordinate system: default option and mainly used for all axial fan, compressor and turbine blades.
2. $(m, r\theta)$ coordinate system: sometimes used for radial impellers, m denotes the arc length along the construction streamline.
3. (x, y) coordinate system: mainly used for linear cascades.

The default (m', θ) option, which is the only one described a bit more detailed in the present paper, was introduced by Drela and Youngren (see [11]). θ indicates the circumferential angle and m' is derived from the arc length m of the construction line as shown in Fig. 1. Choosing the dimensionless arc length

$$m' = \int \frac{dm}{r} = \int \frac{\sqrt{dx^2 + dr^2}}{r} \tag{2}$$

guarantees a conformal mapping from two-dimensional to three-dimensional space. The transformation from physical (x, y, z) space to the (m', θ) airfoil design plane is thus angle-preserving and therefore aerodynamic flow acceleration caused by airfoil curvatures behaves similar in two-dimensional and three-dimensional space. By entering a construction B-spline for each profile in (x, r) coordinates (see Fig. 1), it is possible to map each two-dimensional (m', θ) profile to its corresponding rotational construction surface. The same applies to the $(m, r\theta)$ coordinate system, since a four-dimensional (x, r, m, m') B-spline curve is internally constructed with strictly monotonously growing m and m'. Thus, by a given m or m', the corresponding x and r are easily determined by that spline.

In simple cases of linear construction splines (e.g. construction streamline Tip in Fig. 1), the inverse mapping means nothing else than "unrolling" the rotational cylindrical surface, which of course is not possible for arbitrary curved splines (e.g. construction splines hub in Fig. 1).

3. Profile design

The construction profiles are usually designed in a two-dimensional coordinate systems (see Section 2) since a profile can be represented by fewer parameters in 2D space. The profile parameterization consists of typical parameters like angles, non-dimensional lengths, thicknesses and control points, which are used to model the profiles as B-spline

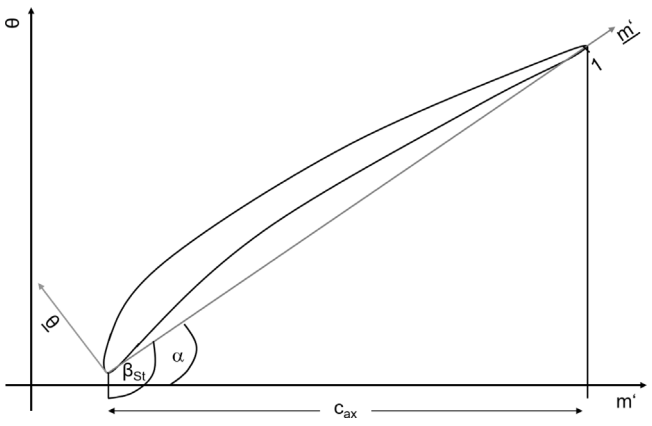


Fig. 2. Unstaggered $(\underline{m'}, \underline{\theta})$ and staggered coordinate system (m', θ).

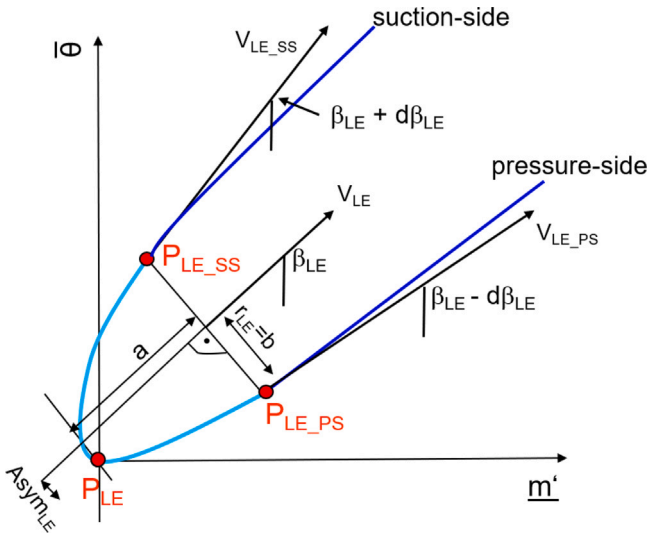


Fig. 3. Parameterization and construction of the leading edge.

curves (see [12]). Using the axial chord length c_{ax} and the stagger angle β_{st} parameters, the profile design space can be further simplified from (m', θ) to the unstaggered $(\underline{m'}, \underline{\theta})$ system, where the leading edge point is located at (0,0) and the trailing edge point is located at (1,0) as shown in Fig. 2. All other profile parameters are then given in $(\underline{m'}, \underline{\theta})$, and the re-transformation from the unstaggered system to the staggered one is given by:

$$\begin{pmatrix} m' \\ \theta \end{pmatrix} = c_{ax} \cdot \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \cdot \begin{pmatrix} \underline{m'} \\ \underline{\theta} \end{pmatrix}, \quad \alpha = \beta_{st} - \pi/2 \tag{3}$$

To start the profile design in $(\underline{m'}, \underline{\theta})$, six characteristic points are determined in a first step: three points in the leading edge region and three in the trailing edge region of the airfoil. See Fig. 3 for the points around leading edge marked in red (the three point around the trailing edge are constructed similar): The leading and trailing edge point P_{LE}, P_{TE} and the four connection points between pressure-side, leading edge, suction-side and trailing edge. The exact position of these points in $(\underline{m'}, \underline{\theta})$ are determined by geometric input parameters like the leading edge radius (r_{LE} in Fig. 3), the leading edge stretching (a/b in Fig. 3), the skewness (Asym_{LE} in Fig. 3), the leading edge- (β_{LE} in Fig. 3), stagger- (β_{st} in Fig. 2) and wedge- ($d\beta_{LE}$ in Fig. 3) angles and some other settings of minor importance (similar geometric input values are given for the trailing edge).

Beside the positioning of the characteristic points at the leading edge, the direction of the suction side curve (denoted as V_{LE_SS} in Figs. 3 and 4) is also determined by the angles and a very similar construction is performed for the trailing edge. Thus, to realize a valid parameterization of the suction side as a B-spline curve of degree $p = 3$, a few additional control points parameters are required and must be provided by the user. In Fig. 4 a B-spline with five control points was chosen as an example (parameters marked in green in Fig. 4 are not already determined by the former described parameters). Naturally, using more additional control points the possible complexity of the profile geometry can be increased. A refinement can be achieved using knot insertion (see [12]), which increases the number of control points without changing the profile shape during different design cycles.

BladeGen offers two different parameterization schemes for the construction of the profile's pressure side. The first one is identical with the construction of the suction side described above. In that case, both sides of the profiles are more or less independent of each other. This has the advantage, that the two sides of the profile can be designed independently. This freedom, however, comes with a drawback: the pressure and suction sides may intersect, leading to an impossible profile shape.

Thus, the second scheme defines the position of the free pressure side control points by a thickness distribution along the suction side. That option is often used during automated optimization procedures to meet some structural requirements by coupling the pressure side by a more or less fixed thickness distribution to a varying suction side.

The next construction step is the connection of suction and pressure side by leading and trailing edge B-spline curves. This is performed by a B-spline interpolation of the characteristic points described above (e.g P_{LE_PS}, P_{LE}, P_{LE_SS} in Fig. 3 for the leading edge) with derivative boundary conditions given by the already constructed profile sides. The user can choose, whether a C^1 or a C^2 geometric continuity between the different profile parts is modeled.

The four parts of the profile (both sides and edges) are then merged to a single B-spline curve. A knot-refinement procedure is performed to increase the number of control points before the transformation into the three-dimensional space, simply by projecting the B-spline control points back from 2D space into 3D using a non-linear transformation in general. Therefore, the final three-dimensional profiles are located only approximately on their respective construction surfaces and not in an exact, mathematical sense, at least as the construction lines are not linear.

This type of profile parameterization was initially developed for compressors and fan blades. Some useful variations for turbine profiles are therefore described in the following section.

3.1. Auxiliary turbine profile parameterization

In Fig. 5 a typical high pressure turbine profile is shown in the $\underline{m'}$-$\underline{\theta}$-system which is also used for compressor profiles (Fig. 2). The leading edge is per definition at $\underline{m'}$=0 and the trailing edge at $\underline{m'}$=1. Due to the large turning of turbine blades and the space needed for blade cooling, the profile has small angles between its tangent near the leading edge and the $\underline{\theta}$-axis. This leads to an inadvertent, massive change in the profile even when adjusting the $\underline{\theta}$-value of the control points at the leading edge only by a small value. Furthermore $\underline{m'}$-values below zero are visible which are not allowed by the design principles for compressor profiles. To overcome these issues, a pseudo skeleton line is created from the leading and trailing edge angles. By that adjustment, the same input as for compressor profiles can be used, but instead of direct $\underline{m'}$-$\underline{\theta}$-coordinates, the $\underline{m'}$-value is used as relative distance on the skeleton line and $\underline{\theta}$-value as orthogonal distance from the skeleton line. This turbine-specific profile parameterization gives additional design freedom and process robustness for turbine blades.

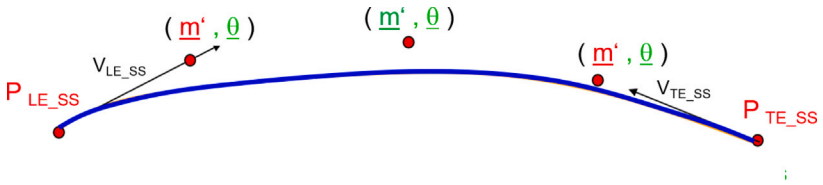


Fig. 4. Parameterization and construction of the suction side.

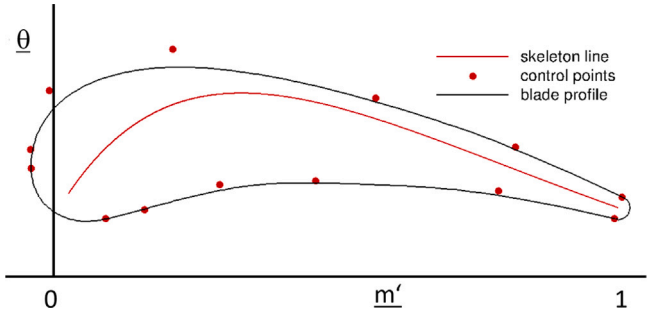


Fig. 5. Turbine profile construction in $\underline{m'}$, $\underline{\theta}$-system.

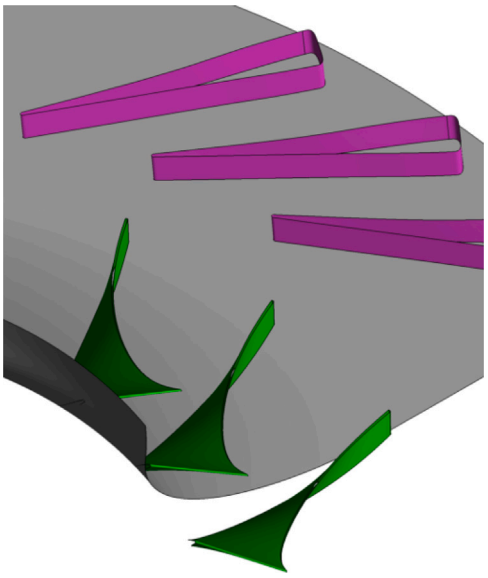


Fig. 6. BladeGen design of a radial impeller (green) with wedge diffuser (purple).

3.2. Wedge diffuser for radial compressors

In centrifugal compressors, wedge diffusers are commonly used to decelerate the flow. Their typical profiles resemble triangles with rounded corners, requiring significantly fewer parameters than the airfoils discussed in the previous sections. The parameterization of the wedge diffuser only requires the three parameters $r_{LE}, r_{TE}, \beta_{ST}$ to define the characteristic start- and end points on suction and pressure side for the 2D profiles (see Fig. 3). The suction and pressure sides themselves are then simply linear connections between these corresponding points (see Fig. 6).

4. Profile stacking

Stacking describes the procedure of mapping the constructed two-dimensional profiles (as described in the last section) into the physical three-dimensional space. There are different options available in BladeGen to stack the profiles. Each needs the four dimensional (x, r, m, m') construction curves as described above in Section 2 and shown by their (x, r)-projection via red dotted lines in Fig. 1. As soon as the two-dimensional (e.g. m', θ) profile control points should be placed on the respective three-dimensional (e.g. x, r, θ) rotational surfaces of these lines, further information is required for the exact positioning in stream-wise and circumferential direction.

4.1. Stacking via reference point

If that option is used, the hub profile P_0 is transformed first, followed by the remaining (non-hub) profiles which are thereafter located relative to the hub profile. The stream wise information for the hub profile P_0 is usually defined via a single, individual value for either x or m and the indication of a specific profile reference point (either leading edge, trailing edge or center of mass). This reference point is placed at the given value in stream-wise direction. This enables the unambiguous determination of the (x, r)-coordinates on the hub construction line (see Fig. 1) based on the m' or m coordinates of the two-dimensional hub profile P_0, allowing the transformation of all control points to be performed:

$$(m', \theta) \longrightarrow (x, r, \theta) \longrightarrow (x, y, z) = (x, r\sin(\theta), r\cos(\theta)) \tag{4}$$

If the stacking law e.g uses the center of mass of each profile as the reference point, the following two variations remain for the non-hub profiles:

1. Positioning via spatial coordinates: The non-hub profile P_k is positioned on its construction surface in such a way that the reference point (e.g. center of gravity) coordinates (x, θ) correspond to those of the P_0 reference point.
2. Positioning via closest distance: The non-hub profile P_k is positioned on its construction surface in such a way that the reference point (e.g. center of gravity) coordinates (x, r) minimize the distance to those of the reference point of profile P_{k-1}.

The second option is needed e.g. for radial impellers with steep or even pure radial outlets, where the point on the construction spline is not unique for a given x value.

After the profiles have been stacked initially by the stacking law (e.g. center of mass), the default stream-wise, circumferential or chord-wise shift of each profile, defined by the respective procedures described above, can be varied by user input to enable bowed, leaned or S-shaped blades.

4.2. Prescribed radial edges

Another option to determine the axial chord and stream-wise shift of each profile is to prescribe leading and trailing edge curves in (x, r) coordinates (see e.g. blue lines notated as LE and TE in Fig. 1). With these edge curves (noted as $C_{\mathrm{LE}}(v)$ and $C_{\mathrm{TE}}(v)$ with $v \in [0, 1]$), the stream-wise positioning of each profile is determined individually by the intersection points of the edge curves with the corresponding construction curves. The parameterization of an edge curve is realized by discrete points, which are interpolated via B-splines and some parameters of global influence to realize shifting and skewing with just a few design parameters. This option is often used because of the

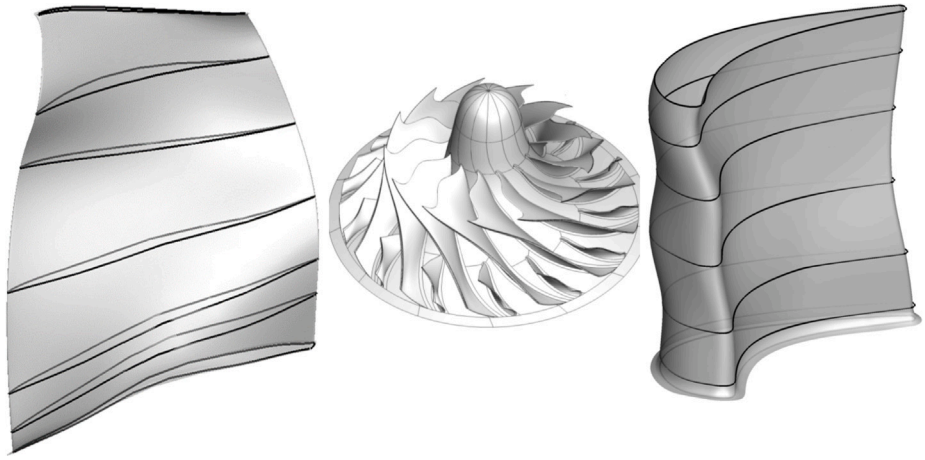


Fig. 7. Example of three-dimensional blade shape with stacked Profiles for axial compressor (left) and axial turbine (right). Radial impeller with main and splitter blades in full annulus view (middle).

intuitive blade-surface shape control. One should note, that in this case the axial chord length c_{ax} (see Fig. 2) is already determined, and cannot be set independently.

5. Skinning of the three-dimensional blade surface

In order to use the parameterized blade in the subsequent analysis and optimization process, the blade needs to be converted into a B-spline surface, which is a common generic surface parameterization used in computer aided design. Skinning is a process of blending the profile curves together to form a B-spline or NURBS surface (see [12]). The blend direction in BladeGen is the radial direction from hub to tip. Although approximation across the section curves could be used, skinning methods in BladeGen interpolate the profile curves, with the result that the profiles are isoparametric curves $\vec{C}(u) = \vec{S}_{init}(u, v_0), v_0 \in [0, 1]$ on the resulting skinned surface $\vec{S}_{init}(u, v)$, as shown in Fig. 7. Before the skinning routine can be applied to the stacked profiles, adjustments have to be made in order to fulfill the compatibility conditions for the construction profiles, i.e. all profiles require the same degree and the same knot vector (see [12] for details).

After performing this standard skinning routine, an initial three-dimensional B-spline surface $\vec{S}_{\mathrm{init}}(u, v)$ is defined by given spline degrees in the two surface directions p and q, knot vectors $\vec{\tau}_u$ and $\vec{\tau}_v$ and a three dimensional control net $\vec{P}_{i,j}$:

$$\vec{S}_{\mathrm{init}}(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u, \vec{\tau}_u) N_{j,q}(v, \vec{\tau}_v) \vec{P}_{i,j}, \tag{5}$$

where $\{N_{i,d}(u, \vec{\tau})\}$ are the corresponding B-spline basis functions of degree d.

It is now possible to use this surface as a basis for further modifications (see Section 6), or to use the procedures described in the following sub-sections to create such a basis surface.

5.1. High-resolution radial stacking

After the initial surface creation (see Eq. (5)), it is still possible to decouple the complexity of the radial design from the number of profiles. For example: bowed or leaned blades can be created although only a few (e.g 2 or 3) construction profiles are available. This is achieved by refining the control net of the skinned surface in radial direction by performing knot insertion in v-direction (see [12]), increasing m in Eq. (5). The three-dimensional shifts $\vec{s}$ in stream-wise, circumferential or chord-wise direction (to create bow, lean, S-shape etc.) are conducted with the refined surface control net $\vec{P}^{\mathrm{refined}}_{i,j}$, which of course approximates the actual geometry the better, the finer it is resolved. For each fixed row $j^* \in \{0, \dots, m\}$, which determines a relative radial height $h_{j^*} \in [0, 1]$ in the control net, the corresponding stream-wise control points $P^{\mathrm{refined}}_{i,j^*}, i \in \{0, \dots, n\}$ are all shifted by the same vector $\vec{s}$, which can defined by the user as a function $\vec{s}(h)$ of relative height:

$$\vec{P}^{\mathrm{new}}_{i,j^*} = \vec{P}^{\mathrm{refined}}_{i,j^*} + \vec{s}(h_{j^*}), \forall i \in \{0, \dots, n\} \tag{6}$$

5.2. Surface-creation with prescribed leading and trailing edges

The same approach of radial knot refinement and subsequent adaptation of the control net is also used for the prescribed leading and trailing edges mentioned before (see Section 4.2), so that the three-dimensional blade shape follows the prescribed edges shape in (x, r) coordinates, even if only a few construction profiles are used in combination with complex curved edges.

In order to suitably modify the control net, to meet the prescribed radial edge curves, the required displacement vectors $\vec{D}_{\mathrm{LE,j^*}}$ at the leading and $\vec{D}_{\mathrm{TE,j^*}}$ at the trailing edge are first determined for a given $j^* \sim h_{j^*}$ of the control net and a given i_{LE,j^*} resp. i_{TE,j^*}, which correspond to the control points indices at leading and trailing edge. The direction of the normalized vectors $\vec{D}_{\mathrm{LE},j^*}$ and $\vec{D}_{\mathrm{TE},j^*}$ is defined by the direction of the chord (at LE resp. TE). The position of the leading and trailing edge control points is thus defined by the scaling parameters λ and μ, which are determined by the following equations for each row j^*:

$$\exists v_{\mathrm{LE},j^*} \in [0, 1] \wedge \lambda \in \mathbb{R} : (\vec{P}_{i_{\mathrm{LE}},j^*} + \lambda \vec{D}_{\mathrm{LE},j^*})_{x,r} = C_{\mathrm{LE}}(v_{\mathrm{LE},j^*}) \tag{7}$$

$$\exists v_{\mathrm{TE},j^*} \in [0, 1] \wedge \mu \in \mathbb{R} : (\vec{P}_{i_{\mathrm{TE}},j^*} + \mu \vec{D}_{\mathrm{TE},j^*})_{x,r} = C_{\mathrm{TE}}(v_{\mathrm{TE},j^*}) \tag{8}$$

Here, $(..)_{x,r}$ denotes the transformation to (x, r) coordinates and the above equations just imply, that the edge control points of the surface are shifted onto the edge curves in (x, r) coordinates. All other (non-edge) control points of the same radial height are shifted via a distance-dependent transfer function. In BladeGen the default transfer function is defined by:

$$f_{j^*}(i) = \max\left(\frac{\|\vec{P}_{i_{\mathrm{LE}},j^*} - \vec{P}_{i,j^*}\|}{\|\vec{P}_{i_{\mathrm{LE}},j^*} - \vec{P}_{i_{\mathrm{TE}},j^*}\|}, 1 \right), \text{ with } f_{j^*}(i_{\mathrm{LE}}) = 0,\ f_{j^*}(i_{\mathrm{TE}}) = 1 \tag{9}$$

Some other transfer functions are also available in BladeGen (especially for radial impellers), but they are only of minor importance. The final shifts for all control points are determined by:

$$\vec{D}_{i,j^*} = \lambda \vec{D}_{\mathrm{LE},j^*} \left[1 - f_{j^*}(i)\right] + \mu \vec{D}_{\mathrm{TE},j^*} f_{j^*}(i) \tag{10}$$

$$\vec{P}^{\text{new}}_{i,j^*} = \vec{P}^{\text{refined}}_{i,j^*} + \vec{D}_{i,j^*}, \forall i \in \{0,\dots,n\} \tag{11}$$

An alternative method, that first seems to be applicable to fulfill the geometric requirements, such that the resulting surface matches the blade profiles and the leading and trailing edges is the interpolation method of a bidirectional curve network. In this case, the curves in u-direction $C_i(u)$ are the stacked profiles and two curves in v-direction $D_{1/2}(v)$ are defined by the prescribed edge curves (see Fig. 1). Such general methods for bidirectional curve network interpolation like Gordon surfaces (see [13]), or Coons surfaces (see [14]) require quite strong prior conditions which are not fulfilled here: The three-dimensional curve network must satisfy stronger compatibility conditions than before: the curves must be defined in the three-dimensional space and need to intersect mathematically exactly at given (u^*, v^*) for all curves, i.e. the curve network must consist of iso-parametric curves of an interpolation surface. A detailed derivation of the compatibility conditions and a potential solution by the reparameterization of the curve network is presented in [15]. In our case the edge curves are just two dimensional and even an imaginable extension to 3 dimensions would not guarantee a mathematically exact intersection with the profiles. Thus, the presented approach, to use a refined control net for radial edge modification, seems to be reasonable and coincides with the procedure described above via Eq. (6) and in the following fillet Section 6.2. This approach described here can thus be seen as a NURBS surface approximation method for bidirectional curve networks.

6. Three-dimensional blade modifications

With the methods described above, a three-dimensional B-spline surface $\vec{S}(u,v)$ according to Eq. (5) is now defined either for compressor- or for turbine-blades in axial or radial design. This already contains the main essential design features like bowed/leaned blades or S-shaped edges. The additional modifications in the following sub-sections are rather to be regarded as optional design features.

Note that the blade's B-spline surface $\vec{S}(u,v)$ has a different parameterization range in u (responsible for stream-wise isoparametric curves around the blade) and v (responsible for radial isoparametric curves from hub to tip):

$$u \in [0,4], \quad v \in [0,1] \tag{12}$$

While $\vec{S}(u^*,0)$ defines a point on the hub profile and $\vec{S}(u^*,1)$ defines a point on the tip profile, the behavior of the isoparametric curves around the blade are defined as follows (visualized in Fig. 8):

- $\vec{S}(0,v^*)$ defines the connection point between trailing edge and pressure side.
- $\vec{S}(1,v^*)$ defines the connection point between pressure side and leading edge.
- $\vec{S}(1.5,v^*)$ defines the leading edge.
- $\vec{S}(2,v^*)$ defines the connection point between leading edge and suction side.
- $\vec{S}(3,v^*)$ defines the connection point between suction side and trailing edge.
- $\vec{S}(3.5,v^*)$ defines the trailing edge.
- $\vec{S}(4,v^*) = \vec{S}(0,v^*), \forall v^* \in [0,1]$.

This information and the associated possibility of influencing special regions on the blade is used in the following sub-sections.

6.1. Cuttings

There are two different cutting procedures available in BladeGen. The first one cuts the blade surface at hub and/or tip and the second one cuts the trailing edge of radial impellers at constant radius.

6.1.1. Cutting hub/tip region of blades

This option is often used during automated optimization runs, if the endwalls are free to be optimized. In these cases, it makes sense to design a blade on static construction curves that protrude beyond the actual flowpath. The blade endwall regions are then cut by a dynamically changeable flow path, which is defined in (x,r)-coordinates. On the left of Fig. 9, a blade surface (black mesh) is shown which is constructed with 7 profiles (pink) on construction curves defined by the red mesh (for $k \in \{1,2,3,5,7,9,11\}$). On the right of Fig. 9, the hub and tip regions of the blade are cut via the curves $k = 3$ and $k = 9$. Of course, the cutting curves can have any shape and do not have to correspond to construction curves or isoparametric curves on the blade as long as they cover the entire blade from leading to trailing edge. The method used for blade-cutting is quite simple: For a user-specified number of points on the blade sides N_s and blade edges N_e, isoparametric curves $\vec{C}_{radial}(v) = \vec{S}(u^*,v),\ u^* \in [0,4]$ are extracted on the blade. On these curves, the minimum v_{min} and maximum v_{max} values are determined such that $\vec{C}_{radial}(v_{min})$ transformed to (x,r) is on the hub cutting curve and $\vec{C}_{radial}(v_{max})$ transformed to (x,r) is on the tip cutting curve. If e.g. only tip-cutting is to be done: $v_{min} = 0$.

Then a user-specified number of radial points N_r is distributed on $\vec{C}_{radial}(v)$ between v_{min} and v_{max} by a given arc-length distribution. In this way a regular mesh of size: $2 * (N_s + N_e) \times N_r$ is created by sampling the surface. Thereafter this regular mesh is interpolated by a B-spline surface (standard procedure in [12]). The rearrangement of the knot vector $\vec{\tau}_u$ according to Fig. 8 is subsequently carried out by a decomposition into Bézier patches and subsequent joining in the order shown in Fig. 8.

6.1.2. Trailing edge cutting for radial impellers

A common procedure for the dimensioning of a radial impeller is to cut the trailing edge at constant radius. This option is also available in BladeGen. The left side of Fig. 10 shows an example where an initial radial impeller (gray) is cut at r=0.1 (red). The connection between suction side and pressure side at the trailing edge is linear, thus in that case the final shape has edges/kinks and is not C^1 continuous anymore at the connection points from trailing edge to pressure/suction side (see Fig. 10 rhs). The mathematical method used here is similar to the one described above in Section 6.1.1 (i.e. a surface interpolation of a regular mesh) apart from the final multiplicity of the knot vector $\vec{\tau}_u$ at the blunt trailing edge region, which is set to 3 to allow discontinuous first derivatives/kinks.

6.2. Fillets

A fillet is the smooth connection between the surface of the blade and the end walls at the hub and/or tip. It is mainly necessary to avoid mechanical stresses like notch tension in these regions but has also an aerodynamic impact. In BladeGen, the fillet is constructed as an integral part of the blade surface and not as a separate component. The idea of fillet creation is shown in Fig. 11. For a fillet the initial profile at the end wall (blue line in Fig. 11) is blown up to create the new boundary curve of the fillet on the hub surface (green line in Fig. 11). A second curve (red line in right part of Fig. 11) defines the radial end of the fillet, where the fillet is smoothly and C^2-continuously connected to the rest of the blade. These two curves are connected by the fillet surface and the connection is defined by a third curve which defines the radial transition. As already mentioned at the end of Section 5, all transformations are actually performed on the blade surface control net, which has been previously refined radially in the fillet regions to ensure a good approximation of the final surface. Fig. 12 shows a detail view of the resulting control net for a blade with a hub fillet.

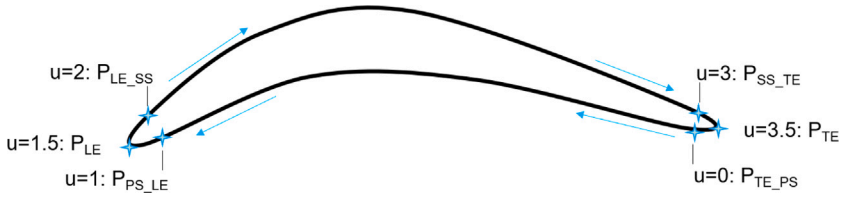


Fig. 8. Isoparametric curve around the blade with characteristic points.

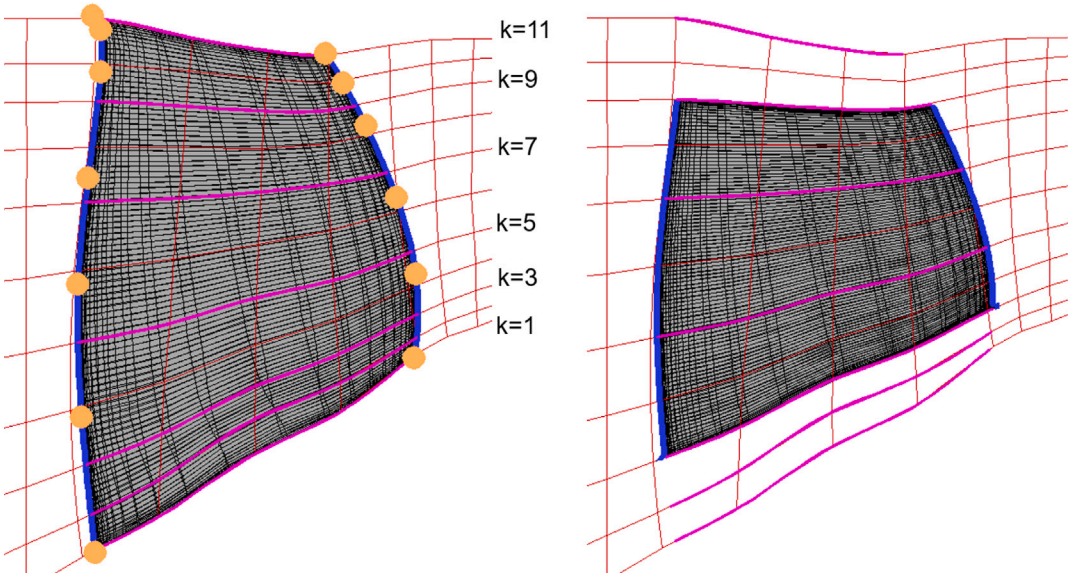


Fig. 9. Original blade (left) and cut blade at hub and tip (right) constructed via 7 profiles (pink) and prescribed leading/trailing edges (blue lines).

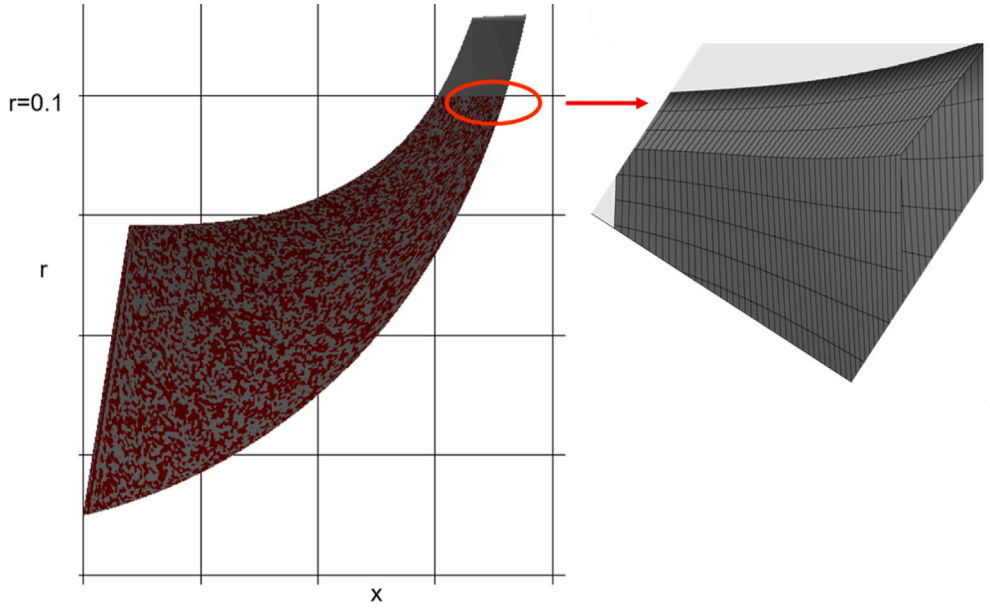


Fig. 10. Cut trailing edge at $r = 0.1$ for a radial impeller.

7. Application

Turbomachinery blade design often demands a delicate balance between aerodynamic performance, manufacturability, and structural integrity. The proposed engineering-based parameterization provides a practical and intuitive framework by leveraging classical design parameters such as angles, thicknesses, lengths, and radii. This approach enables designers to make straightforward adjustments that are easy to interpret and align seamlessly with traditional design practices, streamlining the optimization process.

In contrast to Free-Form Deformation (FFD) methods, which allow unconstrained geometry modifications, the engineering parameterization preserves interpretability and practicality. Alternative methods, including B-spline parameterization, modal decomposition, and proper

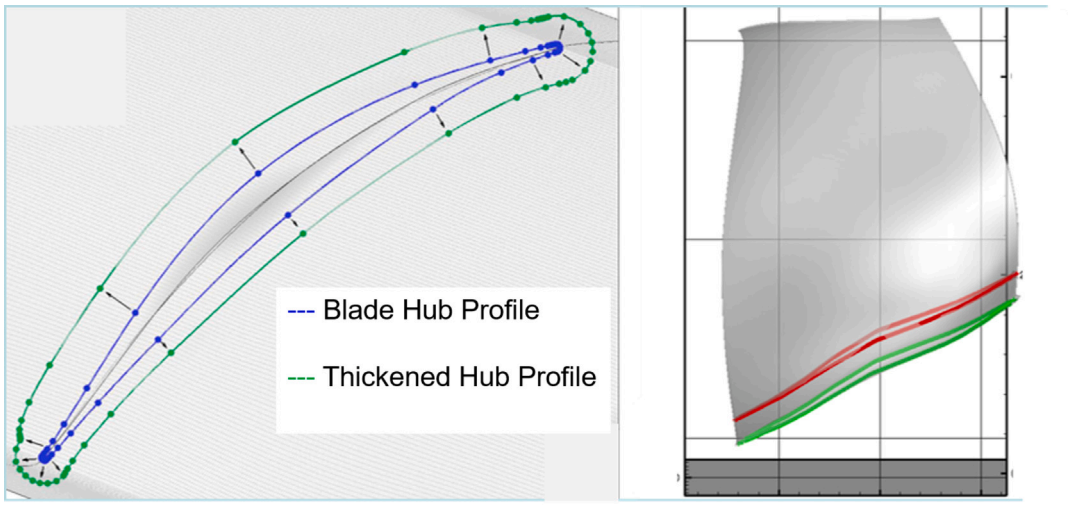


Fig. 11. Fillet parameterization: the three-dimensional edge curves of the fillet depicted in green and red.

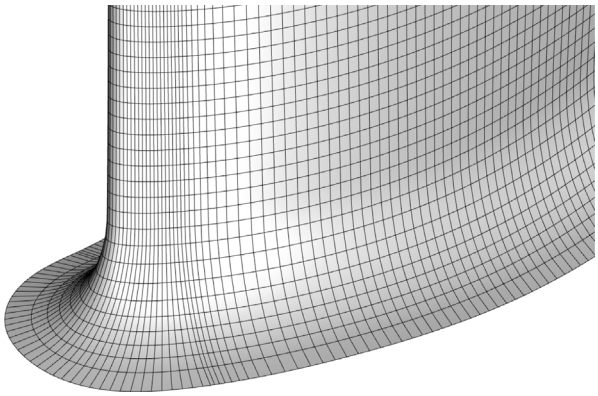


Fig. 12. Mesh of control points of a B-spline blade surface with a fillet.

orthogonal decomposition (POD), offer advanced control over geometric variations and can capture intricate design features. However, these approaches often necessitate specialized expertise to evaluate their aerodynamic and structural impacts, introducing additional complexity to the design workflow.

To illustrate the benefits of the parameterization, a very simple example of initial airfoil geometry is considered in Fig. 13(a). Assuming the inflow conditions of this airfoil are changed, such that the inflow angle is reduced by five degrees, the airfoil is now operating under an incidence (Fig. 13(b)). CFD simulations reveal excessive flow separation on the pressure side and a suboptimal position of the stagnation point. To correct the flow around the leading edge, the airfoil is re-staggerd by five degrees (Fig. 13(c)), removing the incidence. This intuitive adjustment is an example of how designers can quickly improve airfoil performance through the variation of just a single parameter. To also adjust the turning and outflow angle further adjustments have to be made.

A second scenario involves a multi-objective optimization problem of this airfoil aimed at minimizing aerodynamic losses at design point (objective 1) and off-design (objective 2), computed as an average at plus and minus 5 degrees incidence. Using the same parameterization, an optimization tool based on a genetic algorithm optimizes the geometries, producing a Pareto front. Analysis of the Pareto-optimal solutions reveals that parameters like stagger angle significantly influence movement along the Pareto front, while parameters like leading-edge radius are critical for reaching the front (see Figs. 14(b) and 14(c)).

This analysis demonstrates how the engineering parameterization facilitates deeper understanding of design trade-offs, empowering designers to make informed decisions.

8. Software architecture

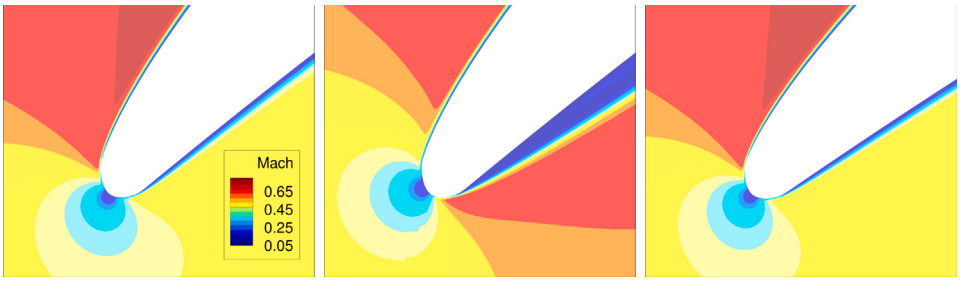
BladeGen has a flexible architecture that allows for integration into a wide range of workflows and use cases. The BladeGen system consists of five main components: BladeGen's core library, a library for the geometry modeling of NURBS, the standalone BladeGenerator executable, the C++ bindings, and the Python bindings. A high-level block diagram of BladeGen's software components is provided in Fig. 15.

BladeGen's core library. The core of BladeGen is a C library that implements the algorithms and data structures necessary for converting the BladeGen parameterization into a three-dimensional B-spline surface of the blade geometry as described in the previous sections. This library provides the underlying foundation for the entire BladeGen system.

NURBS library. BladeGen also includes an in-house developed geometry library to model B-spline and NURBS curves and surfaces. It provides standard algorithms such as knot insertion, interpolation of points, or surface skinning, similar to the methods described in [12]. This library allows for the creation and manipulation of complex geometries, providing a versatile tool for generating different geometries such as blades.

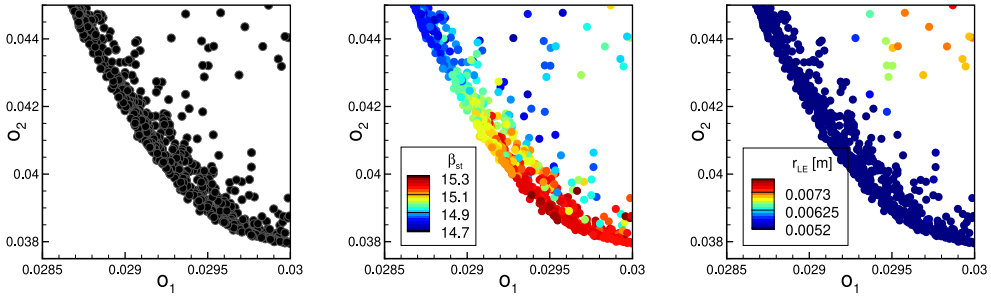
Standalone executable. In addition to the C library and geometry library, BladeGen is also available as a standalone executable. This allows users to run BladeGen as a standalone application, making it easily accessible and it is in fact the default way of using BladeGen today by the users. The design parameters of the blade need to be provided using multiple input files by the user.

C++ bindings. To enable integration with other software tools and workflows, BladeGen provides C++ bindings that give a direct access to the BladeGen's algorithms and data structures. The C++ bindings are the fastest option for generating blade geometries and are easy to integrate into existing workflows. The C++ bindings encapsulate the parameterization of the blade and its profiles and separate it from the geometry generation process using an object oriented interface. The C++ bindings allow for a direct integration of BladeGen into



(a) Initial Airfoil (b) Changed inflow conditions (c) Adjusted airfoil

Fig. 13. Mach contour plots of an airfoil under initial inflow conditions, changed inflow conditions, and with adjusted stagger angle.



(a) Optimization database (b) Stagger angle (c) LE radius

Fig. 14. Optimization database with stagger angle and leading edge radius highlighted to show variation regarding the two objectives.

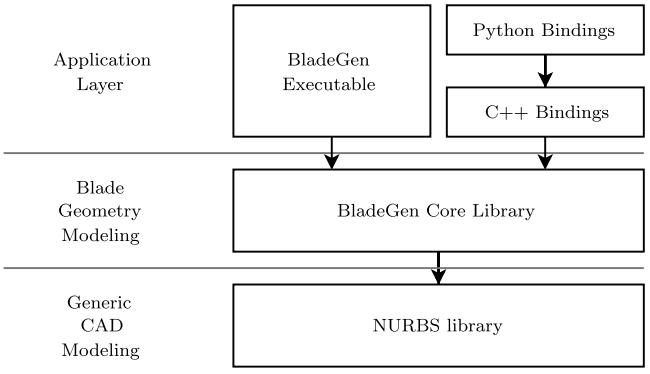


Fig. 15. The software architecture of BladeGen.

other software tools and workflows, eliminating the need for a slow file I/O when using the BladeGen executable instead. In particular, BladeGen was integrated into the DLR's engine design environment GTlab (see [10]) in this way, a software tool for simulating, designing and optimizing gas turbine systems.

Python bindings. BladeGen also has Python bindings that provide another easy access to its functionality by tool developers. The Python bindings are based on the C++ bindings and are created with `pybind11` (see [16]). This allows to integrate BladeGen into existing Python workflows, enables geometry scripting, and offers an interactive exploration of the different geometry parameters of BladeGen. The Python bindings thus provide a convenient and flexible way to use BladeGen's capabilities within the Python ecosystem.

Interfacing OpenCASCADE. BladeGen also includes the option to interface with the OpenCASCADE Technology (see [17]), which is a widely used open source CAD framework, allowing a further modification of the BladeGen-generated geometries by the users. OpenCASCADE is used to export the blade geometry into standard CAD export file formats such as Step, IGES or BRep as well as to estimate physical properties of the blade, such as mass, center of gravity, and the main axes of inertia.

To incorporate the blade design process into the overall turbomachine design workflow, BladeGen has been integrated into the design environment GTlab [10], as shown in Fig. 16.

9. Algorithmic differentiation

BladeGen has been differentiated by means of algorithmic differentiation (AD) (see [18]). Precise derivatives, without the need of tuning sensitive step-widths, of the final shape with respect to the design parameters are essential for gradient based shape optimization. This is reflected in by reports about differentiation of CAD-Kernels by means of algorithmic differentiation [19,20] or the application of the complex-step method to free form deformation (FFD) of turbomachinery blades [21]. Using algorithmic differentiation, these derivatives can be computed accurately without the problems of finite differences that require tuning of a differentiation step width. Furthermore, algorithmic differentiation offers two modes for computing derivatives: The forward mode computes derivatives of arbitrarily many outputs with respect to one input, while the reverse mode calculates the derivative of a single output with respect to arbitrarily many inputs.

We denote BladeGen as a vector function $\mathbf{y} = f(\boldsymbol{\alpha})$ that maps design parameters $\boldsymbol{\alpha}$ to points $\mathbf{y}$ on the blade surface in three dimensional space. Then the forward mode reads as a directional derivative

$$\dot{\mathbf{y}} = \frac{\partial f}{\partial \alpha}\dot{\boldsymbol{\alpha}} \tag{13}$$

where $\dot{\boldsymbol{\alpha}}$ is the direction of differentiation.

Differentiating all blade surface point coordinates with respect to one design parameter can be achieved by choosing $\dot{\boldsymbol{\alpha}} = \mathbf{e}_i$, the ith unit-vector.

For the reverse mode of AD consider a function that computes a scalar valued output quantity $z = g(y)$ from the blade surface. Think of e.g. blade volume or aerodynamic properties obtained from

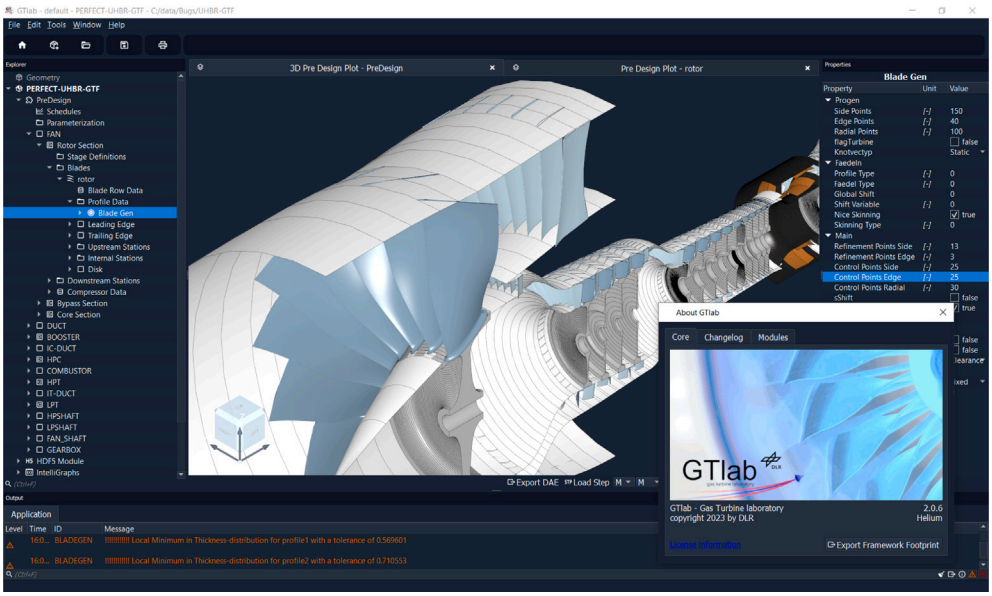


Fig. 16. Screenshot of BladeGen integrated into the engine design platform GTlab, in particular parts of the fan blade parameterization.

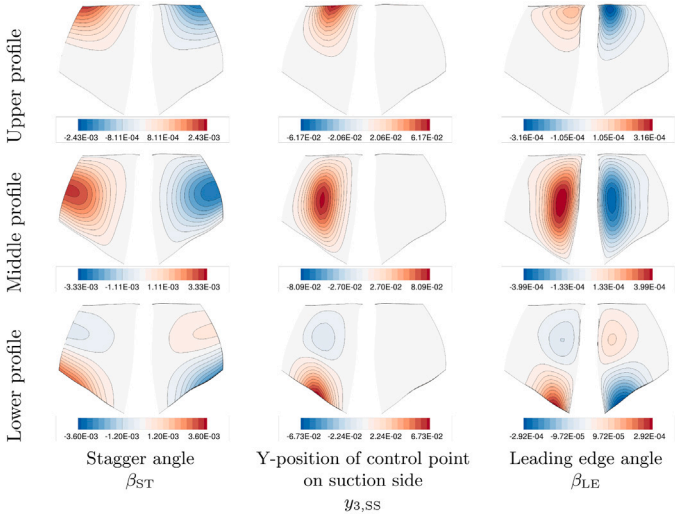


Fig. 17. Derivatives of blade surface points with respect to design parameters on the three construction profiles, projected on the surface normals.

a CFD calculation. Since $z = g(f(\alpha))$ applying the chain rule and then transposing both sides of the equation one obtains

$$\frac{\partial z}{\partial \alpha}^T = \frac{\partial f}{\partial \alpha}^T \frac{\partial g}{\partial y}^T \tag{14}$$

which is the gradient of z with respect to all input variables. The equation is evaluated from right to left. First the surface sensitivities of the output quantity $\frac{\partial g}{\partial y}^T$ are calculated, e.g. by reverse differentiation of the implementation of g or an adjoint simulation. This vector is propagated through the reverse differentiated of blade-generator to obtain the gradient of the output quantity with respect to an arbitrarily large number of blade design variables.

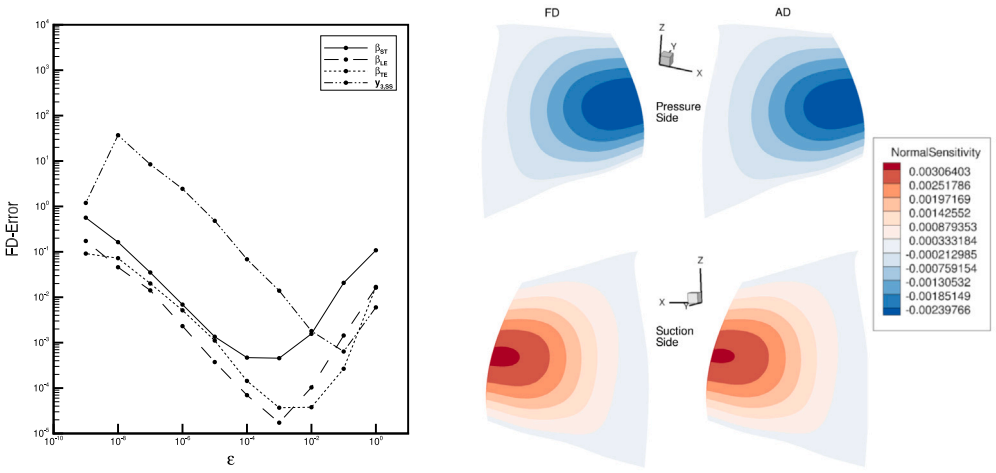
Note, that on the right hand side of Eq. (13) and (14) the matrix vector product is evaluated without explicitly computing the Jacobian $\frac{\partial f}{\partial \alpha}$.

Both modes of algorithmic differentiation have been implemented for BladeGen by using ADOL-C (see [22]). Only the core functionality implemented in C++ had to be differentiated, the python interface, graphical user interface and output routines used from OpenCASCADE did not need to be differentiated. ADOL-C implements the operator overloading approach, where all floating point variables are changed to a type, which re-defines the algorithmic operations so that derivatives can be computed for each elementary operation $(+, -, *, /, \sin, \ldots)$. These constitute, by the chain rule of differentiation, the derivative of the program.

Fig. 17 shows exemplary results obtained from forward differentiation. These are sensitivities of each surface node with respect to design parameters at the three different construction profiles. As each surface node has three coordinates, the derivatives are vectors which are projected on the surface normal of the blade $\mathbf{n}$ to obtain a scalar representation $\dot{\mathbf{y}} \cdot \mathbf{n}/\|\mathbf{n}\|$. This can help understanding the influence of parameters on the blade surface. These are generated by using the traceless forward mode.

Fig. 18 shows the validation of the forward mode against finite differences for parameters on the middle profile. A variation of the finite difference step size is shown in Fig. 18(a). The error is calculated as the square-norm of the difference between the sensitivities projected on the surface normal

$$e_i = \left\| \frac{f(\boldsymbol{\alpha} + \varepsilon \alpha_i) - f(\boldsymbol{\alpha})}{\varepsilon \alpha_i} \cdot \frac{\mathbf{n}}{\|\mathbf{n}\|} - \dot{\mathbf{y}}_{\mathrm{AD}} \cdot \frac{\mathbf{n}}{\|\mathbf{n}\|} \right\|_2 . \tag{15}$$



(a) Step size variation (b) Comparison for β_{ST}, middle airfoil, $\varepsilon = 10^{-2}$

Fig. 18. Validation of forward AD with finite differences.

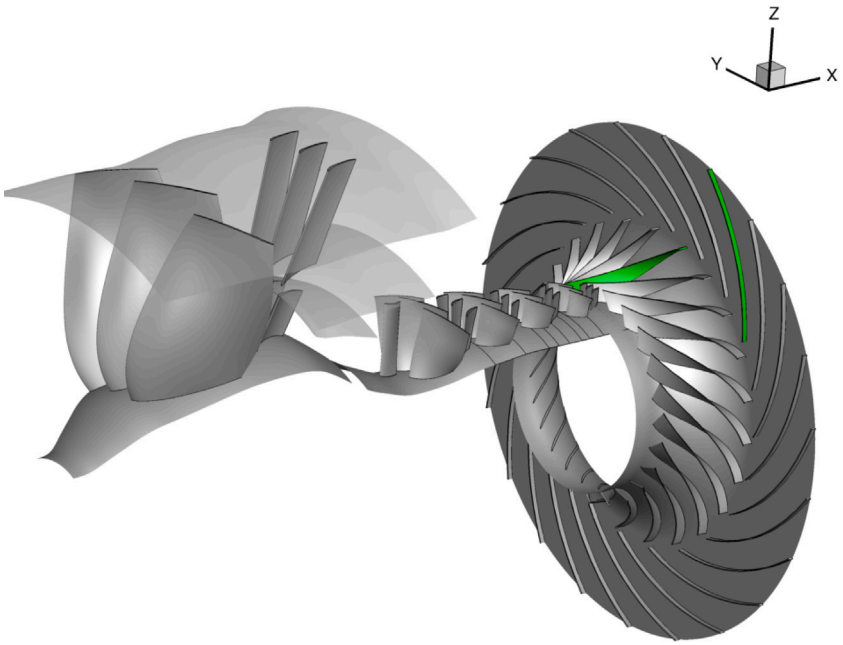


Fig. 19. BladeGen Design with radial stage similar to the compression system of the PW308c engine.

For each parameter there exists a step size that minimizes the finite differences error (c.f. Fig. 18(a)). Larger ε produce nonlinear behavior, smaller ε lead to precision loss due to subtractive cancellation. Since the optimal step width differs between parameters Fig. 18(a) illustrates that the tuning of step-widths for finite differences can be quite intricate even when a visual comparison does not show recognizable differences. To this end, Fig. 18(b) shows a visual comparison of the surface sensitivities between FD and AD and corresponds to the solid line in Fig. 18(a) at $\varepsilon = 10^{-2}$. It illustrates how an error of that norm is perceived when comparing contour plots.

Detailed benchmarks of the differentiated software, concerning time and memory consumption, have not been conducted since many different aspects would have to be considered for a thorough study, especially when comparing to finite differences. Reverse mode differentiation certainly increases execution time by an estimated factor of 5 to 10 depending on the test case. Main memory consumption also increases. Both could be brought down by introducing optimizations to the AD implementation. FD execution time depends on the number of design parameters which is usually not less than 20 for 3D designs. On the other hand FD need more hard disk storage for intermediate results. However the absolute numbers are very much negligible considering the run time of the whole design evaluation chain and the available cpu and memory on recent hardware. The ability to calculate derivatives without step-width related errors is what warrants the effort of applying algorithmic differentiation to BladeGen. More importantly, it adds the capability to use BladeGen's geometry generation in adjoint process

chains. This allows efficient computation of gradients in a single sweep with respect to an arbitrary number of design parameters, enabling large-scale gradient-based design optimizations.

10. Conclusion and outlook

In this section one exemplary design example is shown to demonstrate the wide range of applications of the blade parameterization presented. BladeGen is the standard method of blade parameterization for turbomachinery applications at DLR. Therefore, there is a large number of different blades that have been designed with the method presented here: rotors, stators, IGV's, wedge-diffusers in axial and radial design for fans, compressors, turbines in different applications like aircraft engines (civil and military), stationary gas turbines or turbo components in other energy processes like heat pumps (see eg. [6]). As not all of this work can be shown within the scope of this paper, the following will concentrate on the compression system of a single engine, which contains some of the most important blade types.

Fig. 19 shows the fan and compression system of an aero engine similar to PW308c engine (see [23] for more details) that powers DLR's Falcon 2000LX research aircraft. The engine is a two-shaft design with a single stage fan and a three stage low pressure turbine mounted on the low-pressure shaft. On the high pressure shaft, the five stage axial/centrifugal compressor is driven by a two-stage high-pressure turbine. All blades and vanes including the axial and radial components (radial impeller and diffuser in green) are designed by BladeGen parameterization. For the 13 blade rows of the compression system shown in Fig. 19, a parameterization with approximately

$$780 = 13\,(\text{blade rows}) \cdot 3\,(\text{airfoils}) \cdot 20\,(\text{profile parameters})$$

parameters is required to enable a customized solution for all multidisciplinary requirements. Geometry generation takes approximately 5 s per blade row if the call is controlled from the GTLab pre-design environment with integrated data management, thus eliminating time-critical I/O (see Section 8 and [10]). Of course, both the computer architecture and the depth of the parameterization (e.g. number of stacked airfoils, fillets) will affect the exact processing time.

Beside the wide range of manual use in various national and European multi-disciplinary turbo-machinery design projects, BladeGen has proven its suitability and robustness through its millions of calls within automated optimizations. With the relatively new possibilities given by AD (see Section 9), gradient-based methods can extend the established surrogate based optimization methods in the future, avoiding the numerical bottleneck of finite differences in the process chain.

The also relatively new possibilities of BladeGen integration into C++ environments and Python workflows, in combination with an enhanced interface with the OpenCASCADE framework (see Section 8), will allow even more flexibility in future geometry parameterization.

CRediT authorship contribution statement

Christian Voß: Writing – review & editing, Writing – original draft, Supervision, Software, Project administration, Methodology, Conceptualization. **Martin Siggel:** Writing – review & editing, Writing – original draft, Software. **Jan Backhaus:** Writing – review & editing, Writing – original draft, Software. **Georgios Goinis:** Conceptualization, Validation, Writing – review & editing. **Andreas Pahs:** Writing – original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] Dunham J. A parametric method of turbine blade profile design. ASME International Gas Turbine Conference and Products Show J 1974. http://dx.doi.org/10.1115/74-GT-119.

[2] Moran J. An introduction to theoretical and computational aerodynamics. John Wiley & Son; 1984.

[3] Agromayor R, Anand N, Müller J, Pini M, Nord L. A unified geometry parametrization method for turbomachinery blades. Computer- Aided Des 2021. http://dx.doi.org/10.1016/j.cad.2020.102987.

[4] Torreguitart IS, Verstraete T, Mueller L. Optimization of the LS89 axial turbine profile using a CAD and adjoint based approach. Int J Turbomach Propuls Power 2018. http://dx.doi.org/10.3390/ijtpp3030020.

[5] Voß C, Aulich M, Kaplan B, Nicke E. Metamodel assisted aeromechanical optimization of a transonic centrifugal compressor. ISROMAC 15 2014.

[6] Schaffrath R, Nicke E, Forsthofer N, Kunc O, Voß C. Gradient-free aerodynamic optimization with structural constraints and surge line control for radial compressor stage. ASME Turbo Expo 2023. http://dx.doi.org/10.1115/1.4067687.

[7] Karimian A, Schmidt R, Janke C. Robust optimization of a compressor blade through combination of an adjoint-based multistart approach and global Gaussian process regression. ASME Turbo Expo 2024. http://dx.doi.org/10.1115/GT2024-121351.

[8] John A, Shahpar S, Qin N. Novel compressor blade shaping through a free-form method. Int J Turbomach Propuls Power 2016. http://dx.doi.org/10.1115/1.4035833.

[9] Voß C, Aulich M, Kaplan B, Nicke E. Automated multiobjective optimisation in axial compressor blade design. ASME Turbo Expo 2006. http://dx.doi.org/10.1115/GT2006-90420.

[10] Reitenbach S, Vieweg M, Becker R, Hollmann C, Wolters F, Schmeink J, Otten T, Siggel M. Collaborative aircraft engine preliminary design using a virtual engine platform, part A: Architecture and methodology. In: AIAA SciTech 2020 Forum. 2020, p. 0867. http://dx.doi.org/10.2514/6.2020-0867.

[11] Drela M, Youngren H. A user's guide to mises. MIT Comput Aerosp Sci Lab 1989.

[12] Piegl L, Tiller W. The NURBS book. In: Monographs in visual communication, Berlin Heidelberg New York: Springer; 1997.

[13] Gordon WJ. Spline blended surface interpolation through curve networks. J Math Mech 1969;18(10):931–52.

[14] Coons S. Surface for computer aided design of space forms. MIT Proj MAC 1967;TR-41.

[15] Siggel M, Kleinert J, Stollenwerk T, Maierl R. TiGL: an open source computational geometry library for parametric aircraft design. Math Comput Sci 2019;13(3):367–89. http://dx.doi.org/10.1007/s11786-019-00401-y.

[16] Jakob W, Rhinelander J, Moldovan D. pybind11–Seamless operability between C++ 11 and Python, Accessed: 2023-02-06. https://github.com/pybind/pybind11.

[17] Open CASCADE Technology, Accessed: 2023-02-06. https://www.opencascade.com/open-cascade-technology/.

[18] Griewank A, Walther A. Evaluating derivatives: principles and techniques of algorithmic differentiation. Philadelphia: Society for Industrial and Applied Mathematics; 2008.

[19] Torreguitart IS, Verstraete T, Mueller L. Cad kernel and grid generation algorithmic differentiation for turbomachinery adjoint optimization. In: ECCOMAS congress 2016, VII European congress on computational methods in applied sciences and engineering. 2016.

[20] Banović M, Mykhaskiv O, Auriemma S, Walther A, Legrand H, Müller J-D. Algorithmic differentiation of the open CASCADE technology CAD kernel and its coupling with an adjoint CFD solver. Optim Methods Softw 2018;33(4–6):813–28, https://10556788.2018.1431235.

[21] Kenway G, Kennedy G, Martins JRRA. A CAD-free approach to high-fidelity aerostructural optimization. In: 13th AIAA/ISSMO multidisciplinary analysis optimization conference. 2010, http://dx.doi.org/10.2514/6.2010-9231.

[22] Griewank A, Juedes D, Utke J. Algorithm 755: ADOL-C: A package for the automatic differentiation of algorithms written in C/C++. ACM Trans Math Softw (TOMS) 1996;22(2):131–67. http://dx.doi.org/10.1145/229473.229474.

[23] Wittrock D. Dimensionierung und modellierung von radialverdichtern im initialen triebwerksentwurf. Diss Ruhr- Universität Boch 2023.