Spline-Based Parameterization Techniques for Twin-Screw Machine Geometries

Hinz, Jochen; Möller, Matthias; Vuik, Cornelis

DOI
10.1088/1757-899X/425/1/012030

Publication date
2018

Document Version
Final published version

Published in
IOP Conference Series: Materials Science and Engineering

Spline-Based Parameterization Techniques for Twin-Screw Machine Geometries

IOP Conf. Series: Materials Science and Engineering **425** (2018) 012030 doi:10.1088/1757-899X/425/1/012030

Spline-Based Parameterization Techniques for Twin-Screw Machine Geometries

Jochen Hinz, Matthias Möller and Cornelis Vuik

Delft University of Technology, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft Institute of Applied Mathematics, Van Mourik Broekmanweg 6, 2628 XE Delft, The Netherlands

E-mail: J.P.Hinz@tudelft.nl, M.Moller@tudelft.nl and C.Vuik@tudelft.nl

Abstract.
The fully automated generation of computational meshes for twin-screw machine geometries constitutes a mandatory aspect for the numerical simulation (and shape-optimization) of these devices but proves to be a challenging task in practice. Therefore, the successful generation of computational meshes requires sophisticated mathematical tools. Commercially available classical mesh generators can produce high quality meshes from no more than a description of the rotor contours. However, since we are particularly interested in numerical simulations using the principles of *Isogeometric Analysis*, a spline-based geometry description rather than a classical mesh is needed.

In this paper, we propose a practical approach for the automated generation of spline-based twin-screw machine geometry parameterizations in two dimensions. For this purpose, we adopt the principles of *Elliptic Grid Generation* and present a parameterization algorithm that is compatible with an automated simulation pipeline based on the principles of isogeometric analysis.

To demonstrate the proposed techniques, we apply them to an example geometry and present the resulting parameterizations. Finally, we give a qualitative explanation of how the discussed techniques can be utilized to generate geometry parameterizations in three dimensions and their applications to shape-optimization on a variable rotor-pitch.

1. Introduction

The generation of analysis-suitable meshes for twin-screw geometries constitutes the first step towards the numerical simulation and shape-optimization of twin-screw machines. However, the full automation of this process remains difficult, often contributing substantially to the total amount of labour hours and computational costs. On the one hand, this is caused by the inherent difficulty of generating a mesh from no more than a description of the boundary contours, and, on the other hand, further aggravated by the challenging characteristics of twin-screw geometries, such as tiny clearances and rapidly changing gap sizes (see figure 1). To the best of our knowledge, there are two commercially available classical structured mesh generators that only require a description of the geometry contours as input: *twin-mesh* [1] and *SCORG* [2], both being capable of exporting directly into *ANSYS CFX* format [3]. We are particularly interested in using Isogeometric Analysis (IgA) [4] techniques to perform shape-optimization on twin-screw machine geometries with variable rotor pitch. Even though the commercial mesh generators produce high-quality classical meshes, they cannot be used in an IgA-setting where a

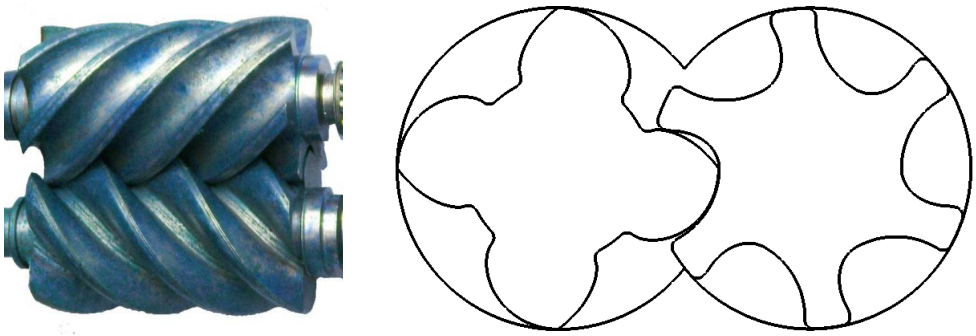


Figure 1. Rotary-screw compressor (Wikipedia, file: *Lysholm_screw_rotors.jpg*) (left) and a cross section showing the rotor profiles with casing (right).

spline-based parameterization of the target geometry is mandatory. This is the main motivation for the techniques presented in this paper.

The general idea of a spline-based parameterization is to build a continuous mapping operator $\mathbf{x} : \hat{\Omega} \rightarrow \Omega$ comprised of higher-order spline functions that maps the entire computational domain $\hat{\Omega}$ onto the target geometry Ω (or an approximation thereof). The spline-parameterization $\mathbf{x}$ is then directly used for an IgA-based simulation. Unlike in the classical case, application-specific features such as boundary layers are added after the geometry description has been completed. This is accomplished by performing knot-refinement on $\mathbf{x}$ (see section 2) and has a negligible impact on the overall computational costs.

A volumetric parameterization of the geometry is accomplished by generating a large number of planar parameterizations at various discrete rotational angles θ_i and stacking them in the z-direction. Hereby, it is desirable to achieve a degree of smoothness in the cell boundaries of the mesh corresponding to consecutive slices in order to achieve the same smoothness in the elements of the resulting volumetric parameterization. This translates to the requirement of smoothly varying control points of the mapping $\mathbf{x}$ as a function of the rotation angle θ. Therefore, it is furthermore desirable to, if possible, avoid topology changes in the planar slices, even though this may be a challenging task. Variable pitches are easily accomplished by a tighter or wider stacking of the discrete slices.

It should be mentioned that a spline-based description can be turned back into a classical mesh by performing a large number of function evaluations in $\mathbf{x}$ and connecting the resulting point cloud by linear edges. Hereby, the evaluation points determine the properties of the mesh and have to be chosen wisely. However, this topic will not be covered in this paper.

For the purpose of generating folding-free planar parameterizations using spline-functions, we adopt the principles of *Elliptic Grid Generation* (EGG) and present a numerical scheme that is suitable for an IgA-based computational approach. In section 2, we will briefly review the concept of (B-)spline functions while in section 3, we present the various possible topologies along with their advantages and disadvantages. In section 4, we discuss the principles of EGG and in sections 7 and 8, we will present the computational approach along with the parameterization strategy we employed for the results presented in section 9

2. B-Splines

B-splines are piecewise-polynomial functions that can be constructed so as to satisfy various continuity properties at the places where the polynomial segments connect. Their properties are determined by the entries of the so-called *knot vector*

$$\Xi = \{\xi_1, \xi_2, \ldots, \xi_{n+p+1}\}. \tag{1}$$

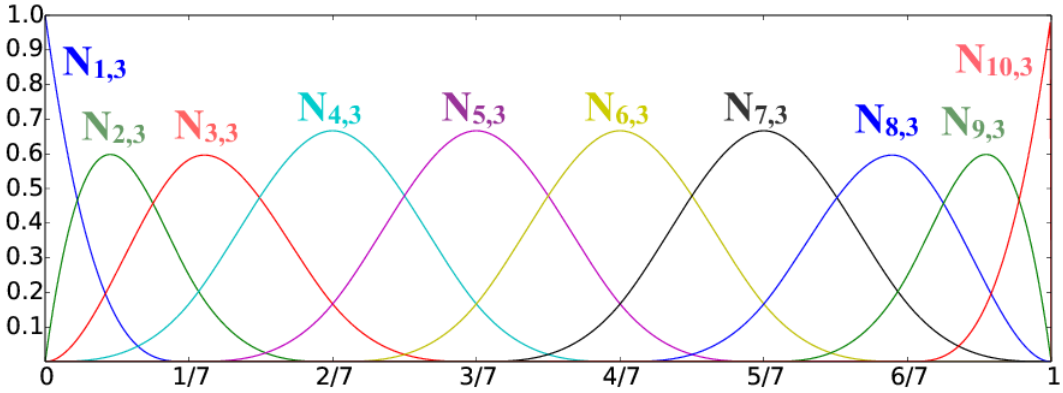


Figure 2. The univariate B-spline basis resulting from the knot-vector Ξ_3.

The knot vector is a monotone-increasing sequence of parametric values $\xi_i \subset [0,1]$ that determine the boundaries of the segments on which the spline-basis is polynomial. Selecting some polynomial order p, the p-th order spline-functions $N_{i,p}$ are constructed recursively, utilizing the relation (with $\frac{0}{0} \equiv 0$)

$$N_{i,q}(\xi) = \frac{\xi - \xi_i}{\xi_{i+1} - \xi_i} N_{i,q-1}(\xi) + \frac{\xi_{i+q+1} - \xi}{\xi_{i+q+1} - \xi_{i+1}} N_{i+1,q-1}(\xi), \tag{2}$$

starting from

$$N_{i,0} = \begin{cases} 1 & \text{if } \xi_i \leq \xi \leq \xi_{i+1} \\ 0 & \text{otherwise} \end{cases}, \tag{3}$$

and iterating until $q = p$. The support of basis function $N_{i,p}$ is given by the interval $\mathcal{I}_{i,p} = [\xi_i, \xi_{i+p+1}]$ and the amount of continuous derivatives across knot ξ_j is given by $p - m_j$, where m_j is the multiplicity of ξ_j in $\mathcal{I}_{i,p}$. In practice, $\xi_1 = 0$ is repeated $p+1$ times as well as ξ_{n+p+1} such that $\xi_1 = \ldots = \xi_{p+1} = 0$ and $\xi_{n+1} = \ldots = \xi_{n+p+1} = 1$. As a result, the resulting basis $\sigma = \{N_{1,p}, \ldots, N_{n,p}\}$ forms a non-negative partition of unity on the entire parametric domain $[0,1]$, that is:

$$\sum_{i=1}^{n} N_{i,p}(\xi) = 1, \tag{4}$$

with

$$N_{i,p}(\xi) \geq 0, \tag{5}$$

for all spline functions $N_{i,p}$ [4]. Figure 2 shows the $p = 3$ B-spline basis resulting from the knot-vector

$$\Xi = \left\{0, 0, 0, 0, \tfrac{1}{7}, \tfrac{2}{7}, \tfrac{3}{7}, \tfrac{4}{7}, \tfrac{5}{7}, \tfrac{6}{7}, 1, 1, 1, 1\right\}. \tag{6}$$

The extension to bivariate spline bases is now straight-forward: given two univariate bases $\sigma_\Xi = \{N_1, \ldots, N_n\}$ and $\sigma_\mathcal{H} = \{M_1, \ldots M_m\}$, we build a bivariate basis $\Sigma = \{w_{i,j}\}_{(i,j)\in\{1,\ldots,n\}\times\{1,\ldots,m\}}$, by means of a tensor-product, where

$$w_{i,j}(\xi, \eta) = N_i(\xi) M_j(\eta). \tag{7}$$

The knots corresponding to the knot-vectors Ξ and $\mathcal{H}$ (without knot-repetitions) used to construct σ_Ξ and $\sigma_\mathcal{H}$, respectively, hereby become the boundaries of the polynomial segments, which can be regarded as the counterparts of classical elements.

We construct the mapping of a B-spline surface as follows:

$$\mathbf{x} = \sum_i \sum_j \mathbf{c}_{i,j} w_{i,j}, \tag{8}$$

where the $\mathbf{c}_{i,j} \in \mathbb{R}^2$ are referred to as the *control points*. We refer to the $\mathbf{c}_{i,j}$ with $i \in \{1, n\}$ or $j \in \{1, m\}$ as the *boundary control points* while the remaining $\mathbf{c}_{i,j}$ are called *inner control points*. As $\mathbf{x}$ is a linear combination of the $w_i \in \Sigma$, it will inherit the local continuity properties of the basis. This implies that many geometrical features can be better captured by a clever choice of the knot multiplicities in Ξ and $\mathcal{H}$.

An additional appealing feature of spline basis functions is the possibility of knot refinement. Let

$$f = \sum_i a_i N_{i,p} \tag{9}$$

be a function from the linear span of the spline basis $\sigma = \{N_{1,p}, \ldots, N_{n,p}\}$ with corresponding knot-vector Ξ. We can refine σ by adding additional knots to Ξ, resulting in the refined basis $\tilde{\sigma}$. It can be shown that $\operatorname{span} \sigma \subset \operatorname{span} \tilde{\sigma}$. For an algorithm to prolong the a_i to the refined basis, see [4].

In the following, we shall drop the tensor-index notation and introduce a global index with $(i, j) \to i + (j-1)m$. The mapping then simplifies to

$$\mathbf{x} = \sum_i \mathbf{c}_i w_i. \tag{10}$$

For many geometries a parameterization with only one mapping operator is not possible, which is why several mappings that jointly parameterize the geometry have to be employed. The individual geometry segments that result from each of the mappings are referred to as *patches*.

3. Choice of Topology

As discussed in section 1, we would like to parameterize the geometry from figure 1 (right) for all rotational angles θ. The discrete angles θ_i then either correspond to the screw-machine at time-instances t_i in the planar case or to some cross-section of the screw-machine in the z-direction. A volumetric parameterization can therefore be acquired by parameterizing the geometry for a large number of θ_i and interpolating the planar cross-sections in the z-direction.
Figure 3 shows the possible topologies that come to mind. Here, black lines indicate boundaries at which the grid is held fixed and red lines indicate boundaries that slide along the grid. The various patches are indicated in different colors. Since the target geometry is of genus two (it has two *'holes'*), at least two patches are needed.

Even though figure 3 (right) takes advantage of the symmetries of the geometry, topology changes are unavoidable. For instance, since the blue patch bounded by the region surrounding the CUSP-points and the two rotor lobes (henceforth referred to as the *separator*) is static while the others rotate, patches (such as the grey patch) will eventually disappear, making a computational simulation with IgA-techniques difficult.

In figure 3 (center), two O-type patches are employed which lead to a sliding interface (in the separator region). Sliding interfaces are generally difficult to handle since element conformity is difficult to achieve. In an IgA-setting, the CUSP-points themselves pose an additional problem:

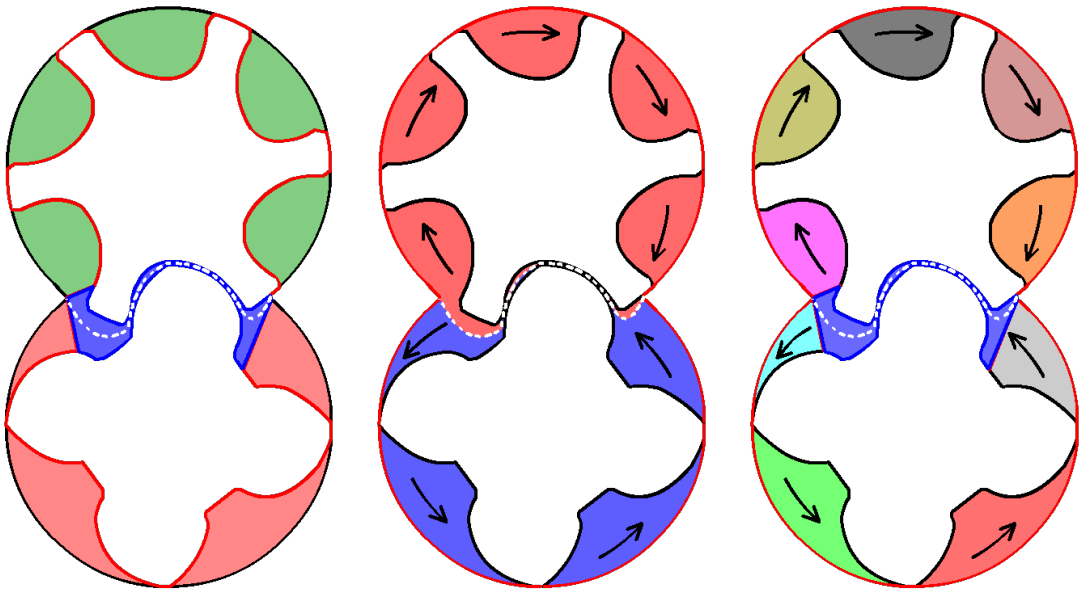


Figure 3. The various possible topologies for the parameterization of the fluid-part of the screw-machines.

as the patches slide along, the CUSP-point C^0-continuities can only be captured by adding repeated knots to the knot-vectors. If several cross sections are interpolated in the z-direction for a volumetric parameterization, they all have to be prolonged to the same knot-vector, soon leading to an infeasibly dense trivariate knot-vector.

Therefore, we are aiming for the topology of figure 3 (left) in which the patches are held fixed at the casings while the rotors slide along and the separator is parameterized with one static patch. Here, no topology changes are required and no sliding interface exists. In all cases, we need to generate the dotted white curve connecting the two CUSP-points, in order to parameterize the separator using two patches with mutual element conformity as well as conformity to the C-type patches.

4. Elliptic Grid Generation

Having discussed the aimed-for topology in section 3, we need a tool to generate analysis-suitable (i.e., bijective or folding-free) parameterizations at every discrete angle θ_i for geometries Ω from no more than a boundary description $\partial\Omega$.

As it is known to produce folding-free meshes in many applications, *Elliptic Grid Generation* (EGG) is among the most popular meshing approaches, especially in settings where computationally inexpensive algebraic methods such as [5] [6] and [7] fail due to the complexity of the target geometry. However, due to the higher chance of success, higher computational costs can be expected.

Assuming Ω is topologically equivalent to the unit quadrilateral $\hat{\Omega} = [0,1] \times [0,1] \subset \mathbb{R}^2$, a mathematical operator $\mathbf{x} : \hat{\Omega} \to \Omega$ that maps $\partial\hat{\Omega}$ onto $\partial\Omega$, and is furthermore folding-free exists and can be constructed. For this purpose, EGG imposes the Laplace-equation on the components of the inverse mapping $\mathbf{x}^{-1}$. Assuming the free topological variables are given by the tuple (ξ, η), the equation takes the form:

$$\begin{cases} \Delta\xi(x,y) &= 0 \\ \Delta\eta(x,y) &= 0 \end{cases} \quad \text{s.t. } \mathbf{x}^{-1}|_{\partial\Omega} = \partial\hat{\Omega}. \tag{11}$$

This system of equations is scaled in order to yield an equation for $\mathbf{x}$ that is suitable for a computational approach. The resulting equations read [8]:

$$\begin{cases} \mathcal{L}(x,y,x) &= 0 \\ \mathcal{L}(x,y,y) &= 0 \end{cases} \quad \text{s.t. } \mathbf{x}|_{\partial\hat{\Omega}} = \partial\Omega, \tag{12}$$

where

$$\mathcal{L}(x,y,f) = g_{22}f_{\xi\xi} - 2g_{12}f_{\xi\eta} + g_{11}f_{\eta\eta}, \tag{13}$$

with $g_{11}(x,y) = \|\mathbf{x}_\xi\|^2$, $g_{12}(x,y) = \mathbf{x}_\xi \cdot \mathbf{x}_\eta$ and $g_{22}(x,y) = \|\mathbf{x}_\eta\|^2$. Since the target space of $\mathbf{x}^{-1}$ is always convex, the bijectivity of the exact solution of (12) follows from the maximum principle [9]. As a result, we may conclude that any sufficiently accurate approximation $\mathbf{x}_h$ of $\mathbf{x}$ will also be bijective. This property distinguishes EGG from most other meshing techniques, which may tend to produce folded meshes, due to them being less grounded in mathematical theory.

4.1. Discretization

Traditionally, (12) is approximately solved using a finite-difference approach [8]. However, since this only yields a finite collection of grid points that can serve as the vertices for a classical mesh, we have to look for different options. We discretize the equations with FEM-techniques. First, we introduce the operator

$$\tilde{\mathcal{L}}(x,y,f) = \frac{\mathcal{L}(x,y,f)}{g_{11}+g_{22}}. \tag{14}$$

As a next step, we select a $p \geq 2$ bivariate B-spline basis $\Sigma = \{w_1, \ldots, w_n\}$ with global C^1-continuity (i.e., the $w_i \in \Sigma$ possess at least one continuous derivative in $\hat{\Omega}$). By Σ_0, we denote the subset of Σ consisting of inner basis functions, that is, the collection of $w_i \in \Sigma$ that vanish on $\partial\hat{\Omega}$ (corresponding to the inner control points from section 2). We discretize (12) as follows:

$$\forall w_i \in \Sigma_0 : \begin{cases} \int_{\hat{\Omega}} w_i \tilde{\mathcal{L}}(x_h, y_h, x_h) \mathrm{d}\boldsymbol{\xi} = 0 \\ \int_{\hat{\Omega}} w_i \tilde{\mathcal{L}}(x_h, y_h, y_h) \mathrm{d}\boldsymbol{\xi} = 0 \end{cases}, \quad \text{s.t. } \mathbf{x}_h|_{\partial\hat{\Omega}} \simeq \partial\Omega. \tag{15}$$

Denoting by $\mathcal{I}$ and $\mathcal{I}_0$ the index-sets of Σ and Σ_0, respectively, $\mathbf{x}_h$ will be of the form

$$\mathbf{x}_h = \sum_{i\in\mathcal{I}_0} \mathbf{c}_i w_i + \sum_{i\in\mathcal{I}\setminus\mathcal{I}_0} \mathbf{d}_i w_i, \tag{16}$$

where the $\mathbf{c}_i$ denote the inner control points and the $\mathbf{d}_i$ the boundary control points, which follow from a regression of the input point cloud and serve as to ensure that the mapping operator satisfies $\mathbf{x}_h|_{\partial\hat{\Omega}} \simeq \partial\Omega$ (see section 5). As the $\mathbf{d}_i$ are known, the objective is to find the $\mathbf{c}_i$ such that $\mathbf{x}_h$ satisfies (15). Equation (15) leads to a nonlinear root finding problem of the form

$$\mathbf{F}(\mathbf{c}) = \mathbf{0}, \tag{17}$$

where the vector $\mathbf{c}$ contains the unknown inner control points $\mathbf{c}_i$.

The scaling introduced in (14) has the advantage of allowing for a more scaleable convergence criterion $\|\mathbf{F}(\mathbf{c})\| \leq \epsilon$, that is, the value of ϵ that corresponds to a converged solution is less sensitive to the characteristic length-scale of the geometry (and can therefore be taken approximately equal in all cases). The choice of Σ and the optimal selection of the inner control points $\mathbf{d}_i$ shall be the topic of section 5, while the computational approach for solving (17) will be the topic of section 7.

5. Contour Approximation and Choice of Basis

Given four input point clouds corresponding to each boundary of Ω: $P_\alpha = \{\mathbf{p}_\alpha^i\}_{i=1}^{I_\alpha}$ with $\alpha \in \{s, e, n, w\}$, the selection of a suitable spline basis Σ and the corresponding boundary control points $\mathbf{d}_i$ (see section 2) constitutes a preliminary step before (17) is tackled computationally. For this purpose we select a coarse initial basis $\Sigma_\square$ (resulting from two coarse univariate knot-vectors) and four sets of monotone increasing parametric values $\{\xi_\alpha^i\}_{i=1}^{I_\alpha}$, each starting on $\xi_\alpha^1 = 0$ and ending on $\xi_\alpha^{I_\alpha} = 1$. Let $\mathbf{m}_s(s) = (s, 0)$, $\mathbf{m}_e(s) = (1, s)$, $\mathbf{m}_n(s) = (s, 1)$ and $\mathbf{m}_w(s) = (0, s)$. The objective is to select the $\mathbf{d}_i$ such that $\mathbf{x}_h(\mathbf{m}_\alpha(\xi_\alpha^i)) \simeq \mathbf{p}_\alpha^i$. To this purpose a least-squares regression is carried out by minimizing the functional

$$R(\partial\Omega, \mathbf{d}) = \frac{1}{2} \sum_{\alpha\in\{s,e,n,w\}} \sum_{i=1}^{I_\alpha} \left\|\mathbf{x}_h(\mathbf{m}_\alpha(\xi_\alpha^i)) - \mathbf{p}_\alpha^i\right\|^2 \tag{18}$$

over the vector of boundary control points $\mathbf{d} = (\ldots, \mathbf{d}_j, \ldots)^T$. Hereby, it is advisable to constrain the corner control points to the corners of the input control points in order to avoid mismatches. In practice, (18) can suffer from instabilities. This is usually a result of the local amount of DOFs exceeding the local amount of points. We add a small least-distance penalty term to (18) in order to improve the stability:

$$\tilde{R}(\partial\Omega, \mathbf{d}) = \frac{1}{2} \sum_{\alpha\in\{s,e,n,w\}} \left(\sum_{i=1}^{I_\alpha} \left\|\mathbf{x}_h(\mathbf{m}_\alpha(\xi_\alpha^i)) - \mathbf{p}_\alpha^i\right\|^2 + \lambda \int_{\gamma_\alpha} \left\| \frac{\partial \mathbf{x}_h}{\partial \boldsymbol{\xi}} \cdot \hat{\mathbf{t}} \right\|^2 \mathrm{d}s \right), \tag{19}$$

where $\hat{\mathbf{t}}$ is the unit tangent vector and γ_α the subset of $\partial\hat{\Omega}$ that corresponds to side α. Here $\lambda > 0$ is a small penalty factor whose value should be chosen small enough not to noticeably alter the outcome of (19) while avoiding instabilities.

After (19) has been minimized, the mismatch

$$r_{\alpha,i} = \left\|\mathbf{x}_h(\mathbf{m}_\alpha(\xi_\alpha^i)) - \mathbf{p}_\alpha^i\right\| \tag{20}$$

serves as a local refinement criterion. Whenever $r_{\alpha,i}$ exceeds the approximation tolerance $\mu > 0$, we place a knot in the center of the element corresponding to the knot-vector of side α that contains ξ_α^i in order to locally increase the resolution of the basis.

Above steps are repeated until the convergence criterion is reached. Upon completion, we are in the possession of the coarse- and fine-grid bases $\Sigma_\square$ and Σ with corresponding vector of boundary control points $\mathbf{d}_\square$ and $\mathbf{d}$. The fine-grid basis then constitutes the coarsest possible basis to properly resolve the boundary condition

$$\mathbf{x}|_{\partial\hat{\Omega}} = \partial\Omega. \tag{21}$$

The tuple $(\Sigma_\square, \mathbf{d}_\square)$ serves as a means to build initial guesses for the computational approach employed to solve (17) (see section 7).

It should be noted that the choice of Σ is purely based on its capabilities to properly approximate $\partial\Omega$. However, this does not mean that it yields sufficient resolution to appropriately approximate the solution of (12) at every point in $\hat{\Omega}$. Heuristically, folding due to insufficient accuracy is uncommon. Should it nevertheless happen, we refine Σ by adding knots to the knot-vectors and prolong $\mathbf{d}$ to the refined basis Σ_r. Equation (17) is solved with Σ_r to yield a better approximation at internal points of $\hat{\Omega}$. This step can be repeated until the approximation is sufficiently accurate and bijectivity is achieved.

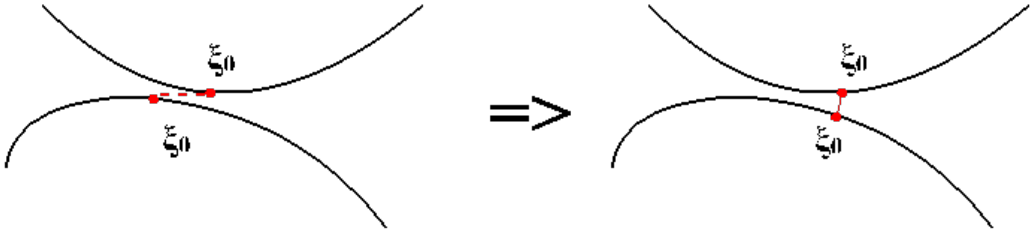


Figure 4. Skewed isolonies due to poorly parameterized boundary contours and improved parametric properties resulting from reparameterization. Near the tiny gaps, it is of major importance that the same parametric value is assumed on either side.

6. Choice of Parametric Values

The parametric values $\{\xi_\alpha^i\}_{i=1}^{I_\alpha}$ from section 5 have a profound influence on the parametric properties of the resulting parameterization and have to be chosen wisely. By default, the input point clouds are chord-length parameterized. Defining l_i recursively by

$$l_{i+1} = l_i + \|\mathbf{p}_{i+1} - \mathbf{p}_i\|, \tag{22}$$

starting with $l_1 = 0$ and ending on l_{I_α}, a chord-length parameterization corresponds to taking

$$\xi_\alpha^i = \frac{l_i}{l_{I_\alpha}}. \tag{23}$$

Using (23) ensures that the parametric velocity at the boundaries is (approximately) constant and therefore constitutes the default choice. In the presence of extreme aspect ratios (tiny gaps), however, we have found (23) to lead to dissatisfactory results. To ensure the quality of the resulting parameterization, we need to ensure that the (approximately) same parametric value is assumed on either side of the gaps (see figure 4).

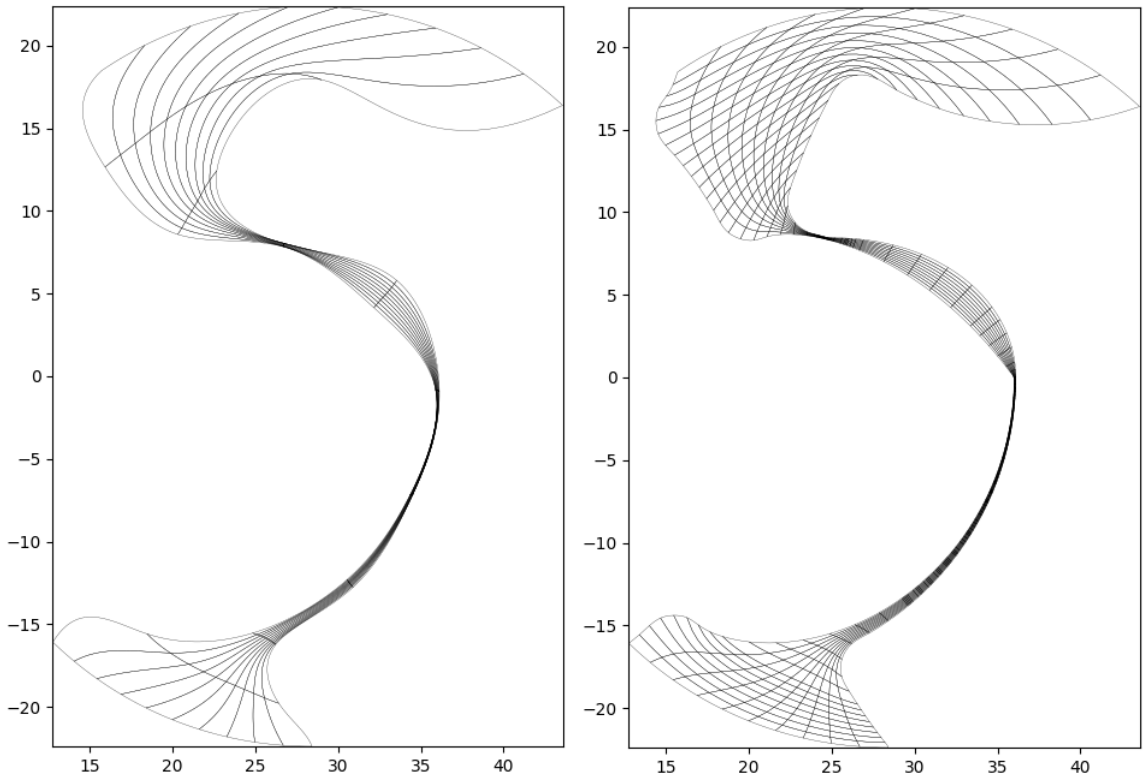
For this purpose, we employ the matching algorithm proposed in [10] to two opposite point clouds P_w, P_e (or P_n, P_s) in order to match pairs of points that are too close. Let $\mathcal{I}_w = \{2, \ldots, I_w - 1\}$ and $\mathcal{I}_e = \{2, \ldots, I_e - 1\}$, upon completion of the matching, we are in the possession of a finite set of matched tuples

$$\mathcal{I}^m = \{(i, j) \mid \mathbf{p}_i \in P_w \text{ and } \mathbf{p}_j \in P_e \text{ have been matched}\}. \tag{24}$$

The tuples $(i, j) \in \mathcal{I}^m$, $i \in \mathcal{I}_w$, $j \in \mathcal{I}_e$ can be utilized to build a reparameterization function that improves the parametric properties of the mapping. By $\{\hat{\xi}_\alpha^i\}_{i=1}^{I_\alpha}$, we denote the default chord-length parameterization resulting from (23). We assign the parametric values $\{\hat{\xi}_w^i\}_{i=1}^{I_w}$ to the points $\mathbf{p}_i \in P_w$. Furthermore, we set $\xi_e^j = \hat{\xi}_w^i$ whenever $(i, j) \in \mathcal{I}^m$. The question remains what parametric value to impose on the unmatched points $\mathbf{p}_j \in P_e$. Given the set $\mathcal{I}^k = \{j \mid \xi_e^j \text{ is known}\}$ (note that $\xi_e^1 = 0$ and $\xi_e^{I_e} = 1$), we carry out a monotone cubic spline-interpolation [11] of the values $\{\hat{\xi}_e^i\}_i$, $i \in \mathcal{I}^k$ versus the known values $\{\xi_e^i\}_i$, $i \in \mathcal{I}^k$, to yield the monotone reparameterization function $\xi_e'(\xi)$. The ξ_e^i then follow from evaluating ξ_e' in the $\hat{\xi}_e^i$, that is

$$\xi_e^i = \xi_e'\left(\hat{\xi}_e^i\right). \tag{25}$$

In practice, reparameterization is a crucial ingredient for the successful parameterization of the target geometry and should therefore always be employed along with the numerical approach that is the topic of section 7.



(a) Coarse-grid solution (b) Fine-grid solution

Figure 5. Coarse- (a) and fine-grid solution (b) of a challenging input geometry.

7. Computational Approach

After completion of the regression from section 5, possibly in conjunction with reparameterization (see section 6), we are in the position to tackle the root-finding problem from equation (17). The point cloud regression yields the tuples $(\Sigma_\square, \mathbf{d}_\square)$ and $(\Sigma, \mathbf{d})$ of coarse- and fine-grid bases with corresponding boundary control points. We tackle (17) with a truncated Newton-approach. We first solve the coarse-grid problem $\mathbf{F}_\square(\mathbf{c}_\square) = \mathbf{0}$ using transfinite interpolation [5] to generate an initial guess $\mathbf{c}_\square^0$. Both the coarse- and the fine-grid problem are of the form $\mathbf{G}(\mathbf{c}) = \mathbf{0}$, where $\mathbf{G}$ is nonlinear in $\mathbf{c}$. Given some initial guess $\mathbf{c}^0$, the new iterate is computed using the following recursive relation:

$$\left.\frac{\partial \mathbf{G}}{\partial \mathbf{c}}\right|_{\mathbf{c}=\mathbf{c}^i} \Delta\mathbf{c} = -\mathbf{G}(\mathbf{c}^i), \tag{26}$$

$$\mathbf{c}^{i+1} = \mathbf{c}^i + \delta\Delta\mathbf{c}. \tag{27}$$

Here $0 < \delta \leq 1$ is a truncation parameter whose optimal value is estimated from the current and updated tangents and residuals. Upon solution of the coarse-grid problem $\mathbf{F}_\square(\mathbf{c}_\square) = \mathbf{0}$, the coarse-grid solution $\mathbf{c}_\square$ is prolonged to the fine-grid basis Σ and serves as an initial guess for the fine-grid problem $\mathbf{F}(\mathbf{c}) = \mathbf{0}$.

In practice, the coarse-grid problem typically converges after $4 - 6$ iterations, while the fine-grid problem requires an additional $2 - 3$ iterations. Thanks to the relatively small number of DOFs in $\Sigma_\square$, the impact of the large number of required coarse-grid iterations is manageable.

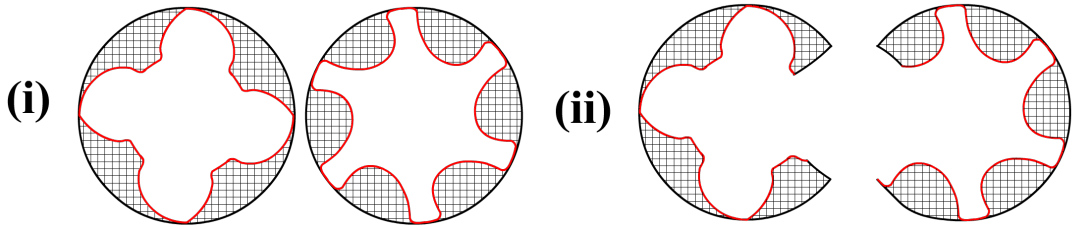


Figure 6. Steps (i) and (ii) of the parameterization strategy.

Since the required number of fine-grid iterations is approximately halved, the expected speed-up is $\sim 50\%$. Furthermore, the robustness of the approach is greatly improved: thanks to the high quality of the initial guess, failure of convergence is extremely uncommon. As an example, figures 7 (a) and (b) show the mapping corresponding to the coarse- and fine-grid solution of a challenging geometry. Here, we performed reparameterization (see section 6) on the western and eastern boundaries, keeping the western boundary chord-length parameterized while letting the eastern boundary float. Convergence on the fine grid is reached within 3 nonlinear iterations.

8. Application to Twin-Screw Machine Geometries

The computational approach discussed in section 7 constitutes the basic ingredient for the parameterization approach that will be the topic of this section. When no good initial guess is available, we will employ the hierarchical approach from section 7. Else, we will use the initial guess to solve the fine-grid problem right away. Selecting K uniformly-spaced discrete angles θ_k from the interval $[0, \pi/2]$, the objective is to compute a planar paramterization for every θ_k. The approach consists of the following steps:

(i) Generate separate O-type parameterization for the male and female rotors with casing for every θ_k.

(ii) Cut the O-parameterizations at the CUSP-points in order to produce two C-parameterizations for every discrete angle.

(iii) Combine the cuts with the male and female rotor parts to form a contour description of the separator.

(iv) Compute single-patch parameterizations for the separator at every discrete angle.

(v) Use the single-patch parameterized separators to generate curves, connecting the two CUSP-points and splitting the separator in half.

(vi) Generate parameterizations on the left and on the right of the splitting curves using the same knot vector as for the C-grids to acquire a conforming parameterization.

The key-steps are depicted in figures 6 and 7.

To generate the O-parameterizations from (i), we first generate exact (chord-length parameterized) cubic spline-fits through the input point clouds of the rotors and casings, using one of the FITPACK [12] routines. We evaluate both spline fits in N uniformly-spaced points over the parametric interval $[0, 1]$ and utilize the resulting point clouds to build male and female reparameterization functions η'_m and η'_f, respectively (see section 6). Hereby the casings are held chord-length parameterized (i.e., their reparameterization functions are simply given by the identity function) while we let the rotor parameterizations float.

At every discrete angle θ_k, we act with the canonical rotation matrix on the control points of the fitted spline-curves and reparameterize such that the CUSP-points always coincide with the same parametric values η_1 and η_2. Since the casings are chord-length parameterized, we

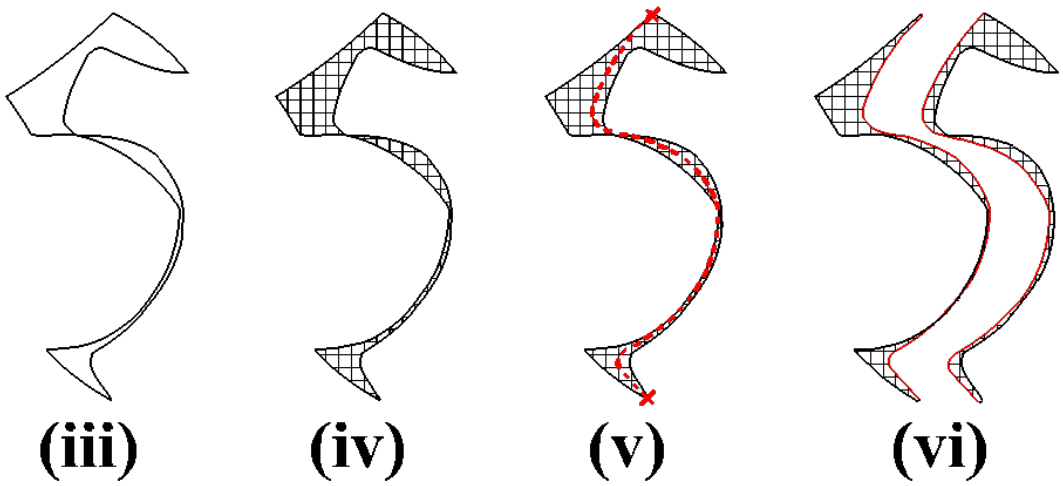


Figure 7. Steps (iii) to (vi) of the parameterization strategy.

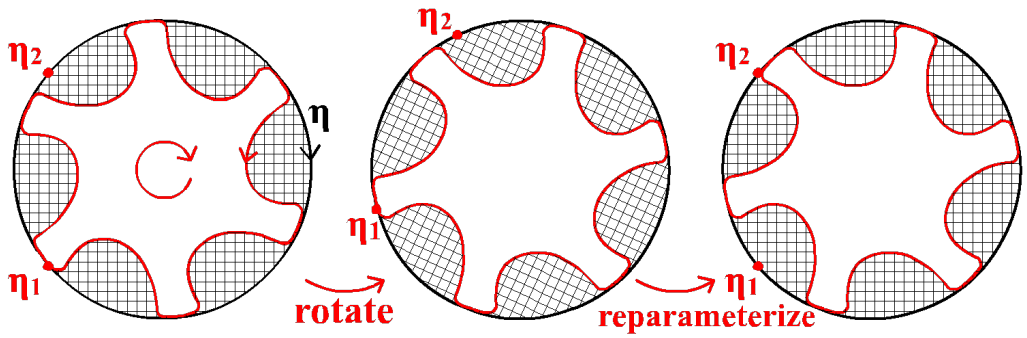


Figure 8. At every angle θ_k we act with the canonical rotation matrix on the control points of the spline-fit and reparameterize such that the CUSP-points always correspond to the same parametric values η_1 and η_2.

reparameterize simply by a shift of $-\theta_k/(2\pi)$ in the parametric domain. The rotor spline-fits have to be shifted by the same value in the reparameterized η'-domain. Therefore, we compute the shift in η by inverting $\eta'_{m,f}$ (see figure 8). As a next step, we utilize the shifted spline fits to generate a large number of uniformly-spaced points which serve as an input point cloud. We utilize the reparameterization functions $\eta'_{m,f}$, which we shift in a way similar to the spline fits, to assign parametric values to the rotor point clouds.

For the O-parameterizations, we utilize a $p \geq 2$-th order knot-vector which is periodic in the η-direction while disregarding the northern and southern boundaries. We compute parameterizations corresponding to θ_k, $k = 1, \ldots, 6$ utilizing the hierarchical approach from section 7. Upon completion, for each θ_{k+1}, we extrapolate the inner control points of $\theta_{k-6}, \ldots, \theta_k$ to θ_{k+1} and utilize them as an initial guess.

Upon completion of the K slices for each rotor, we add the $p+1$-times repeated knots η_1 and η_2, the η-values that correspond to the CUSP-points in $\hat{\Omega}$, to the knot vectors of the O-grids. This way, the two separate male and female rotor O-grids are each split into two parts: one C-grid and one grid which is discarded. By cutting in the domain, we avoid accidentally cutting the rotor lobes twice, as a result, the cuts are not (necessarily) straight.

Upon completion, our database is filled with left and right C-grid parameterizations for each

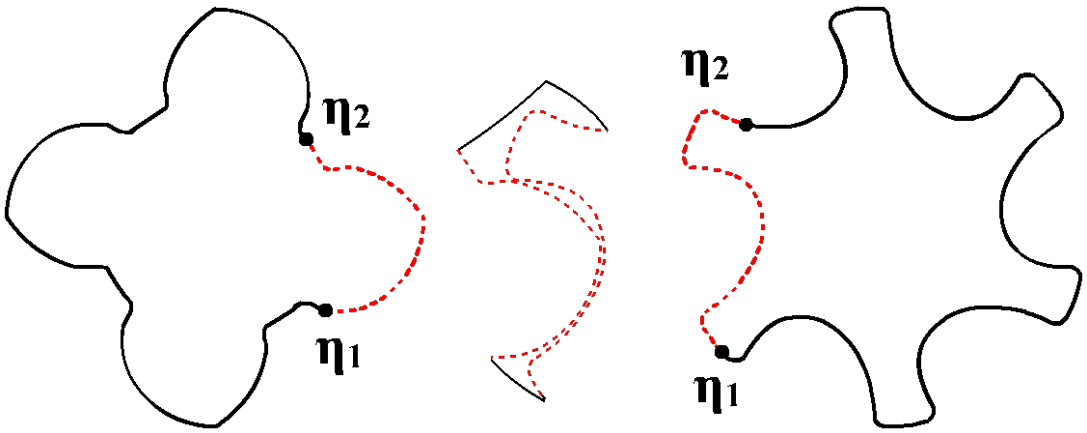


Figure 9. The exact rotor spline-fits are evaluated from η_1 to η_2 and the resulting point clouds are combined with the C-grid cuts

discrete angle θ_k. Having completed steps (i) and (ii), a description of the separator contours can be acquired by evaluating the exact rotor spline-fits from η_1 to η_2 (on both sides) and combining the resulting point clouds with the C-grid cuts (see figure 9).
The resulting geometries are parameterized with one patch. Instead of sequentially parameterizing all K slices, we start off by building parameterizations for each L-th slice (with $L \ll K$), keeping the western boundary chord length parameterized. Here, we employ the hierarchical approach from section 7. Let η'_k, denote the reparameterization function for the k-th slice. Upon completion of every L-th parameterization, we are in the possession of K/L single-patch parameterizations for the separator as well as a reparameterization function η'_k, $k = 1, L, 2L, \ldots$ for every L-th slice. We build reparameterization functions for the remaining slices by blending the available η'_k to achieve full smoothness of the parametetric properties in θ.

We use the K/L available single-patch parameterations and interpolate them in θ. The inner control points of the resulting interpolation function are extracted and serve as an initial guess for the remaining slices, in a way similar to the rotor O-grids.

Upon completion of all K-slices, we traverse $\hat{\Omega}$ from one CUSP-point to the other, leaving splitting-curves in our wake (see figure 10).
Their properties are fully determined by the path taken. To maximize the quality of the parameterizations in step (vi), we traverse $\hat{\Omega}$ such that the separator is split most-evenly on both sides of the small gaps. Upon completion, we are in the possession of a splitting-curve for all discrete angles θ_k. As a last step, we utilize the resulting database to parameterize the separator with two patches, one on each side of the splitting curve. Hereby, we employ the same computational approach as for the single-patch parameterized separator. To achieve boundary conformity between separator and C-grids, we employ the same knot-vector(s) in the ξ-direction.

9. Results

We have implemented the parameterization approach proposed in section 8 in the FEM python-library *Nutils* [13]. The geometry has been parameterized for $K = 200$ discrete angles over the interval $[0, \pi/2]$ (corresponding to a quarter rotation on the male rotor after which the initial position is again assumed). Figures 11 to 13 show the two rotor C-grids at $\theta = \theta_1$, $\theta = \theta_{75}$ and $\theta = \theta_{150}$.
We have used $L = 5$ to fill database with 40 single-patch parameterizations of the separator. The single-patch parameterized separator along with the computed splitting-curve is plotted for

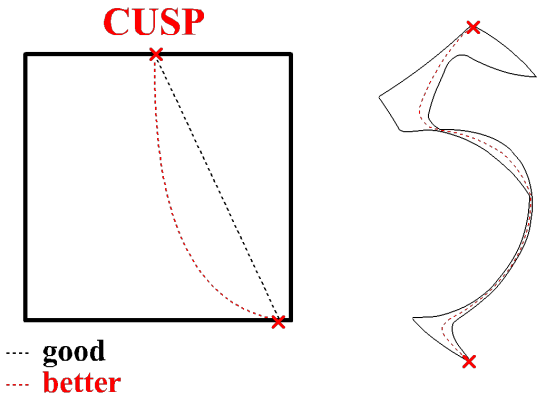


Figure 10. A splitting curve can be generated by traversing the computational domain from one CUSP-point to the other. The properties of this splitting curve is tuned by changing the path taken. We choose the path such that the separator is split most-evenly at the narrow gaps.

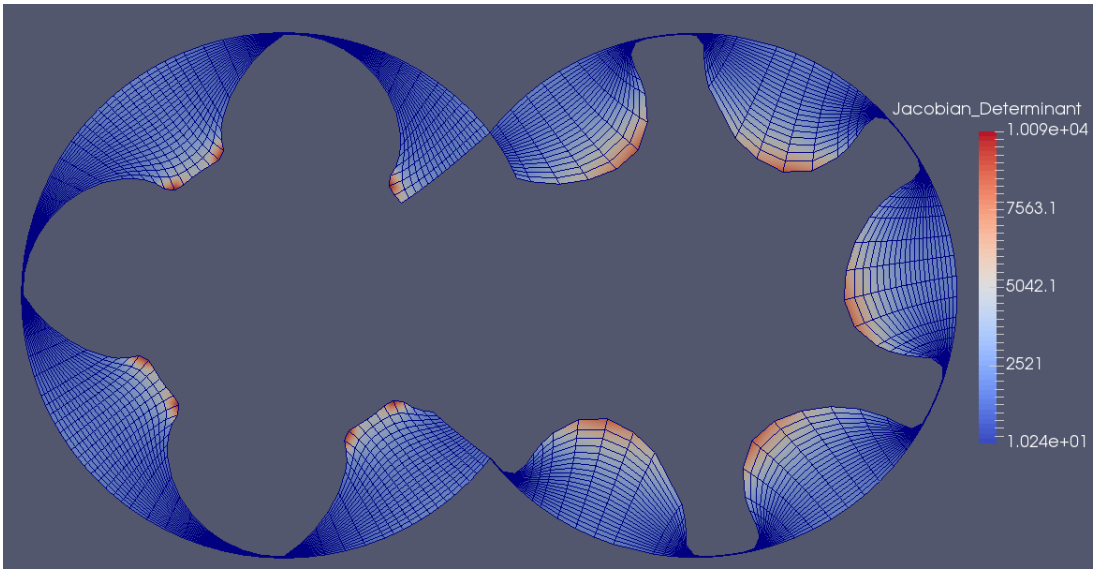


Figure 11. The two C-grids at $\theta = \theta_1$

$\theta = \theta_1$, $\theta = \theta_{75}$ and $\theta = \theta_{150}$ in figure 14. Here, we do not plot the isolines for improved visibility. Figure 15 shows the final geometry at $\theta = \theta_{100}$ and 17 figure (b) a zoom-in on the conforming separator showing the parametric properties by the splitting curve.
Finally, figure 16 shows the final geometry at $\theta = \theta_1$ and 17 (a) a zoom-in on the separator.

With $K = 200$, the computation of the rotor O-grids converges after 1 iteration as soon as enough slices for a 5-th order extrapolation are available. The 40 ($L = 5$) initial single-patch parameterized separators that we utilize to build an interpolation function typically converge within 3 iterations on the fine-grid using the hierarchical approach from section 7. Using the interpolation function to build an initial guess, for the remaining slices convergence is typically reached within 1 iteration with a maximum of 2. The same level of efficiency is achieved for the two-patch parameterized separator.

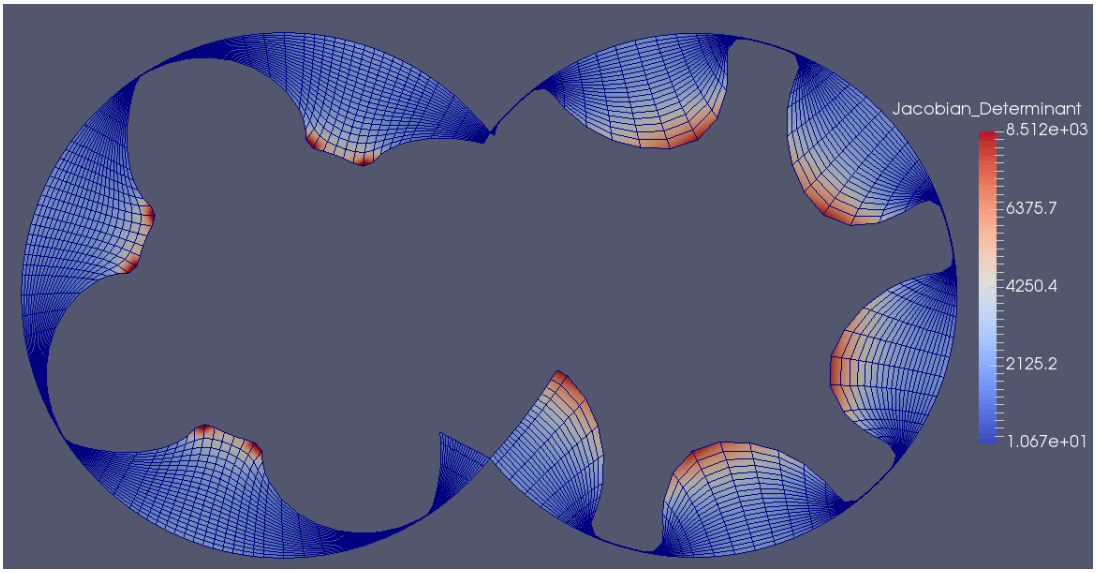


Figure 12. The two C-grids at $\theta = \theta_{75}$

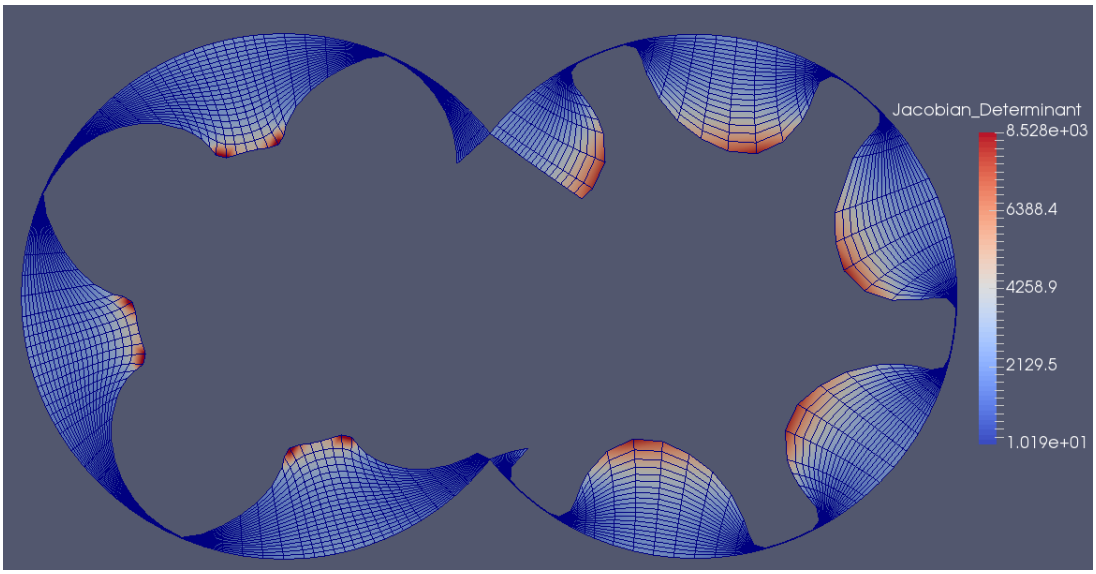
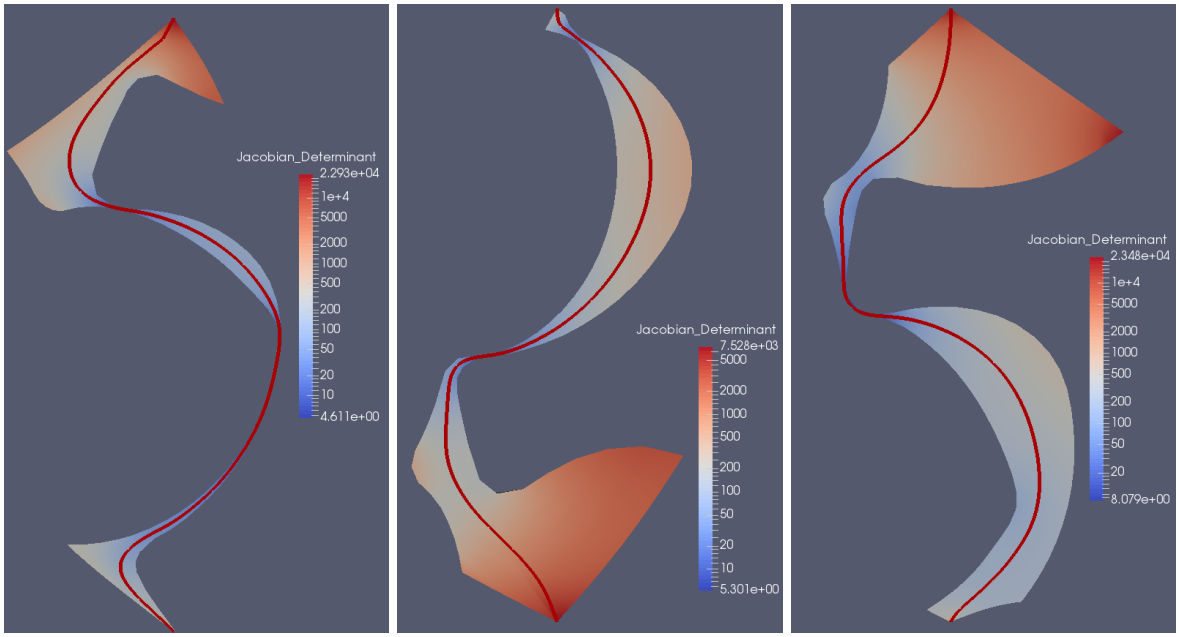


Figure 13. The two C-grids at $\theta = \theta_{150}$

10. Discussion

We have successfully implemented the approach proposed in section 8 utilizing the principles from sections 4 to 7. Here, inter- and extrapolation in the rotational angle θ greatly improves the efficiency since the number iterations is reduced to only 1 in almost all cases (against typically 3 without interpolation). The quality of the interpolation (and by that the expected number of required iterations) greatly depends on the value of K. Here, we used $K = 200$ which is a realistic number for an accurate flow-simulation. The reduction to only 1 iteration is remarkable since it implies that in this setting, EGG is nearly as efficient as algebraic parameterization techniques while yielding superior results.

The properties of the splitting curve, which are fully determined by the path taken in $\hat{\Omega}$, have a profound influence on the parametric properties of the two-patch parameterized separator as



(a) $\theta = \theta_1$ (b) $\theta = \theta_{75}$ (c) $\theta = \theta_{150}$

Figure 14. The single-patch parameterized separator along with the generated splitting curve for various values of θ.

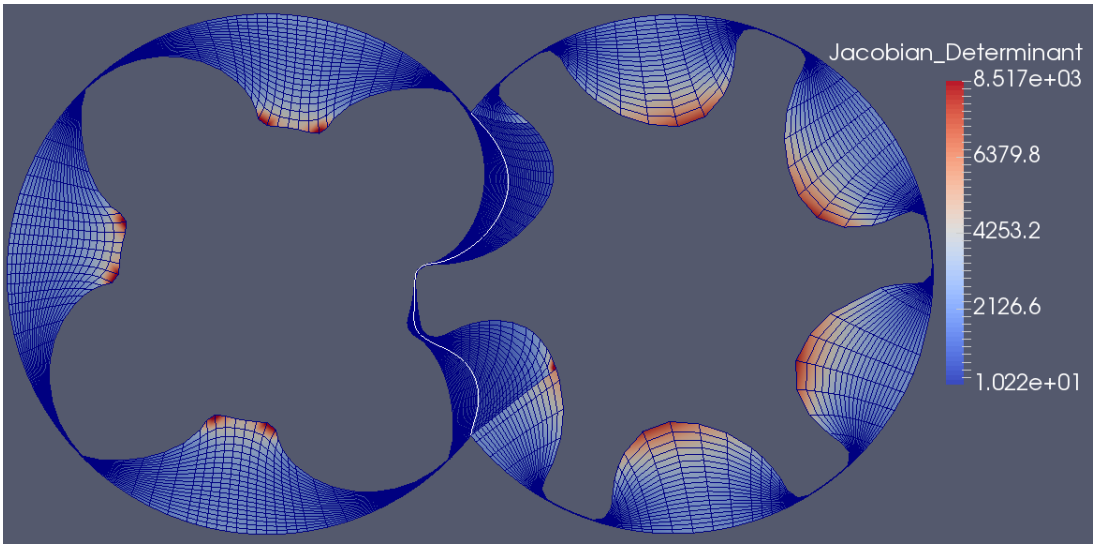


Figure 15. The two-patch parameterized separator along with the splitting curve at $\theta = \theta_{100}$

can be seen in figure 17. In (a), we clearly see the steep inter-element angles at the splitting curve interface. In (b) this is less pronounced. We conclude that an approximate halving of the separator by the splitting curve may not be the best quality criterion for all rotational angles θ and that further optimization is necessary. Hereby, it will be important to make a good trade-off between the steepness of the inter-elements angles at the splitting curve and C-grid interfaces which can, in turn, serve as a splitting curve selection criterion. A posteriori smoothing may be an option, too.

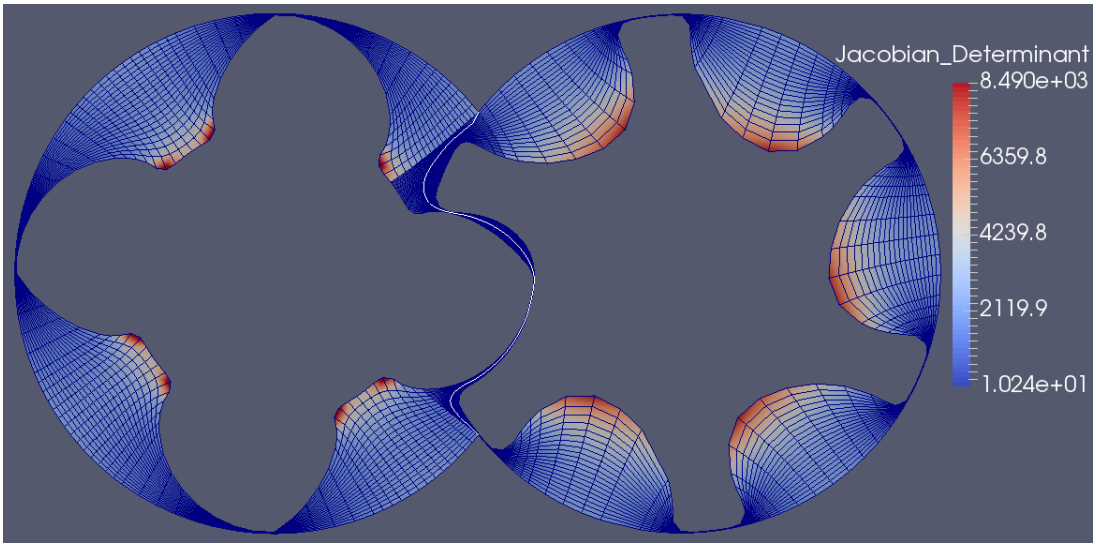
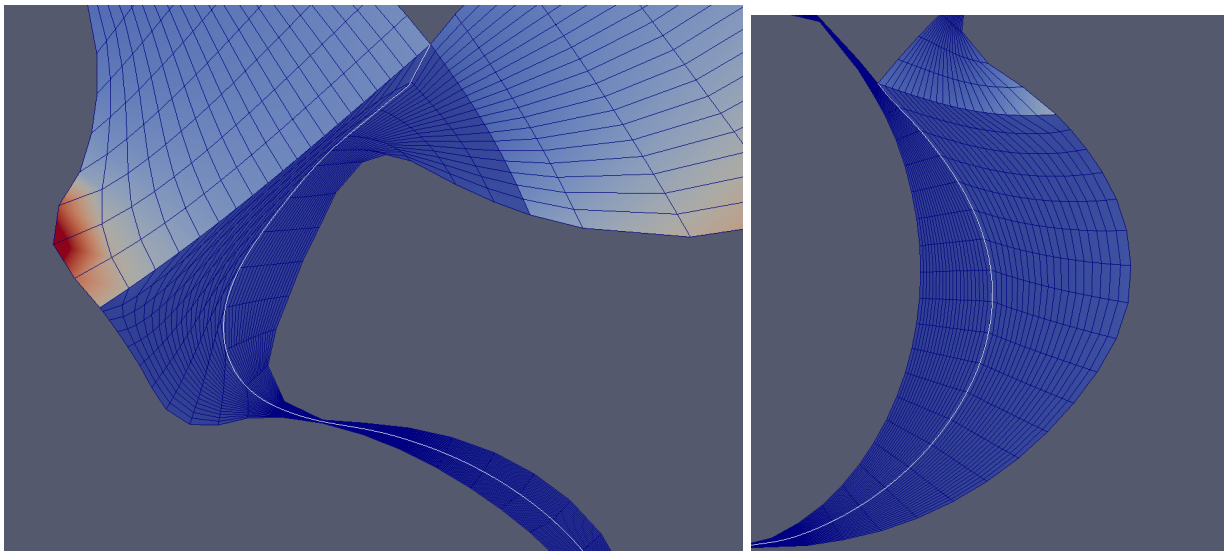


Figure 16. The two-patch parameterized separator along with the splitting curve at $\theta = \theta_1$



(a) $\theta = \theta_1$ (b) $\theta = \theta_{100}$

Figure 17. A zoom-in on the two-patch parameterized separator at $\theta = \theta_1$ (a) and $\theta = \theta_{100}$ (b).

10.1. Shape Optimization

As stated in section 1, we are particularly interested in performing shape optimization on a variable pitch-function. Given a particular rotor profile input, the objective is to minimize the objective function over the three-vector of shape parameters $\boldsymbol{\alpha}$ comprised of left- and right pitches $\dot{\theta}_{l,r}$ and the z-coordinate l_z at which the pitch changes. We base our approach on the observation that the planar slices at angle θ_k coincide with planar cross-sections of a volumetric parameterization in the z-direction.

The idea is to parameterize the geometry at a large number of discrete angles θ_k with angular increment $\Delta\theta$ and fill the database with a large number of planar slices $\mathbf{x}_h^\theta$. A parameterization

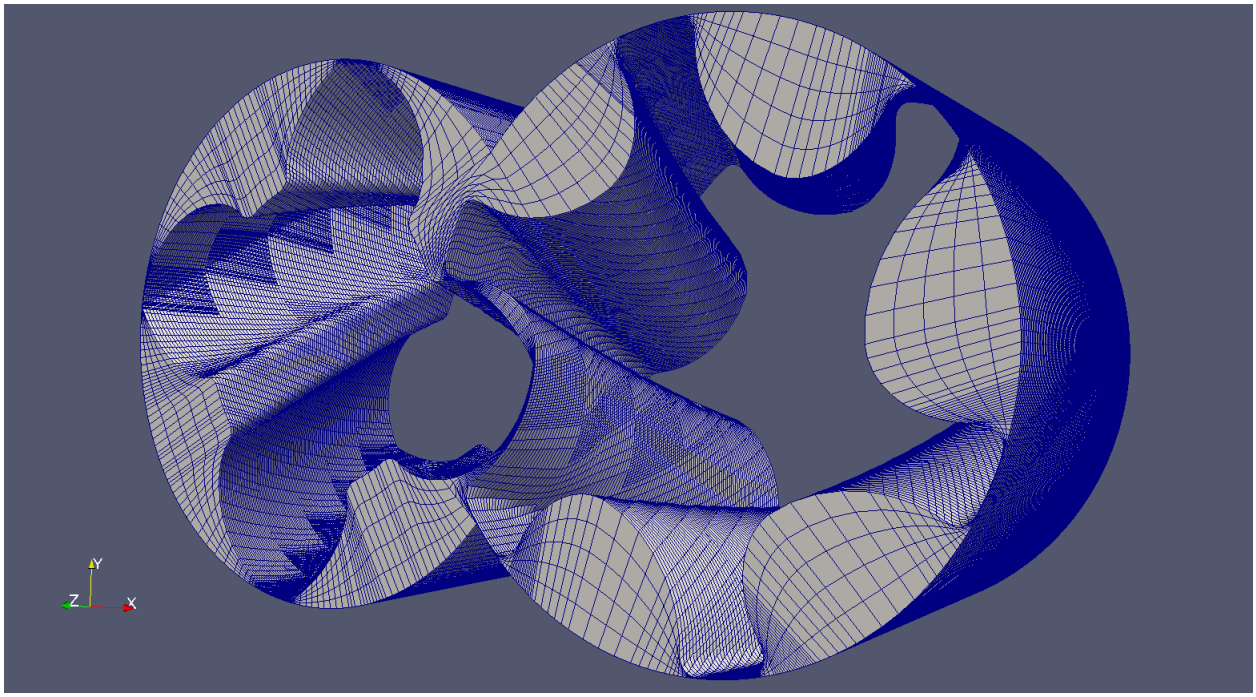


Figure 18. Part of a volumetric parameterization acquired by stacking a large number of planar slices.

for a particular configuration of $\boldsymbol{\alpha} = \left(\dot{\theta}_l, \dot{\theta}_r, l_z\right)^T$ is accomplished by a proper stacking of the slices in the z-direction. Hereby, a (locally) larger pitch will require a higher density of slices for an accurate description of the geometry, while a lower pitch allows for less slices. A database should therefore be generated with a slice-density that corresponds to the largest admissible pitch in the design space. Lower-pitched segments can be parameterized using a subset of the available slices $\mathbf{x}_h^\theta$. Since interpolation in the z-direction is a relatively cheap operation, a decent parameterization for a certain $\boldsymbol{\alpha}$ comes at a relatively low cost.

Figure 18 shows a segment of a volumetric geometry with constant pitch, generated by the stacking of a large number of planar slices in the z-direction.
Finally, figure 19 shows a segment of the separator with non-constant pitch along with a dotted red line to indicate the z-coordinate at which the pitch changes. This geometry has been constructed using the same planar slices as figure 18 but with a tighter stacking in the stronger-pitched region.

11. Conclusion

In this article we presented a practical approach for the parameterization of twin-screw machine geometries with spline functions. For this, we adopted the principles of Elliptic Grid Generation and presented a computational approach that is compatible with the principles of Isogeometric Analysis. We presented automated boundary contour reparameterization techniques that further improve quality of the resulting parameterization.

We have successfully applied the approach to a twin-screw machine geometry. We have concluded that the parametric properties can be improved by optimizing the properties of the splitting curve that is a necessary ingredient for the parameterization of the separator. Finally, we have given a qualitative explanation of how the proposed techniques may be employed for a database-driven shape-optimization on a variable rotor pitch and presented an example of a

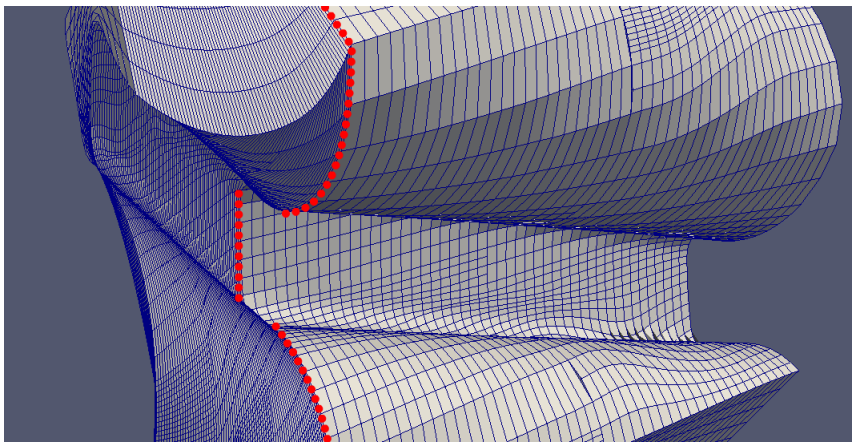


Figure 19. Part of a volumetric parameterization of the separator with non-constant pitch. The coordinate at which the pitch changes is indicated by the dotted red line.

volumetric parameterization resulting from the stacking of a large number of planar slices.

ACKNOWLEDGEMENTS
This project (MOTOR) has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 678727.

Bibliography
[1] Twin-Mesh Twin-mesh software `https://www.twinmesh.com/` [Online; accessed 2018-03-24]
[2] SCORG Screw compressor rotor grid generator `http://pdmanalysis.co.uk/scorg/` [Online; accessed 2018-03-24]
[3] ANSYS I Ansys cfx `https://www.ansys.com/products/fluids/ansys-cfx/` [Online; accessed 2018-03-24]
[4] Hughes T J, Cottrell J A and Bazilevs Y 2005 Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement *Computer methods in applied mechanics and engineering* **194** 4135–4195
[5] Coons S A 1967 Surfaces for computer-aided design of space forms Tech. rep. DTIC Document
[6] Gordon W J and Hall C A 1973 Construction of curvilinear co-ordinate systems and applications to mesh generation *International Journal for Numerical Methods in Engineering* **7** 461–477
[7] Gordon W J and Thiel L C 1982 Transfinite mappings and their application to grid generation *Applied Mathematics and Computation* **10** 171–233
[8] Thompson J F, Soni B K and Weatherill N P 1998 *Handbook of grid generation* (CRC press)
[9] Castillo J E 1991 *Mathematical aspects of numerical grid generation* (SIAM)
[10] Hinz J, Möller M and Vuik C 2018 Elliptic grid generation techniques in the framework of isogeometric analysis applications *Computer Aided Geometric Design*
[11] Fritsch F N and Carlson R E 1980 Monotone piecewise cubic interpolation *SIAM Journal on Numerical Analysis* **17** 238–246
[12] Dierckx P 1995 *Curve and surface fitting with splines* (Oxford University Press)
[13] van Zwieten G, Verhoosel C, van Zwieten J, van Opstal T and Hoitinga W 2016 Nutils v2.0 URL `https://doi.org/10.5281/zenodo.822381`

List of Symbols

Symbol	Property
Ω	target geometry
$\partial\Omega$	boundary of the target geometry
$\hat{\Omega}$	computational domain (unit quadrilateral in $\mathbb{R}^n$)
$\mathbf{x}$	mapping operator from $\hat{\Omega}$ onto Ω
$\mathbf{x}_h$	numerical approximation of $\mathbf{x}$
Ξ	(univariate) knot-vector
σ	univariate spline basis
p	polynomial order
$N_{i,p}$	i-th spline function in σ with polynomial order p
Σ	bivariate spline basis
Σ_0	bivariate spline basis comprised of inner basis functions only
w_i	i-th spline function in Σ
θ	rotational angle
(ξ, η)	tuple of free topological variables
(ξ', η')	reparameterization function in ξ and η direction, respectively
$\mathbf{c}$	vector of inner control points
$\mathbf{d}$	vector of boundary control points
$\mathbf{R}(\partial\Omega, \mathbf{d})$	projection residual
K	number of planar slices
L	spacing between the slices used to build an interpolation function
$\boldsymbol{\alpha}$	vector of shape parameters

Subscript	Property
k	k-th index in the total number of planar slices K
(s, e, n, w)	south, east, north, west
(m, f)	male, female