Spline-based meshing techniques for industrial applications

Hinz, J.; Möller, M.; Vuik, C.

Publication date
2020

Document Version
Final published version

Published in
Proceedings of the 6th European Conference on Computational Mechanics

6th European Conference on Computational Mechanics (ECCM 6)
7th European Conference on Computational Fluid Dynamics (ECFD 7)
1115 June 2018, Glasgow, UK

SPLINE-BASED MESHING TECHNIQUES FOR INDUSTRIAL APPLICATIONS

J. Hinz[1*], M. Möller [1] and C. Vuik[1]

[1]Delft University of Technology, Delft Institute of Applied Mathematics (DIAM) Mourik Broekmanweg 6, 2628 XE Delft, Netherlands
{j.p.hinz*, m.moller, c.vuik}@tudelft.nl

Key words: Isogeometric Analysis, Elliptic Grid Generation, (Re-)parameterization

Abstract. Isogeometric Analysis (IgA) has become an accepted framework for the modelling, simulation and optimization (MSO) of engineering processes. However, the fully automatized generation of analysis-suitable parameterizations of geometries as they arise in practical workflows is still a challenging task, which often requires application-specific parameterization approaches.
In this article we present a practical approach [6] based on the principles of *Elliptic Grid Generation* (EGG) for the efficient on-demand generation of analysis-suitable spline-based parameterizations. Starting from a (point cloud) description of the boundary provided by the existing MSO-pipeline, an inverse nonlinear Poisson-type problem is solved to obtain a folding-free (planar) parameterization of the entire domain. The non-linearity is efficiently treated with a globalized hierarchical Newton approach. Automatized boundary contour reparameterization techniques are employed to improve the parametric properties from a numerical viewpoint, such as orthogonal isolines and equally-sized cells. The use of curved instead of straight-sided elements allows us to arrive at an accurate description of the target domain with fewer elements and thus potentially lower computational effort.
Numerical experiments with screw-compressor geometries demonstrate that the proposed algorithm reliably produces high-quality parameterizations typically within $3-4$ Newton-iterations, even in the presence of extreme aspect-ratios. This makes it particularly attractive for the on-demand application within an automatized industrial MSO-pipeline.
To support the demands of modern high-performance computing hardware, only a moderate number of sufficiently large and structured patches is generated which can be mapped one-by-one to the different devices (CPUs/GPUs) with only little communication overhead. Topology changes are avoided in time-dependent and shape-optimization settings.
Finally, the spline-based geometry description can be transformed into a classical mesh by performing a large number of function evaluations in the mapping operator. Its continuous nature allows for feature-based (structured and unstructured) refinement and arbitrary element densities without increasing the complexity of the meshing process.

1 INTRODUCTION

Since the onset of Isogeometric Analysis (IgA) [7], spline-based parameterization approaches have received an increasing amount of interest. Generally, the CAD-pipeline only provides a spline-description of the contours $\partial\Omega$ of the target geometry Ω, so the goal is to generate a bivariate parameterization by appropriately selecting the inner control points of the mapping function $\mathbf{x} : \hat{\Omega} \rightarrow \Omega$. Upon completion, $\mathbf{x}$ is used to perform a *pull-back* of the PDE into the computational domain where it is solved using standard FEM-techniques. Even though in our applications $\partial\Omega$ generally comes in the form of a discrete point cloud input, a continuous contour description, as is mandatory in IgA-applications, can be achieved by performing a regression with a $p \geq 2$ spline-basis. Besides yielding the many advantages of a continuous geometry description, this may reduce the required amount of elements for an accurate approximation of Ω compared to an (exact or approximate) linear fit of the input data. As a result, the minimal inner element density is reduced, too. We are particularly interested in the application of

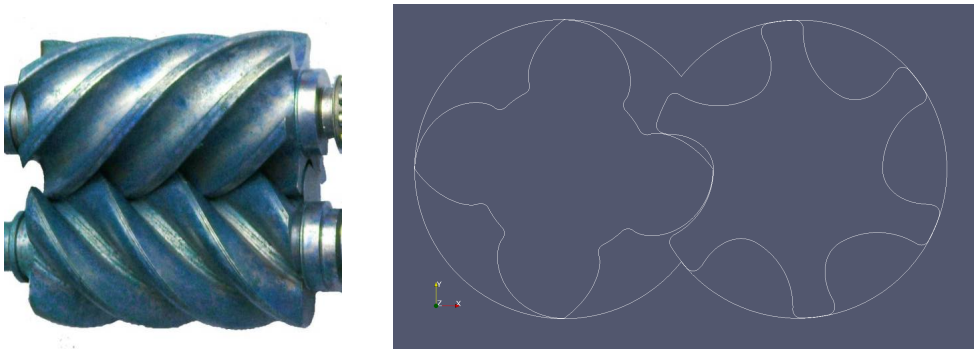


Figure 1: Rotary-screw compressor (Wikipedia, file: *Lysholm_screw_rotors.jpg*) (left) and a cross section showing the rotor profiles with casing (right).

spline-based parameterizations to rotary-screw compressor geometries (see figure 1, left). The 3D-geometry can be *sliced* into many planar geometries by taking cross sections (see figure 1, right). They, in turn, can be parameterized in the plane and stacked in order to retain a volumetric parameterization. The precomputed database of slices can be used to generate volumetric parameterizations with a variable rotor-pitch. The steepness of the rotor pitch can be controlled by a wider or tighter stacking of the slices. This enables shape-optimization on the pitch-function.
The individual slices (in particular the fluid-part between rotors and casing) feature many challenging characteristics such as tiny clearances and rapidly changing gap sizes. To the best of our knowledge, there exist two commercial mesh generators: Twin-Mesh [10] and SCORG [8], both operating with classical (i.e., straight-sided) structured grids.
While these software-packages reliably produce analysis-suitable classical grids, we prefer spline-parameterizations since we have found their continuous nature to greatly facilitate our pursuit for full automation of the MSO-pipeline. Even though most IgA parameterization methods rely on the minimization of a quality functional [5, 11], we utilize

a PDE-based approach that leads to a root-finding problem $\mathbf{F}(\mathbf{c}) = 0$ which can even serve as an additional constraint within the shape-optimization formulation. As a result, both the geometry generation and the solution of the PDE-problem can be tackled with the same numerical technique (i.e., IgA), leading to a symbiosis of both processes and reducing the software demands. Finally, the full system, comprised of quality-functional optimization and root-finding, can be differentiated symbolically with respect to the shape parameters. This facilitates the application of gradient-based optimization algorithms.
Outside of the IgA-context, the mapping $\mathbf{x}$ can be transformed into a classical (structured or unstructured) mesh with arbitrary element density and/or feature-based refinement by performing a large number of function evaluations. It can be used to produce grids with elements whose number exceeds the cardinality of the spline-basis by several orders of magnitude without adding to the computational costs of the meshing process.
The reader not familiar with the basics of (B-)spline functions is referred to [7]. In sections 2 to 6 we introduce the features of the algorithm, while in section 7, we will focus on its applications to rotary-screw compressors.

2 ELLIPTIC GRID GENERATION

In order to produce analysis-suitable parameterizations for geometries Ω from no more than a description of $\partial\Omega$, we employ the technique of *Elliptic Grid Generation* (EGG). Assuming Ω is topologically equivalent to the unit quadrilateral $\hat{\Omega} = [0,1] \times [0,1]$, there exists a bijective mapping $\mathbf{x} : \hat{\Omega} \rightarrow \Omega$ that satisfies $\mathbf{x}|_{\partial\hat{\Omega}} = \partial\Omega$. The goal of EGG is to yield a PDE-based equation for $\mathbf{x}$ that guarantees its bijectivity while featuring favorable parametric properties from a numerical perspective, such as evenly-spaced grid lines and equally-sized cells. To this purpose, the Laplace equation is imposed on the components of $\mathbf{x}^{-1}$, with $\mathbf{x}^{-1}|_{\partial\Omega} = \partial\hat{\Omega}$. Denoting the free topological variables of $\mathbf{x}^{-1} = \boldsymbol{\xi}(x,y)$ by the tuple (ξ, η), the equation takes the form

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

Equation (1) is scaled and inverted to yield an equation for $\mathbf{x}(\boldsymbol{\xi})$ that is suitable for a computational approach.The resulting equations read [1]

$$\begin{cases} L(x,y,x) &= 0 \\ L(x,y,y) &= 0 \end{cases} \quad \text{s.t. } \mathbf{x}|_{\partial\hat{\Omega}} = \partial\Omega, \tag{2}$$

where

$$L(x,y,z) = g_{22}z_{\xi\xi} - 2g_{12}z_{\xi\eta} + g_{11}z_{\eta\eta}, \tag{3}$$

with $g_{11}(x,y) = \|\mathbf{x}_\xi\|^2$, $g_{12}(x,y) = \mathbf{x}_\xi \cdot \mathbf{x}_\eta$ and $g_{22}(x,y) = \|\mathbf{x}_\eta\|^2$.
The imposition of the Laplace equation on the entries of $\boldsymbol{\xi}(x,y)$ rather than $\mathbf{x}(\xi,\eta)$ follows from the observation that the target space of $\boldsymbol{\xi}$ is always convex. As a result, the exact solution of (2) is bijective [1]. Therefore, any sufficiently accurate computational approach with the purpose of approximating $\mathbf{x}$ by a discretized mapping operator $\mathbf{x}_h$, also guarantees the bijectivity of $\mathbf{x}_h$.

3 DISCRETIZATION

Traditionally, (2) is approximately solved using a finite-difference approach [9]. A collection of grid points $(\xi_i, \eta_j),\ (i,j) \in \{1,\dots,n\} \times \{1,\dots,m\}$ is selected from the unit quadrilateral $\hat{\Omega}$, and the second order derivatives in (2) are replaced by a central approximation. The objective of the numerical approach is to approximate $\mathbf{x}(\xi_i, \eta_j) \equiv \mathbf{x}_{i,j}$. The resulting nonlinear equations is tackled with a Picard-based iterative approach.To acquire a (B-)spline-based description of our target geometry Ω, we could perform a spline-fit of the parametric values (ξ_i, η_j) against the grid points $\mathbf{x}_{i,j}$. However, we have noted the following shortcomings:

1. With $p > 1$ interpolation bases, bijectivity is often lost even though a $p = 1$ spline-fit leads to a classical mesh without self-intersections.

2. The input point cloud or a piecewise-linear regression thereof dictates the amount of unknown grid points and, by that, the complexity of the approach. Meanwhile, the required amount of degrees of freedom (DOFs) for a very accurate spline fit to the $\mathbf{x}_{i,j}$ tends to be much lower.

Therefore, we would like to devise a numerical scheme in which the spline-fit is carried out sooner rather than later in the process. To this purpose, we first discretize (2) with a Galerkin-approach. We select a $p \geq 2$ B-spline basis $\Sigma = \{w_1, \dots, w_n\}$ with global C^1-continuity. Defining $\Sigma_0 = \{w_i \in \Sigma \mid w_i|_{\partial\hat{\Omega}} = 0\}$, the discretization is carried out as follows:

$$\forall w_i \in \Sigma_0 : \begin{cases} \int\limits_{\hat{\Omega}} w_i L(x_h, y_h, x_h) \mathrm{d}\boldsymbol{\xi} &= 0 \\ \int\limits_{\hat{\Omega}} w_i L(x_h, y_h, y_h) \mathrm{d}\boldsymbol{\xi} &= 0 \end{cases}, \quad \text{s.t. } \mathbf{x}_h|_{\partial\hat{\Omega}} \simeq \partial\Omega, \tag{4}$$

where $\mathbf{x}_h = (x_h, y_h)^T$ denotes the approximation of $\mathbf{x}$. Let $\mathcal{I}_0 = \{i \in \mathbb{N} \mid w_i \in \Sigma\}$ and $\mathcal{I} = \{i \in \mathbb{N} \mid w_i \in \Sigma \setminus \Sigma_0\}$; then $\mathbf{x}_h$ takes of the following form:

$$\mathbf{x}_h = \underbrace{\sum_{i \in \mathcal{I}_0} \mathbf{c}_i w_i}_{\text{unknown}} + \underbrace{\sum_{j \in \mathcal{I}} \mathbf{d}_j w_j}_{\text{known}}. \tag{5}$$

Here, the boundary control-points $\mathbf{d}_j$ follow from the approximation of $\partial\Omega$ (see Section 4). The discretization from (4) leads to a nonlinear root-finding problem of the form $\mathbf{F}(\mathbf{c}) = 0$, where the vector $\mathbf{c}$ contains the unknown inner control points $\mathbf{c}_i$.

4 CONTOUR APPROXIMATION AND CHOICE OF BASIS

As stated in Section 1, we are predominantly working with point cloud inputs. Therefore, $\partial\Omega$ is only discretely available, which necessitates an exact or approximate spline-fit to the data. Given the four input point clouds $P_\alpha = \{\mathbf{p}_\alpha^i\}_{i=1}^{I_\alpha},\ \alpha \in \{\text{s, e, n, w}\}$ corresponding to each of the four sides of $\partial\Omega$, we first select four monotone increasing sequences $\{\xi_\alpha^i\}_{i=1}^{I_\alpha},\ \alpha \in \{\text{s, e, n, w}\}$, each starting on $\hat{\xi}_\alpha^1 = 0$ and ending on $\hat{\xi}_\alpha^{I_\alpha} = 1$. Let

$\mathbf{m}_s(\xi) = (\xi, 0)$, $\mathbf{m}_e(\xi) = (1, \xi)$, $\mathbf{m}_n(\xi) = (\xi, 1)$ and $\mathbf{m}_w(\xi) = (0, \xi)$. We select a coarse initial basis $\Sigma_\square$ and minimize the functional

$$R(\partial\Omega, \mathbf{d}) = \frac{1}{2} \sum_{\alpha \in \{\text{s,e,n,w}\}} \sum_{i=1}^{I_\alpha} \left\| \mathbf{x}_h(\mathbf{m}_\alpha(\xi_\alpha^i)) - \mathbf{p}_\alpha^i \right\|^2 \tag{6}$$

over the vector $\mathbf{d} = (\ldots, \mathbf{d}_j, \ldots)^T$, while constraining the corner control points to the corners of the input point cloud to avoid mismatches. Unfortunately, the boundary l_2-projection from (6) tends to be unstable when a basis resulting from two arbitrary knot-vectors is taken. This is usually a result of the local amount of DOFs exceeding the amount of points. We stabilize (6) by adding a least-distance stabilization term, leading to the following minimization problem:

$$\frac{1}{2} \sum_{\alpha \in \{\text{s,e,n,w}\}} \left(\sum_{i=1}^{I_\alpha} \left\| \mathbf{x}_h(\mathbf{m}_\alpha(\xi_\alpha^i)) - \mathbf{p}_\alpha^i \right\|^2 + \lambda \int_{\gamma_\alpha} \left(\hat{\nabla} \mathbf{x} \cdot \hat{\mathbf{t}} \right)^2 \mathrm{d}\boldsymbol{\xi} \right) \to \min_{\mathbf{d}}, \tag{7}$$

where $\hat{\nabla}$ denotes the nabla-operator in local coordinates and $\hat{\mathbf{t}}$ the unit tangent vector. By γ_α, we denote the subset of $\partial\hat{\Omega}$ that corresponds to side α. Here $\lambda > 0$ is a small penalty term whose value should be chosen small enough to not noticeably alter the outcome of (7) while avoiding instabilities.

The l_2-projection residual can, in turn, serve as a local refinement criterion. Knots can be added to the knot-vector(s) wherever the local residual exceeds the approximation tolerance $\epsilon > 0$. We recommend initializing the procedure with uniform knot-vectors while halving the size of elements on which the residual is too large. This ensures that the knots are not arbitrarily placed, which has several advantages that will become apparent in Section 7.

5 COMPUTATIONAL APPROACH

Upon reaching the approximation tolerance $\epsilon > 0$, we are in the possession of coarse and fine bases $\Sigma_\square$ and Σ with corresponding boundary control points. This leads to two root-finding problems $\mathbf{F}_\square(\mathbf{c}_\square) = 0$ and $\mathbf{F}(\mathbf{c}) = 0$. An initial guess $\mathbf{c}_\square^0$ for the coarse problem is constructed using transfinite interpolation [2]. The coarse problem $\mathbf{F}_\square(\mathbf{c}_\square) = 0$ is solved utilizing a truncated Newton-approach. Instead of performing the full Newton-step after each iteration, it is truncated with a damping factor $\delta \leq 1$ whose optimal value is estimated from the current and updated tangents and residuals. Upon convergence, we are in the possession of the coarse-grid mapping operator $\mathbf{x}_\square$. Its control points are prolonged to Σ, and the discrepancy between the prolonged coarse grid boundary control points and the $\mathbf{d}_i$ corresponding to $\mathbf{x}_h \in \operatorname{span} \Sigma$ is imposed as a Dirichlet boundary condition to a linear-elasticity [3] problem acting on the prolonged coarse grid geometry. Upon solution of the resulting linear system, the $\mathbf{d}_i$ coincide with the boundary control points of the manipulated coarse grid mapping. Its inner control points then serve as an initial guess for the fine-grid root finding problem $\mathbf{F}(\mathbf{c}) = 0$.

In practice, the nonlinear coarse-grid problem typically converges within $4 - 6$ iterations,

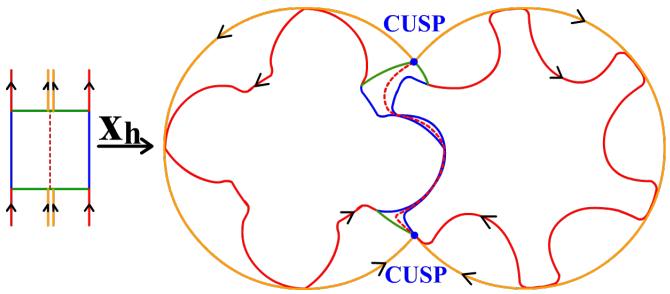


Figure 2: Planar cross section of the twin-screw setup and the aimed-for topology

whereas the fine-grid problem requires only $2-3$ iterations. For more information about the solution process, see [6].

6 PARAMETRIC MATCHING

In Section 4, we discussed how the input data is fit to the employed spline-basis. Given an input point cloud $P_\alpha = \{\mathbf{p}_\alpha^i\}_{i=1}^{I_\alpha}$, we have to select the corresponding parametric values $\{\xi_\alpha^i\}_{i=1}^{I_\alpha}$. By default, they are chosen as a cord length parameterization corresponding to P_α, which shall henceforth be denoted as $\mathbf{C}(P_\alpha) = \{\hat{\xi}_\alpha^i\}_{i=1}^{I_\alpha}$. In the presence of extreme aspect ratios (narrow gaps), however, we have found a naive cord length parameterization to lead to unsatisfactory results.
Let P_α and P_β be two input point clouds corresponding to opposite sides. Whenever a tuple of points $(\mathbf{p}_\alpha^i, \mathbf{p}_\beta^j) \in P_\alpha \times P_\beta$ is close (in terms of Euclidean distance), it is important that the same parametric value is assumed on both sides. Therefore, we apply the matching algorithm proposed in [6]. Upon completion, we are in the possession of a number of matching tuples $\mathcal{I}^m = \{(i,j) \mid \mathbf{p}_\alpha^i \text{ and } \mathbf{p}_\beta^j \text{ have been matched}\}$ with $\{(0,0),(I_\alpha,I_\beta)\} \subseteq \mathcal{I}^m$. We assign the parametric value $\hat{\xi}_\alpha^i$ to index j whenever $(i,j) \in \mathcal{I}^m$. For the best result, we match $\hat{\xi}_\alpha^i$ with the parametric value $\hat{\xi}_\beta^j$. This leaves us with the set $\mathcal{I}^\xi = \{(\hat{\xi}_\beta^j, \hat{\xi}_\alpha^i) \mid (i,j) \in \mathcal{I}^m\}$. Finally, we use a cubic monotone interpolation [4] of the matched $\hat{\xi}_\beta^j$ versus the corresponding $\hat{\xi}_\alpha^i$ and evaluate it in the $\{\hat{\xi}_\beta^i\}_{i=1}^{I_\beta}$ to yield the $\{\xi_\beta^i\}_{i=1}^{I_\beta}$.

7 APPLICATIONS

As stated in Section 1, we would like to apply the discussed techniques to the geometry from figure 1 (right), which represents a cross section of a rotary twin-screw compressor, where the two rotors are counter-rotating and form a very small gap in the middle chamber. It should be noted that the rotors never touch so that the physical topology does not change over time. Our goal is to automatically create multi-patch parameterizations for all cross sections, all having the same topology, namely the one in figure 2 (left). Figure 2 (right) reveals that, besides the input boundary contours, we require two curves for each CUSP-point, connecting them to both rotors. The resulting region shall henceforth be referred to as the *separator*. The dotted splitting-curve that divides the separator into

two parts constitutes the final ingredient.
Thanks to rotational symmetry, the starting point is again reached after a rotation of $\theta = \pi/2$ instead of 2π on the left rotor. In the following, θ will always refer to the rotation angle of the left rotor such that the tuple (θ_l, θ_r) is given by $(\theta_l, \theta_r)(\theta) = \left(\theta, -\frac{2}{3}\theta\right)$. Let m be the number of (uniformly spaced) discrete angles from the interval $[0, \pi/2]$ at which we aim to parameterize the target geometry. We first generate two O-grids of the separate rotors with casing at all m discrete angles. We start by generating cord-length parameterized exact spline-fits for both rotor (R_r, R_l) and casing (C_l, C_r) point clouds, denoted by $\mathbf{c}_{l,r}(\xi)$ and $\mathbf{r}_{l,r}(\xi)$, respectively. The functions $\mathbf{c}_{l,r}$ and $\mathbf{r}_{l,r}$ are then evaluated in N equally-spaced points over the parametric interval $[0, 1]$. The resulting point clouds serve as an input for the parametric matching procedure discussed in Section 6. Keeping the parametric properties of the casing unchanged (i.e., the casing corresponds to P_α in Section 6), the matching algorithm produces reparameterization functions $\xi'_{l,r} : [0, 1] \to [0, 1]$ that improve the parametric properties at the narrow gaps. If N is small, the matching will be cheap but the result will be worse while the converse holds for N large. At $\theta = 0$, the functions $\mathbf{c}_{l,r}$ and $\mathbf{r}_{l,r}$ are evaluated at n uniformly-spaced points $\{\xi_i\}_{i=1}^n$. This serves as to improve the stability of the contour approximation (see Section 4). Unlike the original input point clouds, which stem from external tools and cannot be controlled, the points will be roughly equally-spaced. By $\mathbf{P}_\alpha$, we denote the tuple of input points with corresponding parametric values for side α. The contour approximation is initialized with the coarse basis $\Sigma_\square$ possessing a periodic knot-vector in the η-direction with left and right point clouds

$$\text{left: } \begin{cases} \mathbf{P}_w &= \left(\{\mathbf{r}_l(\xi_i)\}_{i=1}^n, \{\xi'_l(\xi_i)\}_{i=1}^n\right) \\ \mathbf{P}_e &= \left(\{\mathbf{c}_l(\xi_i)\}_{i=1}^n, \{\xi_i\}_{i=1}^n\right) \end{cases} \quad \text{right: } \begin{cases} \mathbf{P}_w &= \left(\{\mathbf{c}_r(\xi_i)\}_{i=1}^n, \{\xi_i\}_{i=1}^n\right) \\ \mathbf{P}_e &= \left(\{\mathbf{r}_r(\xi_i)\}_{i=1}^n, \{\xi'_r(\xi_i)\}_{i=1}^n\right) \end{cases}. \tag{8}$$

For O-grids we disregard $\mathbf{P}_n$ and $\mathbf{P}_s$ and do not add any further constraints.
For angles $\theta \neq 0$, we act with the canonical rotation matrix on the control points of the $\mathbf{r}_{l,r}$. Let

$$a_{l,r} = (\xi')^{-1}\left(\tfrac{\theta_{l,r}}{2\pi}\right) \pmod 1 \tag{9}$$

and

$$\tilde{\xi}_{l,r}(\xi) = \xi \mp a_{l,r} \pmod 1. \tag{10}$$

In order to retain the same parametric properties as for $\theta = 0$, we form the new rotor point clouds by evaluating $\{(\mathbf{r}_{l,r} \circ \tilde{\xi}_{l,r})(\xi_i)\}_{i=1}^n$ and assigning to them the parametric values

$$\xi^i_{l,r} = \left(\xi'_{l,r} \circ \tilde{\xi}_{l,r}\right)(\xi_i) - \left(\xi'_{l,r} \circ \tilde{\xi}_{l,r}\right)(0) \pmod 1, \quad i = 1, \ldots, n \tag{11}$$

(see figure 3). The casings are kept unchanged.
As a result, the grid is held fixed at the casings while the rotors slide along (see figure

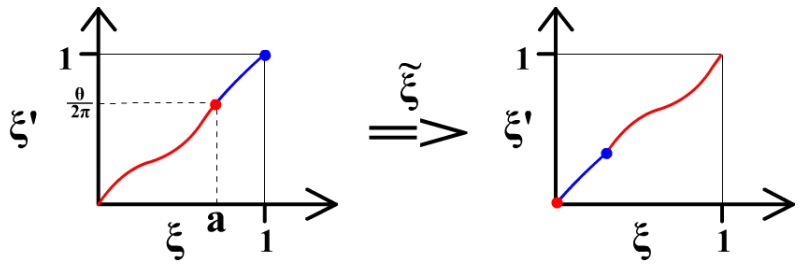


Figure 3: For $\theta \neq 0$, the reparameterization function needs to be shifted.

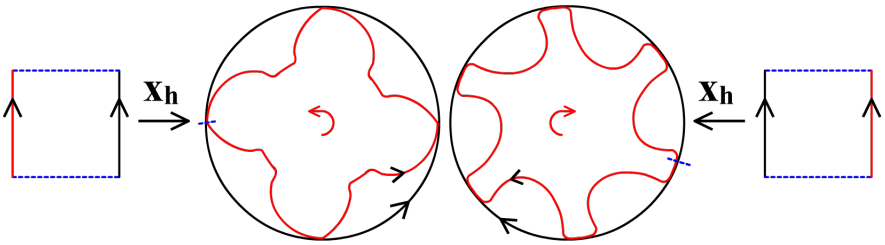


Figure 4: The grid is attached to the casings (black) while the rotors slide along (red).

4). This way, we avoid a sliding interface within the separator region of figure 2 (dotted red line). With the principles of Section 5, a mapping for the first 6 angles is computed. Upon completion, the mappings of the previous 6 angles are prolonged to one unified knot-vector and used to build a 5-th order extrapolation for the current angle. The extrapolated mapping is restricted to the current knot-vector and the inner control points are taken as an initial guess for the root-finding problem at the current angle.
Upon completion, we are in the possession of m mappings for the separate rotor O-grids. If not yet present, $p+1$ knots are added to the O-grids at the η-values that correspond to both CUSP-points. In this way, the O-grids are each split into two parts, one of which is a C-grid (see figure 5 left). Cutting the O-grid along the parametric lines in the ξ-direction (assuming the O-grid is bijective), ensures that the resulting curve never intersects with the rotor contour a second time (which could not be prevented else since the rotor contours do not fence off a convex region). The other pieces originating from the cut are turned back into a uniformly-spaced point cloud and combined into one boundary point cloud that serves as an input for the separator (see figure 5 right). As a result, we are in the possession of m boundary contour descriptions for the separator at each discrete angle. Choosing some $1 < k \ll m$, we again run the matching algorithm from Section 6 for each k-th contour, yielding the reparameterization functions $\xi'_i, i = 1, k, \ldots, m$ (keeping, for instance, the left side fixed). Upon completion we construct a global reparameterization function $\xi' : [0,1] \times [1,m] \rightarrow [0,1]$. This is accomplished by imposing $\xi'(\xi, i)|_{i=j} = \xi'_j, \forall j \in \{1, k, \ldots, m\}$, while smoothly interpolating the ξ'_i for the remaining angles using smooth blending functions. We do not build a reparameterization function at each discrete angle due to its discrete nature: since the algorithm is based on the matching of points, consecutive contours may have a different matching pattern and can therefore exhibit quite different parametric properties. Since we would like to achieve some degree

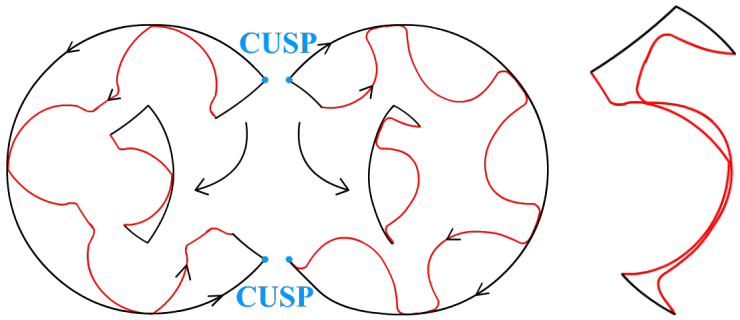
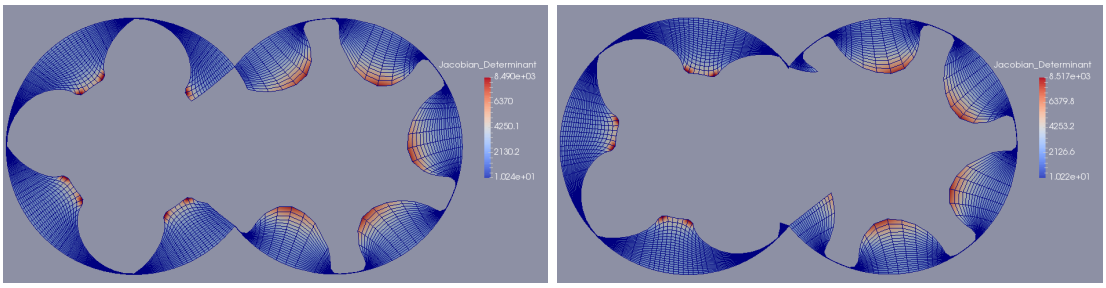


Figure 5: The two separate O-grids are each cut into a C-grid and the remaining part between the CUSP-points (left). Combining the remaining rotor parts with the cuts yields a boundary description for the separator (right).

of continuity in the angle θ, we overcome this downside by blending the discrete data.
As a next step, we compute parameterizations for each k-th geometry using the reparameterization function ξ' in conjunction with the principles of Section 5. Upon completion, our database is partially filled with parameterizations. These parameterizations, in turn, are prolonged to one unified knot-vector after which a 5-th order interpolation of the control points is performed in θ. As in the case of the O-grids, we build an initial guess for the remaining geometries by restricting the inner control points of the interpolation function to the current knot-vector and combining them with the corresponding boundary control points.
The boundary elements of the separator parameterizations formed this way are not conforming to the boundary elements of the two C-grids. To simplify the implementation of the solution coupling between the different patches, we favour the two parameterizations along a shared interface to be the same. Furthermore, the separator is parameterized by only one patch with global $C^{\geq 1}$-continuity. We would like to parameterize it using two patches with a C^0-continuity by the CUSP-points.
Therefore, we generate separating curves that split the separator into two parts (dotted red line in figure 2). We walk across the domain from the local counterpart of one CUSP-points to the other, leaving a splitting curve in our wake. Thanks to the continuous nature of our geometry description, this does not have to be a straight line but can be any non self-intersecting curve. The curve is selected such that its image passes the small gap in the middle with equal distance to the two rotor profiles. Upon completion, we are in the possession of m splitting-curves. They, in turn, are combined with the separator contours to form two new contours for the two-patch parameterization of the separator for each discrete angle.
Again, the parametric values assigned to the input point clouds are based on the principles of Section 6 while keeping the splitting curve arc-length parameterized in both cases (to ensure that the elements on the left and on the right side are conforming). Finally, a database of parameterizations for both sides of the separator are computed using the same approach as for the single-patch parameterized separator.



(a) The two C-grids at $\theta = \theta_1$. (b) The two C-grids at $\theta = \theta_{100}$.

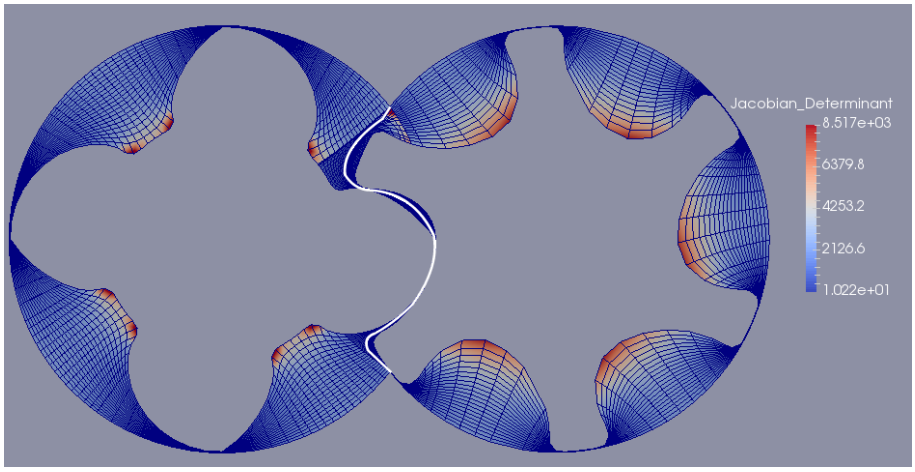
Figure 6

8 RESULTS

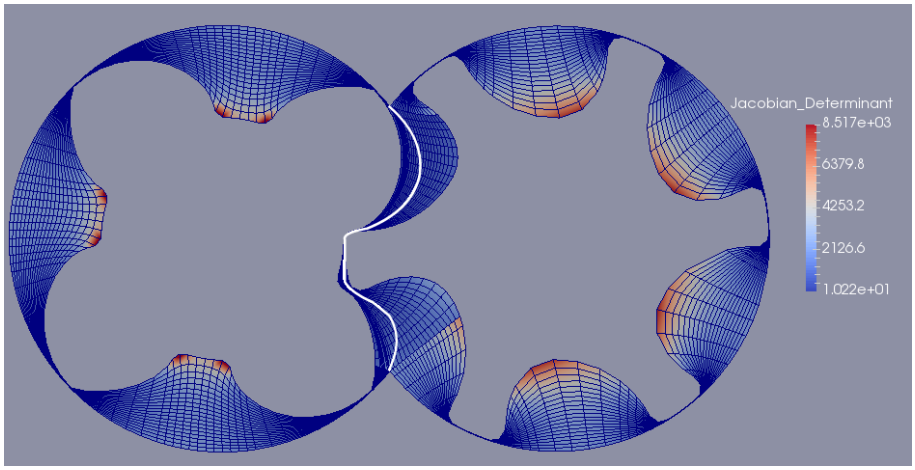
With the principles of Section 7, we generate parameterizations for $m = 200$ discrete angles θ_i. After the first 6 parameterizations have been completed, extrapolation reduces the amount of iterations required until convergence to only one. Figure 6 shows the C-grids acquired upon cutting the O-grids after $\theta = \theta_1$ and $\theta = \theta_{100}$. As a next step, we generate single-patch parameterizations of the separator for every 5-th ($k = 5$) discrete angle and build an interpolation function that serves as an initial guess for the remaining angles. In our experience, the algorithm typically converges after one iteration with a maximum of two.
Figure 7 shows the C-grids with the single-patch separator and the computed splitting-curve at θ_1 and θ_{100}.
Finally, figure 8 shows the two-patch parameterization of the separator along with its parametric properties by the splitting curve.

9 CONCLUSION

In this article, we have presented a PDE-based method for the parameterization of complex geometries using $C^{\geq 1}$-continuous spline-functions. We have successfully applied it to a planar rotary-screw compressor geometry, being able to parameterize it for 200 discrete angles from the interval $[0, \pi/2]$ (after which the initial position is reached again). This shows that the algorithm is particularly well-suited to handle the challenging features of geometries of this type, including extreme aspect-ratios. Furthermore, it is worth noting that the parameterization (apart from setting input parameters) did not require any manual interaction, making it a promising approach for the fully automated parameterization of rotary-screw type geometries. On the other hand, figure 8 reveals that there still exist elements with steep angles at the interface between separator and C-grids. We may conclude that the properties of the splitting-curve have to be optimized further in order to make a trade-off between the steepness of the angles at the C-grid to separator interface and the angles made at the two-patch separator interface.
The feasibility of the approach is significantly improved by database-driven inter- and extrapolation principles, owing largely to the continuous nature of nested spline-spaces. Furthermore, the generation and subsequent optimization of the splitting-curve was made

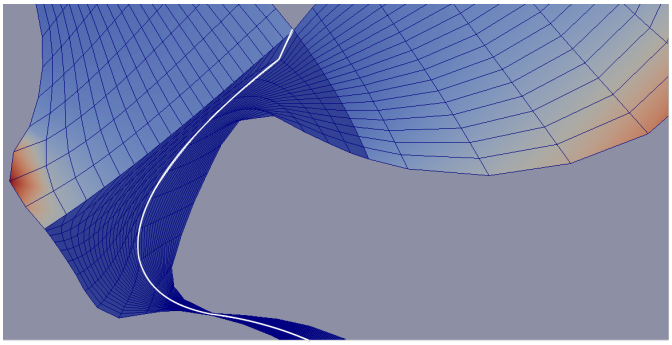


(a) The 3-patch parameterized geometry at $\theta = \theta_1$.

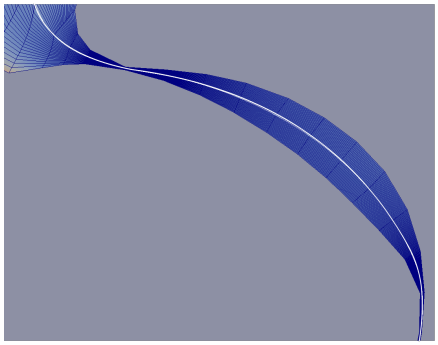


(b) The 3-patch parameterized geometry at $\theta = \theta_{100}$.

Figure 7



(a) The conforming two-patch separator.



(b) A zoom-in on the narrow part.

Figure 8

possible by their powerful properties.
Finally, it remains to mention that the proposed computational approach is not limited to rotary-screw compressor geometries but may serve as a general parameterization tool for similar settings, particularly in those that require the generation of a large number of mappings.

10 ACKNOWLEDGEMENTS

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

REFERENCES

[1] B. N. Azarenok. Generation of structured difference grids in two-dimensional non-convex domains using mappings. *Computational Mathematics and Mathematical Physics*, 49(5):797–809, 2009.

[2] S. A. Coons. Surfaces for computer-aided design of space forms. Technical report, DTIC Document, 1967.

[3] R. Falk. Finite element methods for linear elasticity. 1939, 01 2008.

[4] F. N. Fritsch and R. E. Carlson. Monotone piecewise cubic interpolation. *SIAM Journal on Numerical Analysis*, 17(2):238–246, 1980.

[5] J. Gravesen, A. Evgrafov, D.-M. Nguyen, and P. Nørtoft. Planar parametrization in isogeometric analysis. In *International Conference on Mathematical Methods for Curves and Surfaces*, pages 189–212. Springer, 2012.

[6] J. Hinz, M. Möller, and C. Vuik. Elliptic grid generation techniques in the framework of isogeometric analysis applications. *Computer Aided Geometric Design*, 2018.

[7] T. J. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. *Computer methods in applied mechanics and engineering*, 194(39):4135–4195, 2005.

[8] SCORG. Screw compressor rotor grid generator. `http://pdmanalysis.co.uk/scorg/`. [Online; accessed 2018-03-24].

[9] J. F. Thompson, B. K. Soni, and N. P. Weatherill. *Handbook of grid generation*. CRC press, 1998.

[10] Twin-Mesh. Twin-Mesh Software. `https://www.twinmesh.com/`. [Online; accessed 2018-03-24].

[11] G. Xu, B. Mourrain, R. Duvigneau, and A. Galligo. Parameterization of computational domain in isogeometric analysis: methods and comparison. *Computer Methods in Applied Mechanics and Engineering*, 200(23):2021–2031, 2011.